Sociological Theory

Global Labor: Algocratic Modes of Organization*

A. Aneesh
University of Wisconsin–Milwaukee

This study investigates a practice that allows workers based in India to work online on projects for corporations in the United States, representing a new mode of labor integration. In the absence of direct bureaucratic control across continents, the question arises how this rapidly growing labor practice is organized. The riddle of organizational governance is solved through an analysis of software programming schemes, which are presented as the key to organizing globally dispersed labor through data servers. This labor integration through programming code is distinguished from two other systems of organization—bureaucracy and the market—while bringing out the salient features of each system in terms of its ruling principle: bureaucracy *(legal-rational), the* market *(price), and* algocracy *(programming or algorithm). The logic of algocratic systems is explored methodically to analyze global work.*

This study inquires into a practice that allows workers based in India to work online on projects for corporations in the United States,[1] representing a new mode of labor integration. In the absence of direct bureaucratic control across continents, this rapidly growing, but understudied, labor practice raises a question about its mode of organization: How is this distributed work governed? As globally distributed work does not take place within a single firm, it cannot be governed through usual mechanisms. This study attempts to solve the riddle of organizational governance by demonstrating how global work is governed through the design of the work process itself, focusing on the role of software code as the key to governing globally dispersed labor through data servers.

With globally accessible data servers, this form of labor organization covers a staggering array of work activities: research and development in software, insurance claims processing, accounting, data entry, transcription and translation services, customer interaction services, geographic information systems, animation, data conversion, financial and credit analysis, engineering and design, website development and maintenance, remote education, market research, documentation handling, tax preparation, and human resource services like employee benefits and payroll. There are more than 800 firms in India that provide information technology (IT) and information technology enabled services (ITES) to corporations in the United States and other countries (Nasscom 2006). In 2005–2006, over a million workers were employed by the Indian IT industry whose revenue from foreign sources jumped from US$ 9.55 billion in 2002–2003 to US$ 36 billion in 2005–2006 (Nasscom 2006). The

*Address correspondence to: A. Aneesh, Bolton Hall 710, P.O. Box 413, University of Wisconsin–Milwaukee, Milwaukee, WI 53201. Tel.: (414) 229-2234; Fax: (440) 399-9335. E-mail: aneesh@uwm.edu. Research for this study was funded by the MacArthur Foundation, Population Council, and the Social Science Research Council. I would like to thank Erica Bornstein, Jozsef Borocz, Lee Clarke, John Levi Martin, Eviatar Zerubavel, and anonymous reviewers for their immensely helpful comments.

[1]While India and the United States are the basis of this case study, this labor practice goes beyond the confines of specific nations.

Sociological Theory 27:4 December 2009

National Association of Software and Service Companies (henceforth, Nasscom) claims to be a "truly global trade body with around 850 members, including 150 global companies from the US, UK, EU, Japan and China" operating from India (Nasscom 2006). By inquiring into this labor practice, this article links larger issues of globalization, labor flows, and nation-states to the micro-practices of organizational governance.

ORGANIZATION MATTERS

Recent years have witnessed an array of discussions centered around six kinds of global flows: finance capital (Castells 1996; Singh 1999; Stiglitz 2002), commodities (Gereffi and Korzeniewicz 1994; Krugman 1995), cultural forms (Appadurai 1990; Lechner and Boli 2005; Barker 1997; Featherstone 1990; Jameson and Miyoshi 1998; Morley and Robins 1995; Robertson 1992; Tomlinson 1999; Ritzer 2003), corporations (Barnet and Cavanagh 1994; Dicken 1992; Dunning 1992; Sklair 1998), migrant labor (Castles and Miller 1998; Massey 1998; Sassen 1988), and governance standards (Brubaker et al. 2004; Meyer 1980, 2000; Sassen 1998). However, there has been surprisingly little discussion of emergent organizational systems that make such flows and circulations possible.[2] To complement such substantive analyses, I inquire into organizational modes of globalization.

With enough bandwidth, labor in India has been systemically integrated into corporate sites across the globe, reducing an enormous spatial gulf to a matter of faster or slower transmission speeds. How does one make sense of this development? Is this a new kind of bureaucratic integration (intrafirm) whereby workers located in faraway places can work without physically crossing the borders? Or, is this market integration (interfirm) a new form of international trade[3] (Gross 2006) confirming an old economic principle that if a service is produced at a lower cost by another firm or country, it makes no economic sense to produce it domestically? Relying on the work of Coase (1937), economic analyses tend to confine themselves to these two alternatives of bureaucracy and market: "either a firm makes a component itself (bureaucratic governance) or it buys it from an autonomous supplier (market governance)" (Williamson 1981:10). Thus, firms emerge when transaction costs of market contracting exceed those of bureaucratic governance.

In the case of global software development, however, both alternatives fail to fully capture the emergent reality. While the bureaucratic model may loosely be applied in cases of subsidiaries (e.g., Adobe U.S.A. and Adobe India), there is rarely any bureaucratic integration possible for global software projects due not only to distance but also to country-specific labor regulations. It is difficult to have a single bureaucratic structure to govern different teams located in different regions of the world. As an alternative, one may then be tempted to suggest that it is a simple case of market-based outsourcing where the market replaces bureaucracy as a governance mechanism. Instead of developing the entire project in-house, the medium of money is used to attach a price to the works of different teams around the globe—a practice captured in terms of trade. The market ends up governing these projects by driving inefficient firms out of business. While this argument is valid regarding

[2]Some notable exceptions are Knorr Cetina and Bruegger (2002), Latham and Sassen (2005), and Sassen (2006).

[3]In 2004, N. Gregory Mankiw, Chairman of President Bush's Council of Economic Advisors, said that outsourcing was "just a new way of doing international trade. More things are tradable than were tradable in the past. And that's a good thing" (Gross 2006).

the dimension of price-based subcontracting, it fails to account for its collaborative character focused on the same project. These are not discrete exchanges conducted at arm's length presupposed in market transactions but collaborative ones on a shared project often with a single deadline. Neither the market (discrete transactions) nor bureaucracy (managerial oversight) is sufficient to explain the organizational reality of globally distributed software development. The third way, network forms of organization (Powell 1990), captures well the initial phase—the formation of initial business ties—of this practice, but it, too, covers only a minor aspect of the developmental phase. The question arises: What additional organizational mechanisms inform the emergence of collaborative projects at the global level occurring in parallel, sequence, or real time? Who manages these projects? This study identifies an additional system of governance, termed algocracy. To bring out its salience, this code-based governance system is distinguished from two better-known sociotechnical systems of organization—bureaucracy and the market, while underscoring the prominent features of each organizational form in terms of its ruling mechanism: bureaucracy (legal-rational), the market (price), and algocracy (programming or algorithm).[4]

The algocratic system of governance consists of programming schemes embedded in global software platforms that structure possible forms of work performance. This system enables the monitoring of work through the design of the work process itself. To give a small, quotidian illustration, while filling in the "fields" on a computer screen, a bank teller cannot type in the wrong part of a form, or put the address in the space for the phone number. The embedded code provides existing channels that guide action in precise ways.

The thesis of algocracy thus contributes to the sociological literature on the substantive consequences of technical decision making. Starting with the classic example of assembly line systems where the machinery itself is made to direct the labor process and set the pace from factories to fast-food joints (Braverman 1974; Edwards 1979; Leidner 1993), or how a nonhuman structure like the speed bump (a sleeping cop) contains within it a "motive," delegating a social rule to a device (Latour 1994), or the prioritization of music in octaves by the volt-per-octave standard adopted in the Moog synthesizer (Pinch and Trocco 2002), sociology has long been concerned with social implications of technical decisions. Implications of algocracy for labor control are in step with above scholarship, as it privileges higher level executives whose preferences may be preprogrammed in algocratic platforms. Adding to this general line of thought, I intend to extend the thesis of code beyond its local uses—technical control in organizations—in order to characterize it as a generalized medium like money, with equally far-reaching implications. Code emerges in this account as a language with money-like liquidity, acting as the medium of different kinds of communication: audio, visual, and also monetary. The genealogy of algocracy could be traced as far back as the proposal by Gottfried Leibniz, a 17th-century philosopher and mathematician, to build machines for deducing valid references through a "calculus of reason" (*calculus ratiocinator*). Despite the long history of symbolic logic, it was not until the Cold-War-related American defense strategies that programming acquired its practical importance (Abbate 1999; Aneesh 2001).

To distinguish algocracy from bureaucratic and market systems, this article extends the insights available in theories of differentiation and self-referential systems

[4]While the idea of code as a ruling mechanism has been used in law (Lessig 1999; Reidenberg 1998), its use in studies of organizations and globalization has been limited. Also, the concept of algocracy was developed independently by the author in 1997 in a graduate qualifying paper, and later presented at the American Sociological Association (Aneesh 1999).

(Alexander and Colomy 1990; Luhmann 1984). Modern bureaucracy—as Max Weber (1978) detected—operated on the basis of a legal-rational code that reduced the discretionary power of office holders. Unlike previous bureaucracies of Egypt, China, and medieval Europe that employed old modes of trust and hierarchies of Stände, Weber discerned the primacy of juridical formalism in modern bureaucratic systems. While scholars have questioned the Weberian thesis by showing how real bureaucracies were fraught with informal relationships and how formal rules were often inefficient and dysfunctional (Merton 1968; Selznick 1980), they confused between the system's internal code and environmental pressures, or its operational closure and structural openness. For a system to increase its internal complexity, it must be open to environmental stimulations (e.g., by social life) but it must perform its operations in its own language through its own code. While social and informal action surely has some bearing on a bureaucratic system, informality is not the code by which action could be legitimized within bureaucratic systems. A firm manager may recruit her nephew or friend as an employee but she must not justify the recruitment on the basis of kinship or friendship norms; she must present her decision in terms of merit and credentials. Indeed, norms of kinship and friendship are recoded as nepotism and cronyism in bureaucratic organizations. While kinship and friendship networks may exist in real bureaucracies, they lose their validity basis with the ethical neutralization of lifeworld perspectives.

Similarly, market systems may also be informed by social networks (Granovetter 1985); yet, in systems theory, the market behaves as a differentiated system precisely because it no longer operates through social code despite its apparent embeddedness in social life. Luhmann (1984) resolves, if only implicitly, the dilemma of embedded and disembedded action through a unique strategy: once a system differentiates and turns autonomous (or gets disembedded), it becomes a recursively closed circuit of communication, delimiting itself from its environment in a self-referential manner. This does not imply that the economic system operates in a social vacuum. The informal social world acts as the presupposed and necessary environment for the economic system, which constructs itself from moment to moment by creating a boundary between itself and its environment. It responds to the changes occurring in its environment by registering those perturbations in its own language. A friendly social alliance may operate in the market system but it must eventually be reduced through the code of payment/nonpayment to be part of market transactions. Market systems build their complexity from an array of stimulations—social, political, legal, scientific, or psychic—that must be converted into the system's own language for them to be meaningful. Harry's desire for a cold, refreshing carbonated drink on a hot summer day may be seen as a psychosomatic or cultural need but his desire is registered by the market system only as a specific consumer demand, which can be meaningful to the system not in psychosomatic but in economic terms.

While algocracy may appear to have bureaucratic structures embedded in it (e.g., legally permissible operations for a teller or the greater access to the same transaction available to the manager), the underlying software program is driven by the algorithm, or more deeply, the binary code. Imperatives of programming are not bureaucratic but mathematical even while a programmer codes bureaucratic controls in a software system. Algocracy may encode not only bureaucratic but also nonbureaucratic, less hierarchical governance as seen in peer-to-peer programming schemes or open-source development projects. The notion of algocracy thus implies "rule of the algorithm" or "rule of the code." Elsewhere, I have separated the notion of algocratic

systems from surveillance systems[5] (Aneesh 2006). By encoding a variety of tasks and relationships into digital schemes, programming code enables the emergence of a global organizational space, the subject of this study. Before further elucidating the notion of algocracy, it would be expedient to describe the empirical research that informs this argument.

FIELD RESEARCH

For my analysis of globalization and labor integration, I draw from two projects, the first of which was conducted in 1999–2000 with a focus on software labor flows from India to the United States, and the second, focused on call center work in India, was completed in 2004–2005. In 1999–2000, the author conducted 12 months of ethnographic field research in India, consisting of multiple visits to 20 small, mid-size, and big software firms located in New Delhi, Gurgaon, and Noida—a northern hub of software development in India. The selection of software firms was facilitated by a directory published for the region by the Nasscom. Given the dearth of sociological research on globally operating software firms, field research afforded the empirical basis for the analysis of screen-mediated social action in actual work settings. It also brought out the importance of algorithms, programming, and software applications for even those forms of work—for example, work carried out by call centers—that seem to lack an immediate relation to software development. This research presence in the field generated richer material and allowed a more nuanced analysis than would have been possible by use of standardized interview tools. Without the concrete observation of work processes, it would not have been possible to understand, for instance, the integrative mechanisms of transnational software platforms that allowed two or more teams based in different locations to work on the same project. Indeed, the notion of "algocratic" integration that I developed later owes its origin to my ethnographic observation in various offices. In addition to the physical presence in software development sites and corporate centers where work processes were closely observed, this study is informed by interviews, analysis of virtual software platforms, and other forms of data such as annual reports of Nasscom, reports regarding the IT task force of the Government of India, local technology magazines, and the Indian Information Technology Act 2000.

Of the 20 firms selected for study, and designated as "C1" to "C20," 13 were Indian-owned businesses while the other seven were foreign subsidiaries (Appendix A). Foreign investment in software has been an important factor since the IT industry began in India in the 1960s. Few U.S. technology firms lack presence in India. At the time of research, the firm size ranged from over 11,000 (C4) to less than 10 employees (C11), quite representative of a relatively young industry, which has at present escaped excessive consolidation and mergers. The largest firm, "C4," is one of the oldest Indian IT companies, employing over 24,000 persons in 2003, and servicing clients in over 55 countries. During field research, the number of visits to, and hours spent in, large firms such as C4 outnumbered by a good margin visits to smaller firms.[6] In keeping with the growth of the software industry in India, the size

[5]For those interested in the relationship between code and surveillance, Philip Agre (1994) provides an excellent discussion of two different models of surveillance based on visual and linguistic schemes.

[6]This is not to suggest that smaller firms are less important in a proper characterization of the IT industry in India.

and revenue of these firms has roughly doubled since this research was conducted in 2000. Big and mid-size firms (300 employees and above) tended to have many branches both inside and outside India that helped them coordinate their operations worldwide. While the majority of the clients of these firms were located in the United States, many of them also served clients in Europe, Japan, and the Middle East.

The data collected in 20 software firms allowed a cross-examination of ethnographic observation, interviews, and reports published by Nasscom or firms themselves. Ethnographic observation consisted of sitting with programmers or call center workers at their desks, observing and recording the method and content of their work. Furthermore, 100 formal and informal in-depth interviews were conducted with programmers, systems analysts, project managers, and call center workers, as well as high-level executives, including CEOs, managing directors, and vice presidents. Formal interviews produced more than 50 hours of recorded audio.[7] Although a comparative analysis of these software firms generated knowledge about variation among the firms, this article focuses more on what was common in terms of work practices in order to bring out the major contours of the labor regime. It is also perhaps misleading to call these firms "software" firms, as many of them provided not only programming labor to the firms in the United States but also, as mentioned earlier, other IT-enabled services—for example, telemarketing, accounting, and technical support—based on software applications running on globally accessible data servers. In addition, in 2004–2005, I studied five call centers, including one in-depth, participant observation at one mid-size international call center, employing about 1,000 employees in Gurgaon, a city with the largest cluster of international call centers in India. This ethnographic observation included participating in the training program of the call center as well as observations on the floor where live calls were made. This call center provided services for clients based in the United Kingdom and United States, specializing in telemarketing services. While illustrations of both software and call center work used in this article are relatively self-evident to practitioners of online labor, in their details they inform larger debates on global labor organization, transcending the specific site of inquiry.

The structure of the remainder of the article is as follows. First, I discuss why global computer work cannot be governed through regular mechanisms. After providing several reasons, I develop the idea of algocratic management where work is governed through the design of the work process. The governance mechanisms of algocratic systems are distinguished from bureaucratic and market systems. Under algocratic methods of governance, three processes of global work are discussed as parallel, sequential, and synchronous[8] collaborations. Thereafter, a detailed account of the limits of algocracy is provided, followed by a discussion of their increased importance in contemporary organizations.

HOW ARE GLOBAL COLLABORATIONS POSSIBLE?

The rise of global computer work begs the question why it is not feasible to monitor and regulate these projects through usual organizational mechanisms. Workers and

[7]Of the tape-recorded interviews, 24 were with programmers, 11 with systems analysts, 5 CEOs, 5 call center workers, 2 vice presidents, 2 managing directors, and 1 human resource manager.

[8]The terms—parallel, sequential, synchronous—are frequently used in computer science literature with respect to programming. The use of these terms in this article is broader, as they are also used in the sense of their original everyday meaning.

executives increasingly realize that three kinds of distance make it difficult for the global project to employ regular measures: legal, spatiotemporal, and cultural. There is no legally binding cross-national bureaucratic supervision available to globally distributed teams to monitor their workflows. Generally, global projects do not share organizational rules, the hallmark of bureaucratic governance. While all Indian firms have internal bureaucratic structures, they do not share their hierarchy with their American counterparts. Second, the spatial distance in global computer work does not allow for rich, multilayered, face-to-face interaction, a usual mechanism available in conventional bureaucracies to resolve conflict and miscommunication. In fact, spatial proximity is acutely missed by all parties involved in global collaborations. Raghav, a 34-year-old senior project manager at MetaTech, a software firm located in Noida, India, admitted that global projects lack common organizational mechanisms, "[in the U.S.] people can resolve things through meetings, through discussions, through lectures, through face-to-face interactions," something not available in the collaborative project between his Indian firm and its American client. The accountability of team members to one another, as Ó Riain (2000:185) suggests, is much more easily sustained in face-to-face interactions than in virtual communication. "There is something to be said about face-to-face communication," Vishal, a senior programmer at SynCo, emphasized, "being in face-to-face communication, and being part of the team physically is very different from being just online." Companies attempt to recover the loss of spatiotemporal immediacy by various mechanisms of social integration such as direct phone connections, video conferencing, electronic message boards, and instant messaging. Yet, in the process of overcoming spatial distance, these attempts run into the problem of temporal distance.

A temporal lag in communication threatens synchrony. Suresh Gupta, AlgoTech's CEO, complained of the slow and overpriced connections provided by the Indian government: "Our work—because we are completely integrated—requires something like 100 Mbps or 200 Mbps; what we get is 256K at an exorbitant price. And because it's going through a satellite, there is a delay. There is a ping[9] time of 300 or 400 milliseconds, which is too high for our kind of work. So, what we want is go all the way on fiber optics, so that we can ping at 7 or 8 milliseconds. That'll improve the response of a lot of our systems." Gupta's plans to install his own fiber optic lines may reduce the delay in real-time connections but they cannot overcome the lack of temporal immediacy because of time-zone differences.

Last, the question of cultural distance makes the bureaucratic integration of global collaborations improbable. A single organizational culture requires attitudes, values, and behavior to work in unison, an unlikely situation in global project management. Studying the relationship between the American company, Global, and its Indian counterpart, Shiva, Heeks et al. (2001:56) found that Shiva had a relatively personalized and subjective management culture while Global stressed objectivity and accountability: "It took enormous efforts before the Shiva project leader would produce a standardized monthly progress report, and Shiva staff refused to participate in Global's employee satisfaction survey." Indeed, the problem goes beyond organizational culture and touches on a larger cultural question. Many project managers repeated the importance of a general familiarity with American culture in order to understand and implement changes. Raghav pointed out, "the entry-level people are

[9]PING stands for Packet Internet Groper, a utility to determine whether a specific IP address is accessible. It works by sending a packet or datagram to the specified address and waiting for a reply.

the people who haven't been to the U.S. or abroad; they can't really relate," requiring the presence of a project manager who has physically worked in the United States and who can translate American requirements into a language understood by the average programmer in India.

Without a shared managerial layer, a shared set of rules, shared organizational culture, or even a shared language, the question—how global collaborations are possible at all—must be resolved through a closer scrutiny of the space of collaboration. Here one detects the first feature of algocracy: a virtualization of organizational space, a space that does not pretend to provide all the benefits of face-to-face interaction but a space that does allow a globally shared space to emerge. Karin Knorr Cetina and Urs Bruegger (2002) have introduced an appropriate distinction between *embodied presence* and *response presence*. While embodied presence is face-to-face, response presence describes responses to one another and common objects without being physically present in the same place. With virtual software platforms, it has become possible for workers, managers, and developers to watch the same screen and make changes to the same database. It is the screen's emergence that has made it possible for traders to watch the market simultaneously (Knorr Cetina and Bruegger 2002).

The second major feature of algocracy is programming, which allows the possibility of governing projects through code, implementing governance mechanisms in the design of the work process itself. In the absence of clear lines of command across collaborating firms, programming codes "command lines" that inform virtual platforms.

ALGOCRATIC SYSTEMS: GOVERNANCE THROUGH THE DESIGN OF THE WORK PROCESS

I will explain algocratic systems first at the level of software users, gradually moving up to software development. One of the software applications developed by Meta-Tech was a banking application. Installed on the distributed computer network of their client, an American bank, this banking application turned rules and routines into software code, structuring the work performance of a bank teller who was the final user in this case. To highlight how code guided and governed this work, I quote Amy, an American bank teller: "You log on, do your password, then your screen opens . . . there are functions on the top, that say twenty-one is a cash advance, twenty-two is . . . , and it does nothing until you put in the number for the transaction you're going to do. Then there is a list of the amount—is it cash, is it check, does he want cash back . . . and [the relevant screen] pops up; if it's over a certain amount, another screen pops up and says, did you check ID. So, it's pretty basic, it takes you step by step through the transaction. It says, now give the customer this much money, and asks you if this amount is correct. And so you fill in numbers for all the sections, hit enter, it will take you to the next step. You will validate the thing you're holding—the check, the slip, the transaction—and then it asks if there is anything else you want to do for the customer. And you say yes, or no." By embedding governance in the very design of work process, this code-guided structure demonstrates how the worker's subjective orientation to rules was made less relevant than following the steps suggested by the program, which tended to prevent other ways of doing work. With the development of distributed online banking, algocratic systems extend the organizational space directly to the customer, who can similarly

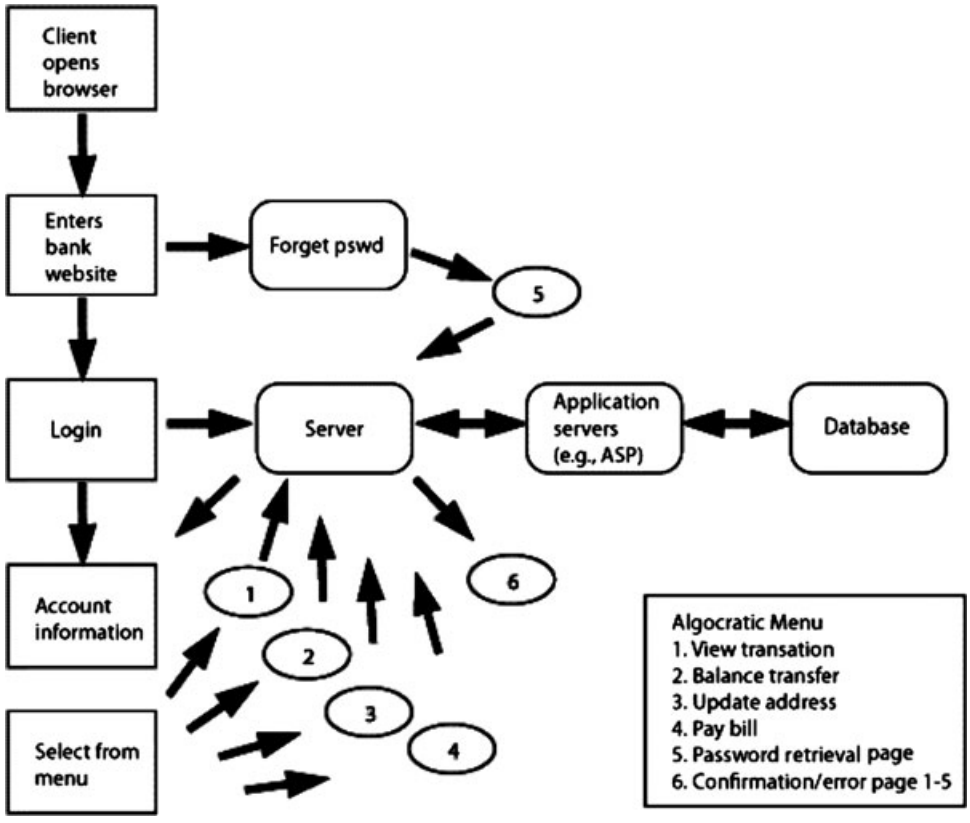

Figure 1. Algocratically governed everyday banking.

perform a variety of banking tasks from any location around the globe. Figure 1 gives a brief overview of algocratically managed space of virtual banking.

One may argue from Figure 1 that algocracy is merely the software version of bureaucracy. At MetaTech, many existing bureaucratic procedures were apparently translated into software systems, as reported by Raghav: "We support . . . banking applications like daily branch opening, account handling, money transfers, everything, the routine tasks for which there's a need to build the software. It's very routine because most rules are documented. You just have to implement those business rules into software programs." Thus, code appears to have merely taken over the managerial function of supervision and guidance, reducing the number of mistakes possible in various transactions. Conway in his classic article on software design hinted at the systems design closely following prior organizational context (Conway 1968). It may seem, then, that bureaucratic rules have been delegated to a code-based device, which implements those rules "on the fly." But algocracy is not bureaucracy for a clear reason: there is no common meta-language shared by bureaucratic (legal code) and algocratic (binary code) systems of governance. Earlier, I misrecognized algocracy as "hyperbureaucracy" (Aneesh 1999), which was also misrecognized as perfect law by Lessig (1999). While bureaucracy and algocracy may be the necessary environments for each other, the two systems of governance are structurally open but operationally closed, to borrow Niklas Luhmann's (1984) phrase. One can detect its structural openness in managerial insistence on replicating the previous work structure into

software systems despite systems analysts' contention about their inefficiencies. But when algocracy negotiates bureaucratic imperatives in its own language, it transforms them.

To separate the two from the commonly used action-theoretic perspective, bureaucratic governance is internally dependent on both *action orientation*,[10] requiring people to learn and embody the authority of laws, rules, and regulations, and *action consequence* when sanctions steer action through functional consequences. In bureaucracy, rule adherence is managed through socialization or training (action orientation), integrating the demands of rules into one's behavior, which acquires the willingness to distinguish between permissible and nonpermissible action. An added mechanism to ensure proper organizational behavior is incentive, which in its negative (penalty) and positive (rewards) forms drives action through its consequences. Market systems, on the other hand, require only action consequence to guide behavior through a price-based spur to action. In contrast, algocratic systems do not need either action orientation or consequence to coordinate action. Programming technologies seek to structure the possible field of action without a similar need for orienting people toward learning the legal rules, as demonstrated through the bank teller's statement. Action is controlled neither by socializing workers into regulatory demands, nor by punishing workers for their failure, but by shaping an environment in which there are only programmed alternatives to performing the work. Thus, work involves a lower focus on the knowledge of regulations and a greater stress on the ability to use a software program.

Thus, the notion of algocracy suggests that authority does not need legitimacy in the same sense because either there are no alternative routes to the permissible ones or the permissible routes are themselves programmed. There is no comparison that can be used to delegitimize authority, which is increasingly embedded in the underlying code,[11] rendering the hierarchical system of authority relations less useful. This is not to suggest that bureaucratic structures and rules have disappeared in the global firm; both systems may indeed exist simultaneously: the software firm's internal structure may look bureaucratic while its external collaboration may be managed through algocratic mechanisms. Table 1 is a stylized comparison to capture the ways in which bureaucratic, monetary, and algocratic systems are different from each other:

Whereas bureaucratic governance, as Weber (1978) has noted, is conducted through juridical formalism with associated rules and regulations, market exchange is governed through price with only two options: whether to buy the product and how much of it to buy. Algocratic governance, on the other hand, as explained earlier, is coded as a program, which automatically determines the range of possible action. In terms of operational code, bureaucracy operates by the permissibility or nonpermissibility of action according to written rules. While there may be ambiguity about the permissibility of a certain course of action, the ambiguity is usually solved by either making the rules clearer, more specific, or by incorporating the ambiguous as a version of the already defined. For a market event, the action must follow the code of payment or nonpayment, as all market transactions to be called as such must be accomplished through some sort of payment (Luhmann 1984). In contrast,

[10]In Luhmann's theory, the term "action" is actually an event that is interpreted as action by the system.

[11]Lawrence Lessig (1999) has used this feature of code to explain how the "fair-use" exception in copyright laws does not work in digitally managed copy protection schemes.

Table 1. Organizational Governance

	Systems of Organization		
Key Features	Bureaucratic	Market	Algocratic
Governance	Written rules (legal positivism)	Price	Program
Code	Permissible/not permissible	Payment/ nonpayment	True/false (1/0)
Medium	Routines	Money	Programming language
Labor integration	Hierarchicy	N/A (only indirect)	Network

algocratic systems use the logical code of truth values defined as true-not true or more conventionally 1 or 0 for their organizational constitution and complexity. For internal communication, algocratic systems use the medium of programming language in contrast to the use of money by market systems and of routines by bureaucracies. In terms of labor integration, hierarchical membership seems crucial to bureaucracies while the market integrates labor only indirectly (e.g., labor market). Algocracy, on the other hand, appears to connect through the structure of computer networks.

In an algocratically managed field of communicative network, information can flow directly from lower level units to top management, with a reduced need for middle managers, flattening bureaucratic hierarchies to a degree. Scholarly findings that computerization is correlated with fewer hierarchical levels (Cappelli 1992; Hodson 1985; Kanter 1989, 1991; Osterman 1994) do not necessarily mean looser control or the absence of governance. They point rather to a new differentiation in systems of governance whereby the specific model of hierarchy is neither needed to the same degree nor possible in work projects that are global in nature. "For the first time in history," Manuel Castells (1996:214) asserts, "the basic unit of economic organization is not a subject, be it individual (such as the entrepreneur, or entrepreneurial family) or collective (such as the capitalistic class, the corporation, the state)... the unit is the network." The notion of algocracy adds to this idea by bringing to light concrete governance mechanisms that are coded into network transmissions, as exemplified in horizontal, multisite, multicountry, modular software development and also network protocols themselves. In this context, the notion of algocracy is both limited and precise: it occupies only the middle layer in the information network composed of three layers (Benkler 2000): the layer of physical infrastructure layer (e.g., hardware, cables, satellites), the layer of logical infrastructure layer (e.g., code), and the layer of content. While programming, the middle layer, may influence to a degree both physical infrastructure (e.g., circuit boards)[12] and content layer (e.g., audio quality through compression algorithms), it is analytically distinct from the other two. It may encode a piece of music (content) as "MP3" and fix it into a physical medium (compact disk) but it is not music itself.

[12]Dodge and Kitchin 2005 use the term *coded infrastructures* to refer "both to networks that link coded objects and infrastructure that is monitored and regulated, either fully or in part, by code."

Moving up from the user (e.g., bank teller or customer) to the developer, one notices that a similar network-oriented algocratic system informs the work of software development. Virtual development platforms such as ClearCase, BSCW, MetaEDIT+, MultiSite, and more general-purpose technologies like CORBA produce an organizational space in step with Dodge and Kitchin's (2005) analysis of code-generated spaces through technicity[13] and transduction,[14] highlighting newer structures of governance. I discuss this in terms of parallel, sequential, and synchronous collaborations.

Parallel Collaboration

As noted earlier, algocratic systems are marked by two interrelated features: virtualization and programming. To overcome the loss of spatiotemporal proximity, this form of organization depends on virtual platforms, allowing far-flung teams to work on a project in parallel. IndiTech, a software firm in Noida, India, was involved in a collaboration with an American company. Its CEO, Gurdayal Singh, was a 50-year-old man with youthful enthusiasm who emphasized the idea of collaboration in his description: "There are twenty people working in the U.S. and twenty people in India. They are doing different things. But the mother ship is the same; it goes into the same product. So you are working on the same database, you are working on the same code. You are working on the same thing . . . sharing . . . a data server. Except for the fact that we are in India, we could be sitting across the room from those people and working." Each site might work on a different stage of a product's life cycle, or divide responsibility along developmental lines, with one working on the application engineering, including specifications analysis, design, and integration and acceptance testing, and another site implementing system components; or the work might be distributed according to the qualifications of the team at each site.

AlgoTech was also engaged in a collaboration involving four different teams: AlgoTech U.S.A., AlgoTech Germany, AlgoTech India, and an Indian software contractor. The project related to telephony software development, a complex area of research and development where standards and requirements changed rapidly, complicating the cross-site development. While the U.S. and German sites had existed for four to five years, and programmers had extensive experience working together, they had never worked on a project where parts of development were divided across sites. How does a virtual platform act as the organizational space for such a global project? Like many other software organizations, AlgoTech used ClearCase as a software development platform to create a virtual organizational unity through embedded controls in the work design. AlgoTech did not have a centralized bureaucracy with face-to-face or legally structured interaction to manage its globally dispersed teams in the United States, Germany, and India. It was the ClearCase platform that enforced a common work process and a common view of the project even if its success was mixed in practice. The common work view did not imply centralized development. Indeed, the notion of *view* in the algocratic platform of ClearCase took on a specific meaning. A view was an isolated "virtual workspace" that provided dynamic access to the entire data repository to programmers scattered in three different countries, as depicted in Figure 2.

[13]The productive power of technology to make things happen.
[14]The constant making anew of a domain in reiterative and transformative practices.

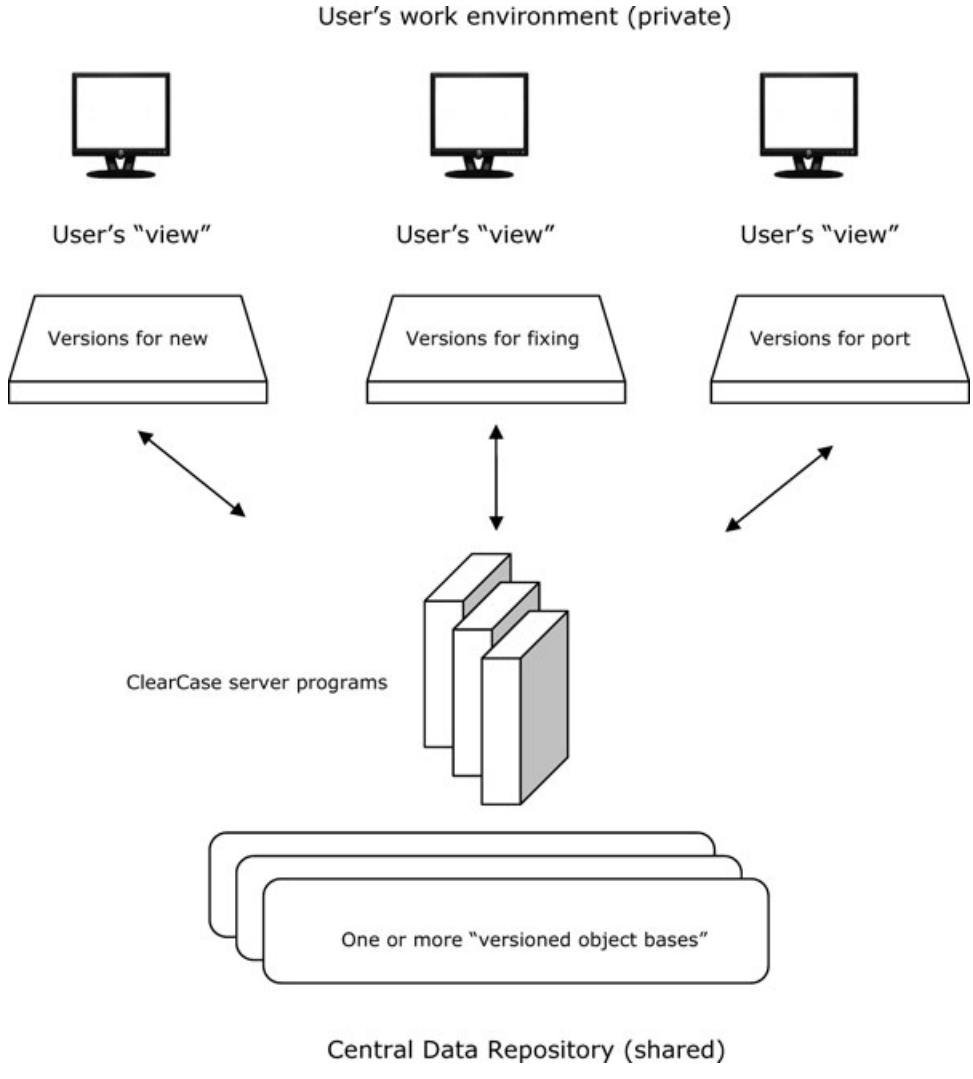

Figure 2. Views in ClearCase.

The changes made to a source file in a particular view were invisible to other views; software builds performed in a view did not disturb the development occurring in other views. But a view's isolation did not render it inaccessible. If needed, AlgoTech's global project members could share a view while working on single or multiple hosts. A 27-year-old programmer, Neeraj, said that he constantly stole a glance at his colleague's view to see the changes being made to a file relevant to his own work. When Neeraj made changes through a client program in ClearCase, the work design of the development platform automatically allowed records to be stored in versioned object bases (VOB) databases. When he checked out a file element in order to edit it, and eventually create a new version of it, an algocratic mechanism called cleartool automatically stored a checkout version event in the element's VOB. All project members across the globe could potentially access this event record, depending on the member's access rights. Furthermore, when Neeraj integrated the

work stored on a subbranch of an element back into the main branch, a merge arrow was created in the element's VOB database, logically connecting the merged versions. The presence/absence of merge arrows enabled globally dispersed teams to determine the work required to integrate the telephony project's branch-resident jobs back into the main line of development.

Unlike bureaucratic systems where legal rules and supervision would be used to control the work process, algocratic governance used the code itself to create process control schemes. With an access control component in the design of the ClearCase platform, AlgoTech limited the use of individual data objects to particular users or groups, and restricted the use of specific commands to authorized individuals and teams. The design of its development environment offered programmed flexibility. For example, a VOB could be opened up to AlgoTech's project members, without having to grant access rights to the "entire universe." Whereas access permissions worked mainly at the file-system level, locks protected all kinds of objects at the VOB-database level—elements, branches, branch types, even entire VOBs. A lock on individual VOB objects prevented AlgoTech's Indian subcontractor from making changes (unless they were put on an exceptions list). The effect of a lock was both small and large. While one lock prevented any new development on a particular branch of a particular element, another lock could be applied throughout a VOB. Locks were useful for implementing temporary restrictions. For example, during the integration period, a lock on a single object prevented the users outside the integration team from making any changes. The work design also ensured that the smallest change, or delta, to the program text (e.g., the deletion or addition of a single line of code) was tracked to synchronize the entire development with the latest version.

Algocratic systems avoid transferring entire files from one team to another, as often witnessed in bureaucracies, where files move from one section to another for comments, decisions, or signatures. Virtual platforms transfer only synchronization packets from site to site, reducing the load of network traffic created by remotely located clients. Synchronization capabilities enable each clone to be updated with the changes—and only the changes—that have occurred at every other remote site since the last synchronization. Thus, components on a distributed network coordinate their actions only by passing messages according to the rules set by algorithms. The concurrent components of distributed systems lack a global clock. They do not synchronize different elements of work in accordance with a single global time frame. As there is no single correct time, their synchronization only means the synchronization of packets achieved by passing messages in an algocratically defined field. Similar to parallel collaboration, algocratic systems were also used to govern sequential collaboration cycles across the globe.

Sequential Collaboration

In insurance claims processing, a claim must be entered into the system before verification, analysis, and approval may be carried out. Is it possible to split the tasks in a way that the first step may be completed in the United States, and the second in India? Indeed, many large insurance companies are able to divide the work in a sequential fashion with a front-end data entry in the United States and back-end processing in India. Using the 12-hour time difference between the United States and India, this approach allows work organization in different time zones to be sequentially patterned for competitive advantage.

This approach allows nonstop 24-hour work carried out by collaborating teams located in India and the United States. "Basically [when] it's night in the U.S., it's early morning here," Vishal, the programmer at SynCo India, explained. "At the end of their day [the Americans] just have to [compile] their problems and the changes they want us to do, and we can fix them in our normal working hours, fix them just in time, and it will be there next morning when they come to their office . . . The claims that are entered in the day [in the United States] . . . will be processed in the nightly batch cycles in Bombay. We actually have about sixty jobs running one after the other, which update the table information." In cases like this one, the time zone difference is an asset for the corporations involved. When the computer information control system is not in use in the United States, Indian workers can process solutions and complete them online. When the office opens in the morning in the United States, the back-end work has already been completed, thus creating a virtual 24-hour office for the American client. However, the temporal organization of labor must follow this pattern strictly if it is to function properly, as the team in India must finish all the tasks within a small window of time before the American side can begin their work. As in parallel work, code is used to create a work design that exercises control over the sequential processing of work. In insurance claims processing, the front end of algocratic system in the United States controls action in a manner similar to that of the bank teller described earlier, while the back-end batch processing application in India governs action by channeling, extracting, and validating information, for instance, about a vehicle identification number against data that are controlled by federal and state governments. In recent years, this form of work is being fully algocratized through software-oriented architecture, completing tasks through real-time web services.

A question remains about algocratic organization: In the absence of legal boundaries and physical location, how are the organizational walls of a process, simultaneously present in India and the United States, closed off? There is surely legal-rational, or bureaucratic, closure of such spaces through the national regulation of firms, but those legal-rational walls must remain open for transnational collaboration. An organizational space that spans continents must clearly be virtual in nature; so, too, its walls. Most intranets—virtual organizational spaces—are protected by a variety of algocratic walls: electronic firewalls, gateways, packet filters, and proxy servers. Firewalls, for example, can examine each message entering or leaving the network, and block those that do not meet the specified security criteria. Similarly, packet filters can look at each packet entering or leaving the network, and accept or reject it based on project-defined codes. Evidently, the security provided by these walls, quite like that provided by physical walls, may be compromised by various means, including viruses and hacking, but that is beside the point. Internet-based groupware packages let organizations build far-flung project teams, moving beyond managerial coordination and integration.

Sequential work designs include a variety of global applications. For instance, they can facilitate the generation of purchase orders, status monitoring, and delivery of goods for companies like the Gap that are directly connected with their globally scattered vendors. Khushwant Sinha, a software project manager at HTC in New Delhi, described how they helped the Gap to develop a new information system: "What Gap does is, like, all their clothes are produced in the Third World, Latin America, India, Bangladesh, and all these countries. They have vendors in all these places, so purchase orders are created between these vendors and Gap, and you want to purchase so many goods of a certain style, cut, size, and this order is

sent out to these vendors. So, the process of automation is purchase order creation, and then getting the goods back and things like that. We were involved in the development activity." A purchase order application installed on the machines of buyers and suppliers makes work progress and the status of orders available on everyone's screens in the network.

These algocratic platforms not only allow parallel and sequential work processes, they can enable real-time, synchronous work as well. Much of real-time customer service work across continents is managed algocratically through various software systems, including dialer software and customer relations management (CRM) systems.

Synchronous Collaboration

At CitiZen, an international call center in Gurgaon, customer service work usually began with customers' interaction with a voice-activated algocratic system, which routed inbound calls based on departments or employee skills, guiding the call through computer voice toward systematically preprogrammed and numbered options. To match and organize multitype customers and multiskilled agents, various queuing models and algorithms were used in the software for operating skills-based routing of calls within a call center. These algocratic systems may or may not work efficiently for the organization or its customers but their use has become pervasive.

For outbound calls, GoCom, a call center specializing in telemarketing services, managed the agent-customer interaction algocratically. Indeed, it was not the agent who dialed the phone number. It was the "Dialer" software that targeted specific profiles from the CRM database with strict parameters covering credit history, buying habits, and such demographic variables as age, gender, income, region, and education while also handling the do-not-call list. When Geeta sold mortgages to potential customers in the United States, the dialer skipped the profiles that were unlikely candidates for a loan.

The dialer resembled assembly line production systems at least in one respect: the pace of work was designed in the program itself. Raj, a 22-year-old agent, complained: "The dialer tells you to take the break . . . You get a break around 7 and you get a break around 9 for half an hour . . . And you are supposed to be logged in for at least 7 hours 45 minutes, overall. Because if you suddenly want to go to the loo, it just cuts down on your time." On an average, the dialer paced the work of agents by dialing 250–300 calls a night, requiring the agent to be always prepared for a possible conversation. The programming scheme of the dialer was based on predictive algorithms, ensuring that Raj only contacted live parties, discarding busy numbers, disconnected lines, and fax or answering machines. While the ability of predictive dialers to distinguish between an answering machine and a live person raised productivity by reducing the time wasted on busy signals, invalid numbers, and answering machines, it also intensified the agent's time use. Predictive dialers tend to dial ahead of the agent's actual availability, queuing up busy signals for automatic redial. In manual dialing, agents could relax and sit idle if the calls dialed ran into answering machines. The predictive dialer, on the other hand, enabled a work regime that connected the agents with live parties most of the time. All advanced predictive dialers algocratically controlled the ratio of calls to agents, altering their dialing rate according to the number of connections made while also adjusting to the behavior of the ongoing telemarketing campaign. The dialer generated statistics

about each agent's current and average call connection time for previous days and locations dialed. While this system is certainly demanding on the agent, it can also result in nuisance for the customer who on occasion may hear only silence when the dialer makes more calls than agents can handle.

LIMITS OF ALGOCRACY

Algocracy is necessary but not sufficient for virtual labor practice. The absence of face-to-face interaction continues to be its Achilles' heel, particularly in software development projects. Indian software professionals are often flown to the United States for a brief initial meeting, as it is not always possible for the client to formulate complete project specifications and communicate them online. "[When] you are developing a project, which interfaces to a lot of other projects within the U.S.," an informant noted, "those interfaces have to be studied there only . . . You can't just ship it here. If you ship it here . . . it's very difficult for the people to understand what the interfaces are; there [could be] ten different departments interfacing with this department." Similarly, at the end of the project, despite the online access to completed software projects, senior software engineers typically return to the United States to ensure successful implementation.

It is not surprising that many companies would often post a senior project manager on site in order to facilitate better communication between the geographically remote teams. Raghav recounted his experience: "Citibank had [changed] all their retail business; there were a lot of changes required in the [software] programs already existing . . . There was a team of people working in India, and there was a project manager on site [in Japan]. I was the project manager, I would take work from the Japanese managers and I would send it offshore to India . . . So any changes, any production problems, anything would immediately come to [notice through] me." To avoid coordination problems arising from geographical distance, Indian firms developed what was known as a 75-25 model. While 75 percent of their workforce remained in India, they established a small office in the United States that comprised the remaining 25 percent of the workforce. It coordinated and mediated between the Indian team and the American client. To provide 24-hour information systems management to a major American insurance company, IsomoTech followed the 75-25 model by setting up a small unit in North Carolina (U.S.A.).

Algocratic integration between India and the United States presents a problem of synchrony across sociocultural divides as well. It is not surprising that the Indian software industry accommodates many failed projects and abandoned transnational business relationships. When problems of social integration occurred, the solutions increasingly took a systemic turn. While it might be difficult to extract a programmer's performance report from an Indian project manager, it was possible to program some functions of monitoring into the system itself. For example, software development platforms acquired a defect-tracking function that could identify the number of errors made per thousand lines of code, producing evaluations of programmers' work without reports or managers.

In step with the findings of previous studies (Freeman 2000; Ó Riain 2000), resistance to certain practices by the staff was visible in many software companies. The main aggravation tended to be the requirement to adjust work hours, too early or too late, to create an overlap with American work hours, a requirement deeply resisted by programmers and project managers alike. AlgoTech's CEO expressed his frustration over the time zone difference between New Delhi and Seattle, which unlike

cities on the East Coast of the United States, offered no overlap with India in work hours: "In our case the problem is that it's exactly twelve and a half hours difference. So there are no common office hours. For example, if the office there started at our 4 p.m. or 5 p.m., I would be so much more happy. Because right now every meeting is an effort. On their part or on our part, both sides. Because we have to stay in office until 10 or 10:30 p.m. to have a reasonable meeting, and they need to start early, and vice versa." During daylight-saving time in the United States, the situation improves, but the U.S. West Coast and Delhi still do not have any temporal overlap, and this puts a heavy demand on programmers, project managers, and executives to work late at night or early in the morning so that they can find a temporal space for a virtual meeting. I noticed that temporal dissonance was always a contentious issue when setting deadlines or phone and video meetings. To overcome such problems, some companies opened a smaller branch outside the United States in the same time zone, for example, in the Caribbean, to gain temporal proximity without higher wage costs.

To a degree, the problem of insufficient overlap of office hours in the United States has been overcome by allowing night work. Call centers already work at night to serve clientele in the United States during their daytime, and there is no shortage of people looking for jobs that required them to adjust and synchronize their lives with regular consumer or work hours in the United States. As virtual integration reconfigures local contexts and the social times of people's lives, it raises significant questions about the meaning of life and work. Under "reversed temporalities of work," to borrow Winifred Poster's (2007) felicitous phrase, workers' integration into a global system effects a break with local mechanisms of social integration, which is usually achieved through temporal symmetry—a temporal coordination that makes communal life possible by organizing activities in the same time frame (Zerubavel 1981). In Indian companies, night work tends to put workers "out of sync" with their own society as they synchronize their life with work hours across the globe. Globalization tends to be a series of different effects in different places (Guillén 2001). After Tarun started working in a call center, he complained: "Hardly anybody recognizes you, nobody recognizes you [in your neighborhood]. You come in when nobody sees you. When you wake up in the evening, you see all your newspapers. At 8 PM, you pick up your newspapers when it's your good morning. You browse through the news for the day that happened yesterday . . . After a while you stop listening to the news, then you stop reading the newspapers. All you want to do is get your pillow and sleep; get up, go, make your calls, come back, and sleep." Spatiotemporal integration, at least in the context of night work, seemed at once spatiotemporal alienation. It must be emphasized that night work is not a new phenomenon. At this century's turn, two out of five employed Americans worked mostly at nonstandard times—in the evening, at night, on a rotating shift, or during the weekend (Presser 2003). People's integration into the 24-hour global economy is also a temporal unhinging of family life, as spouses are not together at home in the evening or at night, and parents are often not home with their children. It was not surprising that the majority of call center workers in India tended to be single and/or without children.

The rate of attrition among call center workers was high. Samir, another agent, recalled, "I think sixteen of us have quit, and two people are still there. We did not know the day we met, but we all ended up leaving. So it was good. I think everybody who stays on in a call center does not have an option . . . The day I quit, senior people who were working there for two years said, they also want to

quit, but they can't. I said, 'why' and they said, 'Where will we get a job?'" The skills learned at the international call centers in India were not transferable to any other industry due mostly to their "other-worldly" training in work processes, voice, accent, culture, and geography of another society. Customer service agents employed in U.S.-oriented processes were given American pseudonyms; they went through the voice and accent training programs, and were taught American history and geography. A few private institutes have opened in India that provide training to call center workers in the neutralization of regional accents. The systemic imperatives of the new global labor regime thus threaten to colonize social mechanisms by shaping the questions of speech, identity, knowledge, and time in their own terms. When temporally isolated pockets of social life are algocratically connected, it may have far-reaching significance in terms of the increasing submission of social life to economic imperatives.

THE SCOPE OF ALGOCRACY

Despite limits and problems associated with the rise of algocratic systems, this form of monitoring has continuously been gaining in importance in fields as diverse as global software development and distance education. With the development of virtual platforms like ClearCase, distributed software projects spanning multiple locations have increasingly become the norm rather than the exception in software development (ACM 2006). Many high-tech American and European firms (e.g., Lucent, Siemens, Motorola, and Nokia) connected their spatially dispersed teams as if they were located in the same building, and all major software companies have a significant presence in India for collaborative software projects (e.g., Adobe, Microsoft, Google, Yahoo, and IBM) in addition to Indian-owned software development firms.

One can also notice the growing importance of algocratic systems in fields other than software development, making the notion of algocracy portable across different domains. As a small example, I will touch upon a case of algocratic management in education where a variety of software platforms such as Desire2Learn (D2L) has emerged to provide education in a different way. D2L replaces the classroom experience of face-to-face interaction with a feature-rich algocratic platform that tends to simulate physical interaction through audiovisual lectures, discussion forums, announcements, and email. It enables an effective system of control through deadlines (e.g., time allotted to a quiz, or the period of availability of a particular discussion forum, or submission deadlines), as the system records by the minute the responses made and papers uploaded on the D2L web-based platform. It allows the instructor to employ the controls embedded in the design itself, opening or closing a project folder to particular students. Such algocratic platforms, quite like software development environments, offer refined access controls. Instructors can view all their courses on a single screen, and differentiate between individuals and groups, and set group restrictions regarding student projects within a single course.

Algocratic systems owe their rapid expansion to the malleability of code, which rivals modern money in its ability to convert a variety of exchange in its own language. The invention of modern money allowed market systems to gain unprecedented autonomy and closure (Luhmann 1984; Marx 1967; Parsons 1963; Weber 1978). With an ability to recode previously noneconomic environments into money-steered economic subsystems, the institutionalization of money was as important as the rise of the capitalist enterprise (Habermas 1988). It was modern money that helped recode labor as a commodity, enabling the institutionalization of wage-labor relations under

capitalism. As a result, it neutralized the lifeworld context of labor by rendering it abstract.

All-purpose money, quite like programming code, is a form of sign system or language. Modern money was created and multiplied at first when bankers discovered that they could make loans merely by issuing promises to pay (or banknotes) to borrowers. More notes could be issued than the gold and silver on hand for two reasons. First, notes were only "signs" or "numbers" printed on paper, readily available in contrast to the limited supply of gold. Second, only a fraction of the total notes outstanding would be presented for payment at a time, and thus needed to be settled. Reserves of metallic money were required only to cover that portion. This liquefaction of commodity money into a system of legible signs bears an important resemblance to software code. The notion of liquidity is important for both money and programming code. Different forms of money—M1, M2, M3[15]—in economics are defined according to degrees of liquidity. Cash, the most liquid variety of money, may be a mere sign, a number printed on paper and backed only by social promises (promissory note) but, being a number, and thus easy to carry, it becomes the easiest means of exchange, allowing commodities to float in a liquid medium.

Programming languages, too, act as a steering medium. The ability to break down, digitize, and then recode into a program a tax accountant's knowledge and skills, a civil engineer's mathematical and visual conception of a three-dimensional structure, an architect's drawing skills, and some managerial skills (e.g., probabilistic decision making) enhances the potential for rearranging elements of work in different configurations. With the liquidity gained through coding, software can be developed for both white-collar jobs and for the control of heavy industrial machinery, making computer-related occupations (e.g., network systems, data communication, software engineering) some of the fastest-growing occupations in the United States for past and coming decades (BLS 2008). The demand for computer-related occupations has increased in "almost all industries as organizations continue to adopt and integrate increasingly sophisticated and complex technologies" (Dohm and Shniper 2007). Like money, programming recodes different forms of work in its own language. Raghu, a hardware engineer, pointed out that the job of hardware engineer itself changed into "mostly a software job. They say you're a hardware engineer, but you're actually out there writing programs to build the chip." Instead of soldering iron and wires, he continued, "now you sit in the front of your computer and you have these little pieces of software, which help you lay down wires on your screen; you don't have a real wire in hand, and you put transistors here and wires there. Then you simulate it to see if it works." Thus, similar to monetization, programming languages have gained the ability to express qualitatively different things in their own language, recoding external distinctions as distinctions within their own system. Clearly, programmed texts are not passive symbols; they are action scripts, acting as machines[16] (simulated models of cars or space shuttles), organizations (enterprise systems or algocratic platforms), consumable commodities (music, movies, video games, or images), or even money (electronic money and credit money).[17] The intertranslatability, or liquidity,

[15]M1 is money that can be spent immediately; it includes currency in circulation and the checkable deposits in depository institutions (banks and thrifts). M2 includes M1 as well as assets invested for the short term, including money-market accounts and money-market mutual funds. M3 includes both M1 and M2 and such big deposits as institutional money-market funds and agreements among banks.

[16]Software patents often claim software to be a machine.

[17]By enabling credit cards to connect with databases, code allows credit to be used as a form of money, achieving high liquidity through instantaneous transactions.

triggered by programming languages contributes to the discussed symbolic traffic in global labor.

CONCLUSION

The global organization of labor, or the especial case of programming labor as discussed in this study, can no longer be captured in terms only of bureaucracy or the market; nor can it be fruitfully analyzed through the logic of social relations alone. The development of virtual work teams, commonly described through the market logic of subcontracting and outsourcing, presupposes a governance system—which I identify as algocracy—that operates through the logic of programming code. Global work cannot be governed through a single legal-rational frame of bureaucracy due partly to the existence of multiple legal regimes of nations and organizations, and partly to the lack of spatiotemporal and cultural proximity, often presumed in bureaucratic operations. While there is usually no shared organizational hierarchy even between a global corporation's subsidiaries in different nations, the bureaucratic model becomes completely inapplicable in cases of third-party outsourcing where the market model with its pricing mechanism (outsourcing) is generally used to explain the subcontracting relationship. Yet, the market model, too, fails to explain why these development projects are jointly developed, and not simply bought and sold in the marketplace. While the logic of market exchange is certainly operative in global work, as a business may "outsource" and "subcontract" work in order to reduce the size of its permanent workforce or capitalize on globally dispersed cheaper labor, an analysis based on market exchange leaves out the collaborative nature of these global projects as well as the reorganization of corporate structure itself. The product of these collaborations is not internally produced within a single organization. It is a jointly produced artifact. These global projects significantly differ from noncollaborative, arm's-length exchange assumed in market transactions. If customers call their bank in the United States and the phone rings at a call center in India that can provide various services by directly accessing the customer's accounts with the parent firm in real time through data servers, this signifies the development of a governance structure that spills over the framework of bureaucratic, market, or social network model.

Inserting the practice of online labor into traditional models—such as bureaucracy and outsourcing—makes it difficult to develop an understanding of emergent organizational forms that are increasingly shaped by possibilities of programming. The rise of programming code as a language of organization matches in importance the historical advent of all-purpose money in market systems. Thus, algocratic organization becomes possible due to an important development in the nature of labor itself: the liquefaction of concrete labor into digital code. As programming languages increasingly become new currencies of labor, they facilitate—like money—a diversity of flows by translating work into symbols, for example, images, sounds, texts, while also providing digitally coded governance structures. What has enabled labor thereby to increasingly become information labor is programming. In a certain sense, software companies in India are in the business of selling algocratic organization, complete with ready-made templates and modules for banking, managing the supply chain, payroll, job costing, sales force, product life cycle, and customer relationships. While code may not always succeed in organizing work as intended, its willing insertion into everyday business operations is pervasive and its usefulness in explaining not only global labor integration but also the decline of bureaucratic hierarchies is significant.

REFERENCES

Abbate, J. 1999. *Inventing the Internet*. Cambridge, MA: MIT Press.

ACM. 2006. "Flexible and Distributed Software Processes." *Communications of the ACM* 49:26–34.

Agre, P. E. 1994. "Surveillance and Capture: Two Models of Privacy." *Information Society* 10:101–27.

Alexander, J. C. and P. B. Colomy. 1990. *Differentiation Theory and Social Change: Comparative and Historical Perspectives*. New York: Columbia University Press.

Aneesh, A. 1999. "Technologically Embedded Authority: The Post-Industrial Decline in Bureaucratic Hierarchies." *Sociological Abstracts*, American Sociological Association.

———. 2001. "Skill Saturation: Rationalization and Post-Industrial Work." *Theory and Society* 30:363–96.

———. 2006. *Virtual Migration: The Programming of Globalization*. Durham: Duke University Press.

Appadurai, A. 1990. "Disjunctures and Difference in the Global Cultural Economy." *Public Affairs* 2:1–24.

Barker, C. 1997. *Global Television: An Introduction*. Malden, MA: Blackwell Publishers.

Barnet, R. J. and J. Cavanagh. 1994. *Global Dreams: Imperial Corporations and the New World Order*. New York: Simon & Schuster.

Benkler, Y. 2000. "From Consumers to Users: Shifting the Deeper Structures of Regulation." *Federal Communications Law Journal* 52:561–79.

Braverman, H. 1974. *Labor and Monopoly Capital: The Degradation of Work in the Twentieth Century*. New York: Monthly Review Press.

Brubaker, R., M. Loveman, and P. Stamatov. 2004. "Ethnicity as Cognition." *Theory and Society* 33:31–64.

Bureau of Labor Statistics (BLS). 2008. *Occupational Outlook Handbook*. Washington, DC: U.S. Bureau of Labor Statistics.

Cappelli, P. 1992. "Examining Managerial Displacement." *Academy of Management Journal* 35:203–17.

Castells, M. 1996. *The Rise of the Network Society*. Cambridge, MA: Blackwell Publishers.

Castles, S. and M. J. Miller. 1998. *The Age of Migration: International Population Movements in the Modern World*. New York: Guilford Press.

Coase, R. H. 1937. "The Nature of the Firm." *Economica* 4:386–405.

Conway, M. E. 1968. "How Do Committees Invent?" *Datamation* 14:28–31.

Dicken, P. 1992. *Global Shift: The Internationalization of Economic Activity*. London: Paul Chapman.

Dodge, M. and R. Kitchin. 2005. "Code and the Transduction of Space." *Annals of the Association of American Geographers* 95:162–80.

Dohm, A. and L. Shniper. 2007. "Occupational Employment Projections to 2016." *Monthly Labor Review* 130:87–125.

Dunning, J. H. 1992. *Multinational Enterprises and the Global Economy*. Reading, MA: Addison-Wesley.

Edwards, R. 1979. *Contested Terrain: The Transformation of the Workplace in the Twentieth Century*. New York: Basic Books.

Featherstone, M. 1990. *Global Culture: Nationalism, Globalization, and Modernity*. Newbury Park: Sage.

Freeman, C. 2000. *High Tech and High Heels in the Global Economy: Women, Work, and Pink-Collar Identities in the Caribbean*. Durham, NC: Duke University Press.

Gereffi, G. and M. Korzeniewicz. 1994. *Commodity Chains and Global Capitalism*. Westport, CT: Greenwood Press.

Granovetter, M. 1985. "Economic Action and Social Structure: The Problem of Embeddedness." *American Journal of Sociology* 91:481–510.

Gross, D. 2006. "Why 'Outsourcing' May Lose Its Power as a Scare Word." in *New York Times*.

Guillén, M. F. 2001. *The Limits of Convergence: Globalization and Organizational Change in Argentina, South Korea, and Spain*. Princeton: Princeton University Press.

Habermas, J. 1988. *The Theory of Communicative Action: Lifeworld and System: A Critique of Functionalist Reason*, Vol. 2. Boston: Beacon.

Heeks, R., S. Krishna, B. Nicholson, and S. Sahay. 2001. "Synching or Sinking: Global Software Outsourcing Relationships." *IEEE Software* 18:54–60.

Hodson, R. 1985. "Working in High-Tech: Research Issues and Opportunities for the Industrial Sociologist." *Sociological Quarterly* 26:351–64.

Jameson, F. and M. Miyoshi. 1998. *The Cultures of Globalization*. Durham, NC: Duke University Press.

Kanter, R. M. 1989. *When Giants Learn to Dance: Mastering the Challenge of Strategy, Management, and Careers in the 1990s*. New York: Simon and Schuster.

———. 1991. "The Future of Bureaucracy and Hierarchy in Organizational Theory." Pp. 63–90 in *Social Theory for a Changing Society*, edited by P. Bourdieu and J. Coleman. Boulder, CO: Westview.

Knorr Cetina, K. and U. Bruegger. 2002. "Global Microstructures: The Virtual Societies of Financial Markets." *American Journal of Sociology* 107:905–52.

Krugman, P. 1995. "Growing World Trade: Causes and Consequences." *Brookings Papers on Economic Activity* 1:327–77.
Latham, R. and S. Sassen. 2005. *Digital Formations: IT and New Architectures in the Global Realm*. Princeton, NJ: Princeton University Press.
Latour, B. 1994. "On Technical Mediation—Philosophy, Sociology, Genealogy." *Common Knowledge* 3:29–64.
Lechner, F. and J. Boli. 2005. *World Culture: Origins and Consequences*. Oxford: Blackwell.
Leidner, R. 1993. *Fast Food, Fast Talk: Service Work and the Routinization of Everyday Life*. Berkeley: University of California Press.
Lessig, L. 1999. *Code and Other Laws of Cyberspace*. New York: Basic Books.
Luhmann, N. 1984. *Social Systems*. Stanford, CA: Stanford University Press.
Marx, K. 1967. *Capital: A Critique of Political Economy*, Vol. 1. New York: International Publishers Co.
Massey, D. S. et al. 1998. *Worlds in Motion: Understanding International Migration at the End of the Milenium*. New York: Clarendon Press.
Merton, R. K. 1968. *Social Theory and Social Structure*. New York: Free Press.
Meyer, J. W. 1980. "The World Polity and the Authority of the Nation-State." Pp. 109–37 in *Studies in Social Discontinuity*, edited by A. Bergesen. New York: Academic Press.
———. 2000. "Globalization: Sources and Effects on National States and Societies." *International Sociology* 15:233–48.
Morley, D. and K. Robins. 1995. *Spaces of Identity: Global Media, Electronic Landscapes, and Cultural Boundaries*. London: Sage.
Nasscom. 2006. *Indian IT Industry: Nasscom Analysis*. New Delhi: National Association of Software and Service Companies.
Ó Riain, S. 2000. "Networking for a Living: Irish Software Developers in the Global Workplace." Pp. 175–202 in *Global Ethnography: Forces, Connections, and Imaginations in a Postmodern World*, edited by M. Burawoy. Berkeley: University of California Press.
Osterman, P. 1994. "How Common is Workplace Transformation and Who Adopts It?" *Industrial and Labor Relations Review* 47:173–88.
Parsons, T. 1963. "On the Concept of Political Power." *Proceedings of the American Philosophical Society* 107:232–62.
Pinch, T. J. and F. Trocco. 2002. *Analog Days: The Invention and Impact of the Moog Synthesizer*. Cambridge, MA: Harvard University Press.
Poster, W. R. 2007. "Saying 'Good Morning' in the Night: The Reversal of Work Time in Global ICT Service Work." *Research in the Sociology of Work* 17:55–112.
Powell, W. W. 1990. "Neither Market Nor Hierarchy: Network Forms of Organization." *Research in Organizational Behaviour* 12:295–336.
Presser, H. B. 2003. *Working in a 24/7 Economy: Challenges for American Families*. New York: Russell Sage Foundation.
Reidenberg, J. R. 1998. "Lex Informatica: The Formulation of Information Policy Rules Through Technology." *Texas Law Review* 76:553–84.
Ritzer, G. 2003. *The Globalization of Nothing*. Thousand Oaks, CA: Pine Forge Press.
Robertson, R. 1992. "Globalization: A Brief Response." *Journal for the Scientific Study of Religion* 31:319–23.
Sassen, S. 1988. *The Mobility of Labor and Capital: A Study in International Investment and Labor Flow*. Cambridge: Cambridge University Press.
———. 1998. *Globalization and its Discontents: Essays on the New Mobility of People and Money*. New York: New Press.
———. 2006. *Territory, Authority, Rights: From Medieval to Global Assemblages*. Princeton, NJ: Princeton University Press.
Selznick, P. 1980. *TVA and the Grass Roots: A Study of Politics and Organization*. Berkeley: University of California Press.
Singh, K. 1999. *The Globalisation of Finance: A Citizen's Guide*. London: Zed Books.
Sklair, L. 1998. "Globalization and the Corporations: The Case of the California Fortune Global 500." *International Journal of Urban and Regional Research* 22:195–215.
Stiglitz, J. E. 2002. *Globalization and its Discontents*. New York: W. W. Norton.
Tomlinson, J. 1999. *Globalization and Culture*. Chicago: University of Chicago Press.
Weber, M. 1978. *Economy and Society: An Outline of Interpretive Sociology*. Berkeley: University of California Press.

Williamson, O. E. 1981. "The Economics of Organization: The Transaction Cost Approach." *American Journal of Sociology* 87:548–77.
Zerubavel, E. 1981. *Hidden Rhythms: Schedules and Calendars in Social Life*. Chicago: University of Chicago Press.

Appendix A

Field Research: Companies Visited, 1999–2000

Company ID	Ownership	Year of Inception	Total Staff	Revenue in Rupees (in Millions)	Revenue in U.S. Dollars (in Millions)
C1	Indian	1997	45	5	0.1
C2	Indian	1990	45	9	0.2
C3	Indian	1996	25	10.4	0.2
C4	Indian	1968	11,495	16,900	368.6
C5	Foreign	1982	784	515.3	11.2
C6	Foreign	1997	492	60	1.3
C7	Foreign	1996	106	30	0.7
C8	Indian	1996	711	717.1	15.6
C9	Foreign	1998	2,366	4,515	98.5
C10	Indian	1995	61	75	1.6
C11	Indian	1994	8	5	0.1
C12	Indian	1995	25	30	0.7
C13	Foreign	1992	593	595.4	13.0
C14	Indian	1996	65	10	0.2
C15	Indian	1995	399	1,385	30.2
C16	Indian	1992	907	1,502.6	32.8
C17	Foreign	1995	30	10	0.2
C18	Indian	1995	10	n/a	n/a
C19	Indian	1997	32	50	1.1
C20	Foreign	1980	140	n/a	0.8