FROM THE BACK OFFICE TO THE FRONT LINES:
THE COMPUTER SOFTWARE DEVELOPMENT LABOR PROCESS
IN A CHANGING BUSINESS ENVIRONMENT

by
Sarah Kuhn

B.A., Harvard University, 1974

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE DEPARTMENT OF URBAN STUDIES AND PLANNING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June, 1987

© Sarah Kuhn 1987

The author hereby grants to M.I.T. permission
to reproduce and to distribute copies of this thesis
document in whole or in part.

Signature of Author

Department of Urban Studies and Planning
May 1, 1987

Certified by
Bennett Harrison
Thesis Supervisor

Accepted by
Langley Keyes, Chair, PhD Committee
Department of Urban Studies and Planning
FROM THE BACK OFFICE TO THE FRONT LINES: 
THE COMPUTER SOFTWARE DEVELOPMENT LABOR PROCESS 
IN A CHANGING BUSINESS ENVIRONMENT 
by 
SARAH KUHN 

Submitted to the Department of Urban Studies and Planning 
on May 1, 1987 in partial fulfillment of the requirements 
for the degree of Doctor of Philosophy. 

ABSTRACT 
This is a study of the work of computer applications 
programmers in a large commercial bank. It attempts a close 
examination of the nature of programming work, using the 
words of programmers and the managers of programming to form 
a description of the work and the recent changes it has 
undergone. With this description as a basis, the study tests 
whether programming work in this setting has been subject to 
fragmentation and routinization, or "industrialization," as 
some have claimed. Finding that it has not, the study 
discusses the forces that have given rise to the present 
organization of programming work, reviews the implications 
of these findings for work in other settings, and points to 
an alternative characterization of programming work. 

The theoretical departure point for the study is Harry 
Braverman's influential book, Labor and Monopoly Capital, 
which argues that capitalist competition in modern society 
results in a "degradation" of the conditions of work for the 
vast majority of workers. Braverman suggests that competi-
tion leads capitalists to assert greater control over the 
"labor process"--that is, over the detailed methods by which 
work is performed--and that this increased control causes a 
serious deterioration in workers' autonomy and in the condi-
tions of employment. In particular, Braverman identifies 
three ways in which work is transformed: (1) Work becomes 
fragmented and a detail division of labor is created. (2) 
The conception of work and its execution are separated. (3) 
Skill requirements become polarized, with a significant 
reduction in skill requirements for the mass of jobs. 
Although Braverman's principal concern is with manufacturing 
and routine clerical work, he briefly discusses the effects 
of this process on computer programming. Philip Kraft and 
Joan Greenbaum offer detailed extensions of Braverman's 
model to programming tasks and workplaces. 

To test this theory, a case study of business applications 
programmers in a commercial bank was conducted. Because 
programming work in financial institutions is said to be the 
most routine, and because a large company offers a greater 
opportunity for the division of labor, a large bank seemed 
to offer the greatest opportunity for the routinization or 
"industrialization" of programming. Open-ended, in-depth 
terviews with about two dozen bank employees were the 
primary data source.
The evidence from the bank did not confirm Braverman's predictions about the reorganization of programming work. (1) The opportunities for fragmentation of work were severely restricted, and even when a programmer worked on a part of a system, such work was integrated with the work of others. A robust internal labor market provided promotion opportunities, and even entry-level employees had jobs with considerable variety. (2) Conceptual work, or "analysis," was not substantially separate from execution, or "coding." Although some employees performed analysis but wrote no code, there were no programmers who were solely coders. Everyone at the bank believed that the systems development process was made more effective when coding and analysis were done by the same person. (3) While the minimum skill requirements for technical proficiency seemed to have dropped for the applications programming generalist, technical skill is only one of the skills required for effective programming. Programmers must also be able to analyze a work process and to communicate with their clients, the system users, to design and maintain successful systems. In recent years, there has been a shift in skill requirements for applications programmers, with a growing emphasis on analysis and communication and a decreasing importance of strictly technical skills, especially coding.

A variety of factors have contributed to the fact that programming at the bank is not organized according to "assembly-line" principles. The fragmented internal market for the programmers' product, the increasingly competitive conditions in a deregulated economy, the need to attract and keep good employees, the state of development of the technology and of the bank's systems, and the constraints of corporate subcultures all contribute to this outcome.

These findings and others from the computer systems area of the bank point the way to a new theory of the nature of work and the determinants of work organization. Programming work may resist industrialization because it is concerned with development—the one-time design of a unique system—rather than with the repetitive reproduction of something already designed. Programmers at the bank, like engineers in an industrial setting, play a key role in both product and process innovation. While they do not set competitive strategy, they maintain the essential infrastructure which supports the bank's existing business, and provide the data needed to manage and to plan for growth. The knowledge of business objectives and of the organizational and business context of their work is essential to their effective performance in this role.

Thesis Supervisor: Bennett Harrison
Title: Professor of Political Economy and Planning
# TABLE OF CONTENTS

Acknowledgments.............................................................................5

Chapter One: INTRODUCTION.........................................................7

The Study.........................................................................................10
Programs and Programming..............................................................11
Findings of the Study.........................................................................13

Chapter Two: PREDICTIONS FOR THE ORGANIZATION OF WORK....16

Braverman’s Model...........................................................................18
The Problem of Realization...............................................................19
Increased Capitalist Control.............................................................20
The Effects of Competitive Pressure and the Attendant Reorganization of Work...23

Destruction of craft skill and the polarization of the skill distribution....23
Fragmentation and the detail division of labor.....................................29
The separation of conception and execution.................................31
Technology.......................................................................................37
The positive potential of technology.................................................38

Braverman-Inspired Studies of Computer Programming...40
Critiques of Braverman.................................................................49
Capitalists are neither omniscient nor omnipotent..........................49
Historical inaccuracy......................................................................51
Shopfloor relations more complex than portrayed............................52
The importance of other contexts....................................................53
Capitalists really seek surplus, not control per se............................55
The new research agenda...............................................................58

Chapter Three: AT THE BANK: PART ONE.................................60

The Case Study...............................................................................60
The Study Method...........................................................................67
Findings of the Case Study..............................................................71
Types of Systems.............................................................................73
"Factory" and "strategic" systems.....................................................74
Evolution and Proliferation............................................................80
Chapter Four: AT THE BANK: PART TWO

The Systems Labor Market

Turnover
Salaries
Conditions of employment
Backgrounds of systems people
Recruitment
Bank's programmer training program
Contract programmers
Professionalism and identification with the larger systems community
Women at the bank
Black workers at the bank

The Rest of the Bank

The systems-business relationship
Accomodations and adjustments
The convergence of the business and systems areas
Who are the users?
PCs and end user computing
Information centers
The work of non-systems people

Chapter Five: COMPARING THEORY AND FINDINGS

Fragmentation and a Detail Division of Labor
No occupational ghetto for coders or maintainers
Assigned to a particular business unit, which limits the extent of the market
Movement from detail work to "whole" work and back, depending on the job
Fragmentation is not equivalent
Chapter Six: CONCLUSION AND SPECULATION

Can the Findings be Extended
to Other Parts of the Bank?.................223
Can the Findings be Extended to Other Banks?...........225
Can the Findings be Extended to Other Systems Workplaces?..........................227

One-person projects..........................228
Can the Findings be Extended to Other Design and Engineering Work?..............229
Can the Findings be Extended to Other Work in General?..........................233
Appendix 1. Some Projects From the Systems Area...........236
Appendix 2. A Textbook Example of a Simple COBOL Program.246
Appendix 3. The Variety of Systems Settings.................247
Appendix 4. Andrew Friedman’s Theory of
             Responsible Autonomy..............253
Figure 1. The Bank After Decentralization.................257
Bibliography.........................................................258
ACKNOWLEDGMENTS

A dissertation, like any good rite of passage, is a community event. It is my pleasure to thank the many people who have helped to give meaning and a sense of engagement to my years in graduate school.

Thanks first to my committee, who have been interested, supportive, and involved during the process of bringing into being this document. Bennett Harrison, my earliest mentor in the department, continued to be a valued role model as I grew. Donald Schön worked closely with me during the final eighteen months of research and writing; his office was the site of many helpful and memorable conversations. Lotte Bailyn first taught me a respect for the methods of social science. I later came to value her also as an astute critic and strong supporter.

I owe a great debt to the men and women of the bank, who astonished me by their generosity with their time and by their enthusiasm for a project of uncertain benefit to them. For whatever accuracy and detail are present in this study, they deserve the credit. Although I would like to thank the twenty-six interviewees by name, I think my gratitude is best shown by allowing them to remain anonymous.

My best thinking is usually done in conversations, and many people over the years have provided stimulating conversation as well as invaluable affirmation. Teresa Amott, Eileen Appelbaum, Barbara Baran, Enno Becker, Barry Bluestone, Louise Dunlap, Roslyn Feldberg, Herb Gintis, Joan Greenbaum, Judith Gregory, Evelyn Fox Keller, Maryellen Kelley, Phil Kraft, Martin Krieger, Wim Overmeer, Paula Rayman, Annalee Saxenian, Jan Schultz, Harley Shaiken, and Lucy Suchman were helpful interlocutors, and some became valued friends as well.

The Harvard-MIT Joint Center for Urban Studies provided me a one-year fellowship at an important time. Bill and Beverly Bouwsma provided a month's working refuge in California. The High Tech Research Group, members of URPE--especially Sam Bowles, Maarty de Kadt, and Luis Prado--and colleagues in the ICON project all contributed to my sense of belonging to a wider circle.
The MIT support staff were supportive in a way that far exceeded the sense intended by the title. Special thanks to Mary Grenham, Meg Gross, and Sandra Wellford; also to Rolf Engler and to the Department of Urban Studies and Planning, which provided the dissertation writer's *sine qua non*: office space.

Any work worth doing is a project of the heart as well as of the head. In addition to the friends and colleagues I have already mentioned, a few special people helped to sustain me and remind me of the larger world outside. My father, Tom Kuhn, has been a role model who set a high standard, and also made it seem worth achieving. My mother, Kay Kuhn, regarded this endeavor as she has all my others: with great generosity and unwavering confidence. My sister Liza and my brother Nat have stood by me, cheering me lovingly on and buying me lunch. Carol Baroudi, who was my first informant; Barbara Lipski, a running mate from pre-graduate school days; and Lindy Biggs, whose path to a doctorate paralleled mine, have been my close and devoted friends. Richard Goldwater has helped me navigate this passage, as well as many others. They have all shared the job of maintaining my animal spirits with my wonderful and most tolerant partner, Ralph LaChance, who met me when I was saying that I would be done with my dissertation in a year. Now, three and a half years later, I am finally done.
CHAPTER ONE
INTRODUCTION

This is a study of the work of computer applications programmers in a large commercial bank. It attempts a close examination of the nature of programming work, using the words of programmers and the managers of programming to form a description of the work and the recent changes it has undergone. With this description as a basis, the study tests whether programming work in this setting has been subject to fragmentation and routinization, or "industrialization," as some have claimed. Finding that it has not, the study discusses the forces that have given rise to the present organization of programming work, reviews the implications of these findings for work in other settings, and points to an alternative characterization of programming work. Two of the biggest forces shaping the transformation of work since the second World War have been the rapid expansion of service occupations and the growing computerization of work. Computer programmers stand at the juncture of these two trends. They are among that vast group of non-manufacturing, non-agricultural employees lumped under the "service" rubric; and as designers of computerized systems they also stand in a unique relationship to the new computer technology. They make up only a small percentage of the total workforce--one-half of one percent of the employed
labor force in 1980—yet their work is sometimes pointed to as a possible model for all postindustrial work.1

In spite of this, programmers have been little studied. Manufacturing work has been the focus of the lion’s share of attention by those who study work methods and job content. There has been increasing interest during the last generation, among both managerial and academic students of work processes, in the study of clerical and other routine office work. The work of technicians, professionals, salespeople, and others in non-routine white collar jobs has been relatively neglected.

This study has two purposes. First, to contribute to the emerging body of studies of the work process and its context in technical, professional, and related fields. A careful examination of computer programming should shed light on the nature of technical and computer-mediated employment. Second, this study seeks to test a specific hypothesis about the the transformation of "the labor process" advanced by Harry Braverman in the mid-1970’s. Braverman suggested that under monopoly capitalism there was occurring an ongoing "degadation" in the quality of jobs held by the vast majority of workers. These jobs, he argued, have become fragmented and subjected to a "detail" division of labor, with conception of a task separated from its

---

1 Employment figures are drawn from the US decennial census of 1980. Computer scientists, systems analysts, and programmers were .53% of the employed labor force over age 16, or 513,863 people.
execution and meaningful skill removed from the majority of work.

Since Braverman, two authors have applied Braverman's model to computer programming. Philip Kraft and Joan Greenbaum argued that the transformation which Braverman projected for many occupations was in fact taking place in data processing workplaces. Drawing on evidence from interviews with programmers and their managers, they argued that, in the words of Kraft, an "industrialization" process was taking place in programming. Today, their work continues prominent among a minute number of studies by social scientists of the content and context of programming work.

The work of Kraft, Greenbaum, and most of all Braverman, defines the starting point for this investigation. All three authors have made a case for seeing programming as an emerging form of routine clerical work. Yet in doing so they have failed to explain anomalous situations: why is all programming work not being routinized? What accounts for the variation across workplaces and across industries in the content and skill level of programming work? The programming workplace described by Kraft and Greenbaum was a generic workplace, a composite of the stories told by interviewees from a range of data processing settings. They gave no sense of why work conditions and job content might vary, or even that they did vary. They also discussed no cases for which the theory did
not fit the organization, leaving little sense of the boundaries and limits of this notion of the "degradation" of work.

The Study

To examine programming work and to test the idea that programming was becoming more routine, quasi-industrial work, this study focuses on applications programmers in the Information Systems area of a large commercial bank. Banks and insurance companies are among the largest employers of programmers, and are reputed to provide programmers with the most routine sort of programming work. (A software development manager at a computer manufacturing company described the conditions of programmers at financial institutions with the term "sweatshop." ) A large commercial bank serving a mass market could be expected to be in the forefront of the industrialization of programming; as a "limiting case" it would be ideal as a site for a test of the extent to which programming has become routinized.

A single-case study approach was chosen because depth of study rather than breadth of coverage seemed important to a careful description of work content and of the complex interrelationship of factors present in the context in which the work was embedded. From the point of view of testing Kraft’s and Greenbaum’s ideas about the transformation of programming, a single case would not prove the correctness of their theory, but it would be sufficient to uncover theoretical problems if they existed.
Furthermore, no case studies of programmers in the context of their organizations have ever been conducted. Thus the scant literature which does exist, while it may describe the perceptions and experiences of individuals, cannot explore the context of their work in all its complexity. The richness of the organizational setting and issues such as the competitive conditions of the industry in which they are employed, the state of the local labor market, and the politics of corporate life are lost in the creation of a generic picture constructed from the perceptions of unrelated individuals.

Programs and Programming

"A computer program," says Frederick Brooks, software engineer and author of a widely-cited book on programming, "is a message from a man to a machine. The rigidly marshaled syntax and the scrupulous definitions all exist to make intention clear to the dumb engine." (19??, p. 164) A program is a written text using letters, numbers, and symbols. This text expresses a set of instructions to be read and executed by a computer. The instructions must be complete and unambiguous2, and must follow exacting rules of syntax, punctuation, and so forth. Although many of the terms which are used to describe written expression in natural language are also used for computer languages ("text", "language", "syntax", "word", "writing") and

2 The criteria of completeness and lack of ambiguity are most eloquently expressed by Joseph Weizenbaum in his excellent and provocative book, Computer Power and Human Reason, 1976.
although many programming languages make heavy use of English words among the mathematical expressions, a computer program has more in common with the holes punched in the paper cards which controlled the Jacquard loom than it does with *Moby Dick* or the poetry of e e cummings.

"Programming" is a generic term used in this study to signify the central activities required to generate a computer program or to construct a computerized system. The tasks of which these activities are composed are named and discussed in Chapter Three. This is fundamentally a study of the programming labor process and its context; it is therefore concerned with programming as an activity, not with the occupation called "computer programming," nor with programmers themselves as individuals.3 Although programmers' words and perceptions are used as a chief source of data, this study is not concerned with the attitudes, work satisfaction, or "consciousness" of programmers except as they apply to understanding the nature

---

3 Maryellen Kelley rightly criticizes Braverman's (and Marx's) use of the term "labor process" for being too vague. She offers a more concrete definition which makes the concept analytically tractable. A single labor process, she says, is characterized by "a common purpose for which actions are undertaken; a specific set of tools, machines, and materials--technical means--that workers use in order to achieve that purpose; and a particular body of knowledge (both formal and informal) upon which workers draw (but to varying extents) in order to determine the appropriate action to be taken, select the correct tool or machine to be used, and execute the specific tasks proficiently." (1984, p. 3)
of programming and its role in the larger organizational and industrial context.4

Findings of the Study

The evidence from the bank reveals a picture of programming as an activity that is substantially at odds with that painted by Braverman, Kraft, and Greenbaum. For applications programmers, who write the programs needed to perform business functions, there has indeed been a simplification of the technical aspects of programming work. Many new and simplified tools are available to expedite the work of programmers, although the increasing complexity of computer systems creates a demand for programmers who specialize in understanding particular aspects of the work. From a strictly technical point of view, the work of applications programmers could be said to be "deskilled." Yet this study reveals that there are many facets of a programmer's work which are not really technical, but which require considerable real skill. Non-technical programming work--such as the analysis of business problems and communication with business "users" of computer systems within the bank--forms a large and growing part of what programmers do at the bank. To call the work of today's

---

4 To attempt to avoid another source of confusion, this is also not a study of "the impact of technology." Although computer technology, and technological change, are an intimate part of the story of applications programming at the bank, to describe the story of the bank as a story of the impact of technology is to suggest that the causal relationship is both simple and unidirectional. It is neither. The bank's adoption of new technologies is both a cause and an effect of the many changes which this research describes.
programmer "deskilled" is to miss the fact that although for some jobs the technical skill requirements have been reduced, there has been a shift at the bank which places increasing emphasis on non-technical skills. As a result, neither the job of programmer nor the individuals who fill those jobs can correctly be seen as less skilled.

While there was a division of labor among programmers at the bank, the major divisions among programmers were along lines of business function, so that each group of programmers was devoted to a particular business application. Within each of these divisions there was also a division of labor based on skill level, personal preference, availability, and other factors. Organizational structure of projects and teams varied from one to the next, so that an individual might have a variety of experiences within the span of a year, a month, or even in the same week. Significant upward mobility kept most programmers from being trapped in entry-level work for an extended period.

Furthermore, the rigid separation of conception and execution suggested by Braverman was absent at the bank. In programming, execution and conception can be hard to distinguish, yet by any measure these were not divided and assigned to different groups of workers. Instead, the various conceptual tasks which are part of the systems development process were distributed—sometimes on the basis of position in the corporate hierarchy of the systems area—
among participants in a particular systems development effort.

A variety of factors contributed to create conditions of work different from those predicted by Braverman, Kraft, and Greenbaum. A change in the type and scale of systems being developed, a history of systems problems caused by misunderstanding and miscommunication between participants in a development project and their corporate clients or "users", a desire on the part of management to attract and hold capable employees, some reduction in the technical difficulty of much applications work, and the special qualities of programming which make it hard to routinize—all these contributed to the difficulty of degrading and "industrializing" programming.

The literature on the labor process, and particularly the work of Braverman, and the contributions of Kraft and Greenbaum, are discussed in detail in the next chapter, Chapter Two. Chapters Three and Four review the findings from field research at the bank, describing the bank and the organization of work from the perspective and using the terminology used by members of the organization. In Chapter Five the evidence from the bank is compared to the predictions of the theories discussed in Chapter Two, this time adopting the language of the theories and applying it to the findings from the bank. Finally, Chapter Six briefly reviews the study's conclusions and considers the issue of the generalizeability of the findings.
CHAPTER TWO

PREDICTIONS FOR THE ORGANIZATION OF WORK

In a landmark work, Labor and Monopoly Capital (1974), Harry Braverman gives a compelling account of the transformation of modern employment that has sparked a revival of interest in the study of the content and organization of work. Drawing his inspiration and many of his analytic tools from Marx, Braverman constructs an account of the changing conditions of work in which he argues that capitalist competition has brought about a progressive "degradation" of work conditions for most workers in the industrialized world. As a result, he argues, the "giant mass of workers...are relatively homogeneous as to lack of developed skill, low pay, and interchangeability of person and function..." (p. 359).

The erosion in the quality of work has reduced occupations formerly characterized by craft-like control of the tools and methods of production to an impoverished state, with "...rationalization and division of labor, simplification of duties, application of mechanization, a downward drift in relative pay, some unemployment, and some unionization." (p. 243) Here is a typical description of these changes, borrowed from the chapter in which he discusses the transformation of clerical work:

"Typists, mail sorters, telephone operators, stock clerks, receptionists, payroll and timekeeping clerks, shipping and receiving clerks are subjected
to routines, more or less mechanized according to current possibilities, that strip them of their former grasp of even a limited amount of office information, divest them of the need or ability to understand and decide, and make of them so many mechanical eyes, fingers, and voices whose functioning is, insofar as possible, predetermined by both rules and machinery. As an important instance of this, we may note the changes in the work of the bank teller, once an important functionary upon whose honesty, judgement, and personality much of the public operation and relations of the bank used to depend. Attached to mechanical and electronic equipment, these employees have been transformed into checkout clerks at a money supermarket counter, their labor power purchased at the lowest rates in the mass labor market, their activities prescribed, checked, and controlled in such a way that they have become so many interchangeable parts."

(p. 340-341)

Through his many descriptions of the transformation of what Braverman and his followers call "the labor process," run three major themes, each capturing an aspect of the transformation. These themes function both as characterizations of the changes which have already taken place in many jobs and occupations, and as predictions of the direction of future change as the process of occupational degradation runs its course. The three themes are:

1) Fragmentation of work and a detail division of labor
2) Separation of conception from execution
3) Destruction of craft skill and polarization of the distribution of skill

For Braverman and for those who draw upon his work, none of these three themes has an independent existence. Though one may be discussed without reference to the others,
all three occur together, and form a totality for analytic purposes.

**Braverman’s Model**

The driving force in Braverman’s model of the transformation of work is capitalist competition. Competition is the prime mover, the essence of capitalism, from which the degradation of the work process for the mass of workers inevitably follows.

"The drive for increased productivity inheres in each capitalist firm by virtue of its purpose as an organization for the expansion of capital; it is moreover enforced upon laggards by the threats of national and international competition. In this setting, the development of technology takes the form of a headlong rush in which social effects are largely disregarded, priorities are set only by the criteria of profitability, and the equitable spread, reasonable assimilation, and selective appropriation of the fruits of science, considered from the social point of view, remain the visions of helpless idealists...Yet no matter how rapidly productivity may grow, no matter how miraculous the contributions of science to this development, no satisfactory level can ever be attained." (pp. 206-207)

In this conviction, Braverman is following closely the work of Marx, who, in Braverman’s words, further "shows how the processes of production are, in capitalist society, incessantly transformed under the principal driving force of that society, the accumulation of capital." (pp. 8-9) The task which Braverman sets himself is to spell out for contemporary readers the ways in which production processes have been and continue to be transformed. The nature of capitalist competition itself receives little attention in
his book; instead, he focuses on the inevitable effects of this competition on the content of work.

The Problem of Realization

The pressures of competition cause the owners of capital escalating concern over the profitability of their operations. As competition advances, increasing levels of profitability and efficiency must be achieved in order to continue to be a contender in the competition for expanded market share. Yet as the capitalist seeks ways to increase profitability, he or she confronts a problem which Braverman describes thus:

"[I]n purchasing labor power that can do much, [the capitalist] is at the same time purchasing an undefined quality and quantity. What he buys is infinite in potential, but in its realization it is limited by the subjective state of the workers, by their previous history, by the general social conditions under which they work as well as the particular conditions of the enterprise, and by the technical setting of their labor." [Emphasis in the original.] (p. 57)

Braverman, again following Marx and other classical economists, makes the distinction between labor, or work

8 In recent decades, the quantitative emphasis of neoclassical economics has drowned out the qualitative and philosophically-oriented approach of the 18th- and 19th-century classical economists. Present-day economists rarely refer to the work of Adam Smith, Ricardo, Locke, and others, now thought of as outmoded ancestors of today's sophisticated and rigorous economic practitioners. Only Marx, of all the classical economists, has a significant following today. One of the consequences of this is that in the minds of students Marx now stands for all classical economists, and is often credited with the development of concepts which he in fact borrowed from his classical predecessors. The distinction between labor and labor power is one example of such a concept. One
actually done, and labor power, the ability to do work. The capitalist purchases labor power rather than labor when hiring a worker at a fixed hourly wage. The amount of work actually performed will vary, as Braverman pointed out above. The challenge to the capitalist who would increase profitability is to increase the amount of actual labor extracted from any given worker at a given wage.

Increased Capitalist Control

The fact that the quality and quantity of work remain undefined even in situations where the rate of pay and hours of work are clearly and definitely set, leads capitalists to use a variety of strategies to increase the quality and quantity of work. In the process, Braverman says, employers shape the process of production in ways designed to support their attempts to realize the maximum potential from the labor they have purchased. No longer is it acceptable to the capitalist simply to supply the machinery and raw materials of production: now the capitalist wants to dictate the techniques and rate of work as well. The factory system gradually replaced the putting-out system because having all producers gathered in a central place under the watchful eye of the capitalist made speed-up possible; capitalists in the steel industry also took over the right to hire

of the sources of the appeal of Marxist thought, though certainly not its only source, is that it is the only surviving representative of a tradition which offers an alternative to the economics profession's heavy quantitative bent, a bent which in the words of Robert Heilbroner gives it "rigor but alas also mortis."
assistants, once the province of skilled craftsmen. Thus the production process itself comes to reflect not just the technical requirements of the work, but also the fact that production is taking place under capitalist conditions.

Of the many effects of capitalist competition on the worker, one of the most central is alienation. Braverman describes it thus:

"Having been forced to sell their labor power to another, the workers also surrender their interest in the labor process, which has now been 'alienated.' The labor process has become the responsibility of the capitalist. In this setting of antagonistic relations of production, the problem of realizing the 'full usefulness' of the labor power he has bought becomes exacerbated by the opposing interests of those for whose purposes the labor process is carried on, and those who, on the other side, carry it on." [Emphasis in the original.] (p. 57)

Thus by breaking up craft control over work and making craftsmen into detail workers under the control of capital, capitalists have brought about the alienation of workers.

9 For essays describing these changes, see Steven Marglin, "What do Bosses Do?" and Kathy Stone, "The Origin of Job Stuctures in the Steel Industry." Both are classics of the literature on the labor process, and both were written about the same time as Labor and Monopoly Capital and share much of Braverman's perspective. The Stone piece is a simplified but compelling account of the reorganization of the steel industry, setting famous events in labor history such as the Homestead strike in the context of capital's attempt to seize control of what had been a craft-controlled labor process.

3 Braverman's uses the term "alienation" in the Marxist sense, indicating the structural fact that the worker loses control of the products of his or her labor. This sense is different from the sense of the term generally used in sociology or psychology, which denotes a mental or social state. In this case as throughout his book, Braverman is not interested in the "conciousness" of workers, but in their objective situation within the economic structure.
from the production process. In addition to the psychological costs which this imposes on individual workers, Braverman argues that there are costs to the production process as well. The problems of converting labor power to labor, which contributed to capitalists' desire to extend their control over the labor process, are made more acute by the very strategy used to address the problem. This then creates a cycle which leads capital to exert further control, causing a deepening sense of disinterest on the part of workers, leading to yet further attempts at control by the owners of capital.

In Braverman's schema, there are two strategies used by capitalists to deal with the problems inherent in realizing the potential embodied in the labor power they employ. The principal strategy, discussed above, involves the transformation of the labor process in ways designed to insulate production, to the extent possible, from workers' reluctance to work. A secondary strategy is the development of institutions and schools of thought whose function is to create the ideological justification for capitalist control of the labor process, and to "habituate" the worker to life under capitalist production. All

4 The exact source of workers' reluctance to work is an interesting subject. Braverman's conviction is that when labor is reluctant, the cause is the alienation and loss of control inflicted upon working people under the capitalist system. Reluctance, then, for Braverman, is not primary or inherent in human nature; instead, it is the reasonable and natural human response to being stripped of control of one's own creative and productive labor.
developments in management thought since Frederick Taylor. Braverman treats as incidental to the main thrust of capitalism, whose essence he believes to be well captured in the writings of Taylor and other proponents of "scientific management." Thus those who have written since Taylor are participating in the second strategy of capital, while top management carries on the task of implementing the first strategy.

The Effects of Competitive Pressure and the Attendant Reorganization of Work

In Braverman's model there are three major effects of capitalist competition on the organization of work. First, the destruction of craft skill and the polarization of the skill distribution; second, fragmentation and the detail division of labor; and third, the separation of conception and execution.

Destruction of craft skill and the polarization of the skill distribution

The form of work organization to which Braverman constantly compares newer, degraded forms of work is the craft form, in which independent producers engage in the manufacture of a complete product, ready for sale on the open market when it leaves the hands of the craftsman.

Here is how Braverman describes craft work:

"From earliest times to the Industrial Revolution the craft or skilled trade was the basic unit, the elementary cell of the labor process. In each craft, the worker was presumed to be the master of a body of traditional knowledge, and methods and
procedures were left to his or her discretion. In each such worker reposed the accumulated knowledge of materials and processes by which production was accomplished in the craft...The worker combined, in mind and body, the concepts and physical dexterities of the specialty..." (p. 109)

As capitalists have acted to reorganize work, the conditions and content of work have been significantly changed.

"The breakup of craft skills and the reconstruction of production as a collective or social process have destroyed the traditional concept of skill and opened up only one way for mastery over labor processes to develop: in and through scientific, technical, and engineering knowledge. But the extreme concentration of this knowledge in the hands of management and its closely associated staff organizations have closed this avenue to the working population. What is left to workers is a reinterpreted and woefully inadequate concept of skill: a specific dexterity, a limited and repetitious operation, 'speed as skill,' etc." (p. 443-444)

Although the craft model Braverman holds up as an example is drawn from the manufacturing and agricultural sectors, he asserts that there are strong parallels between manufacturing and office work, both in the fact that they have an early craft-based labor process, and in the fact that both are similarly transformed and the mass of jobs degraded by the effects of capitalist control.

"Clerical work in its earlier stages has been likened to a craft. The similarities are indeed apparent. Although the tools of the craft consisted only of pen, ink, other desk appurtenances, and writing paper, envelopes, and ledgers, it represented a total occupation, the object of which was to keep current the records of the financial and operating condition of the enterprise, as well as its relations with the external world. Master craftsmen, such as bookkeepers or chief clerks, maintained control over the process in its totality, and apprentices or journeymen craftsmen--ordinary
clerks, copying clerks, office boys—learned their crafts in office apprenticeships, and in the ordinary course of events advanced through the levels by promotion." (p. 298-299)

Indeed, not only are there parallels between manufacturing work and office work, but the difference between the two sorts of work, in lower echelon jobs, have become insignificant compared to the similarities. Both sorts of work, says Braverman, are subjected to the same fragmentation, routinization, and removal of control. Indeed, both are the targets of industrial engineering techniques such as time and motion study. A clerical job no longer contains the promise of high status and upward mobility that it once did; essentially, the lower-level clerical jobs are now simple the equivalent for women of the blue-collar jobs held by working-class men.

Charles Babbage, the 19th century economist and inventor, claimed that the fragmentation of labor allowed scarce skills to be used where they were most needed, but as with the progressive alienation of the worker discussed above, the changes in the work process have in fact set in motion a process that perpetuates and even exacerbates the changes. As Braverman puts it: "...the simplification and rationalization of skills in the end destroy these skills, and, with the skills becoming ever more scarce, the new processes become ever more inevitable--because of the shortage of skilled labor!" (p. 370)

The conviction that the skill level is eroding for the great majority of jobs is a key notion in Braverman's
writing, at least equal in importance to the assertion that conception and execution have become disjoined. It is interesting to note, however, that although he discusses at length the decline in skilled craft work and the reduction of workers to order-takers, Braverman never claims that the skill level of all jobs in the economy, taken on average, has experienced a decline. Instead he offers a convincing refutation of the "upgrading" thesis by carefully examining the basis of the statistical evidence offered in support of the notion that average skill levels have risen. Instead of himself taking a position on what has happened to the average skill level of jobs, he argues that those who inquire about average skill levels ask the wrong question.

"The question is precisely whether the scientific and 'educated' content of labor tends toward averaging, or, on the contrary, toward polarization. If the latter is the case, to then say that the 'average' skill has been raised is to adopt the logic of the statistician who, with one foot in the fire and the other in ice water, will tell you that 'on the average,' he is perfectly comfortable. The mass of workers gain nothing from the fact that the decline in their command over the labor process is more than compensated for by the increasing command on the part of managers and engineers. On the contrary, not only does their skill fall in an absolute sense (in that they lose craft and traditional abilities without gaining new abilities adequate to compensate the loss), but it falls even more in a relative sense. The more science is incorporated into the labor process, the less the worker understands of the process; the more sophisticated an intellectual product the machine becomes, the less control and comprehension of the machine the worker has. In other words, the more the worker needs to know in order to remain a human being at work, the less does he or she know. This is the chasm which the notion of 'average skill' conceals." [Emphasis in the original.] (p. 425)
The thrust of Braverman's argument is that the transformation of the labor process under capitalism has reduced significantly the skill requirements of the majority of jobs while raising the skill level of the minority of highly skilled, influential people who control the production process. It is through this phenomenon that the polarization asserted by Braverman is created.

Braverman admits that there are limits to this tendency, so that exceptions to the general rule of polarization will be found, and the polarization itself will never be absolute.

"This displacement of labor as the subjective element of the process, and its subordination as an objective element in a productive process now conducted by management, is an ideal realized by capital only within definite limits, and unevenly among industries. The principle is itself restrained in its application by the nature of the various specific and determinate processes of production."

He continues: "Moreover, its very application brings into being new crafts and skills and technical specialties which are at first the province of labor rather than management.5 Thus in industry all forms of labor coexist:

5 An illustration of this statement comes from an unexpected source: In 1985, Texas Instruments sponsored a "satellite symposium" on Artificial Intelligence, which was beamed live to campuses and businesses around the country. During the symposium a panel of specialists took questions from the audience. One panelist was asked a question about "knowledge engineering," the new occupation devoted to interviewing experts and codifying their knowledge in order to build so-called "expert systems." In his
the craft, the hand or machine detail worker, the automatic machine or flow process." (p. 172)

This last statement, while extremely plausible, is puzzling coming from Braverman, since he has spent a great deal of effort trying to convince us that the new work generated by the attempt to remove skill from the labor force at large is in fact the province of management, not of labor, and that this is the source of the polarization. The discrepancy between the two positions is not crippling, since Braverman has acknowledged the unevennesses in the process; still, it begs for clarification.

What is the place and role of the middle class in this polarity? Braverman disputes the notion, propounded by observers who point to the growth of clerical occupations, that this growth signals an increase in the size of the middle class. On balance these clerical occupations are proletarian occupations, he states, and simply represent the development of a new form of proletariat.

Still, he does not dispute the existence of a significant middle class.

"Like the petty bourgeoisie of pre-monopoly capitalism (the petty proprietors in farming, trade, services, the professions, and artisan occupations), it [that is, the middle class] does not fit easily into the polar conception of economy and society. But unlike that earlier middle-class mass, which has so largely evaporated, it corresponds increasingly to the formal definition of a working class. That is, like the working class it possesses no economic
or occupational independence, it employed by capital
and its offshoots, possesses no accesss to the labor
process or the means of production outside that
employment, and must renew its labors for capital
incessantly in order to subsist." (p. 403)

Also unlike the members of the old petty bougeoisie,
the present middle class plays a role as a buffer between
capital, embodied in owners and top management, and the mass
of workers who make up the modern proletariat. Yet, like
the proletariat, the managers and professionals of the
middle class are themselves subject to the same pressures of
capitalist competition which have acted to degrade the work
of those below them in the organizational hierarchy.

Fragmentation and the detail division of labor

The first major effect of capitalist competition on
work is the introduction of what Braverman calls "the detail
division of labor." He is careful to distinguish this
fragmentation of whole occupations, which he argues is found
only in capitalism, from the broader and universally
occurring division of labor that leads to craft
specialization.

"The division of labor in society is characteristic
of all known societies; the division of labor in
the workshop is the special product of capitalist
society. The social division of labor divides
society among occupations, each adequate to a branch
of production; the detailed division of labor
destroys occupations considered in this sense, and
renders the worker inadequate to carry through any
complete production process. In capitalism, the
social division of labor is enforced chaotically and
anarchically by the market, while the workshop
division of labor is imposed by planning and
control...While the social division of labor
subdivides society, the detailed division of labor
subdivides humans, and while the subdivision of
society may enhance the individual and the species,
the subdivision of the individual, when carried on
without regard to human capabilities and needs, is a
crime against the person and against humanity." (p.
72-73)

Braverman, then, is distinguishing between an economy
in which individuals specialize in particular products which
they then offer for sale--thus participating through the
market in a society-wide division of labor--and an economic
system in which workers produce only small parts of
products--parts which could not stand on their own as the
basis for exchange with other producers. These two sorts of
economic systems, he argues, have very different
implications for the quality of life on the job and the
welfare and standard of living of the working population.

The distinction between these two systems lies in the
fact that in the first, a complete job is done by and on
behalf of the producer him or herself; in the second, the
work is performed by the worker as part of a production
process organized by another to serve the interests of that
other. The distinction between the two systems is not based
on whether the production process is subjected to an
analysis, and even a division of its constituent parts.
Indeed, Braverman says, "...methods of analysis of the labor
process and its division into constituent elements have
always been and are to this day common in all trades and
crafts, and represent the first form of the subdivision of
labor in detail." (p. 76) The difference between the
analysis of work conducted by workers on their own behalf
and that conducted by capitalists is that:
"The worker may break the process down, but he never voluntarily converts himself into a lifelong detail worker. This is the contribution of the capitalist, who sees no reason why, if so much is to be gained from the first step--analysis--and something more to be gained from the second--breakdown among workers--he should not take the second step as well as the first." (p. 78)

Not only is the production process under capitalism fragmented; capitalism also segments the working population into task-based ghettos which limit wages and interrupt skill development. This segmentation Braverman calls "the Babbage principle" after Babbage, who pointed out that the division of labor has an advantage not noted by his predecessors in the field of economics (most notably Adam Smith.) The detail division of labor, observes Babbage, allows the employer to hire for each task the cheapest and least skilled person capable of performing that particular task. If labor were not so divided, workers would have to have the strength to do the most strenuous task and the skill to do the most difficult--and this would require that the most expensive sort of laborer be hired for an undivided production process. (p. 79-80) Capitalists further justify the detailed division of labor, says Braverman, not only because it increases their control and reduces their costs, but also by claiming that it creates a social benefit by husbanding scarce societal resources--in the form of skilled labor--and saving these resources for the applications in which they are truly needed.

The separation of conception and execution
The characteristic of human work which permits the fragmentation of labor is the fact that the planning of work can be separated from the performance of the task.

"[I]n humans, as distinguished from other animals, the unity between the motive force of labor and the labor itself is not inviolable. The unity of conception and execution may be dissolved. The conception must still precede and govern execution, but the idea as conceived by one may be executed by another. The driving force of labor remains human consciousness, but the unity between the two may be broken in the individual and reasserted in the group, the workshop, the community, the society as a whole." [Emphasis in the original.] (p. 50-51)

Capitalists take advantage of the fact that conception and execution are separable in their attempts to reorganize production.

"The separation of hand and brain is the most decisive single step in the division of labor taken by the capitalist mode of production. It is inherent in that mode of production from its beginnings, and it develops, under capitalist management, throughout the history of capitalism, but it is only during the past century that the scale of production, the resources made available to the modern corporation by the rapid accumulation of capital, and the conceptual apparatus and trained personnel have become available to institutionalize this separation in a systematic and formal fashion." (p. 126)

"A necessary consequence of the separation of conception and execution is that the labor process is now divided between separated sites and separate bodies of workers. In one location, the physical processes of production are executed. In another are concentrated the design, planning, calculation, and record-keeping. The preconception of the process before it is set in motion, the visualization of each worker’s activities before they have actually begun, the definition of each function along with the manner of its performance and the time it will consume, the control and checking of the ongoing process once it is underway, and the assessment of results upon completion of each stage of the process—all of these aspects of production have been removed from the shop floor
to the management office. The physical processes of production are now carried out more or less blindly, not only by the workers who perform them, but often by lower ranks of supervisory employees as well. The production units operate like a hand, watched, corrected, and controlled by a distant brain." (p. 124-125)

The separation of conception and execution is intimately linked to the fragmentation of work. For Braverman, indeed, the two are effectively inseparable. Neither could exist without the other, and both are implemented as a unit in the drive of capitalists to assert control over the labor process and thereby increase profitability. Yet the two are not fully synonymous, and it will in later chapters be helpful analytically to have distinguished them.

The salient feature of fragmented work is the narrowness of the task to which the worker is assigned. The production worker becomes a detail worker whose product is not really a product at all but just an aspect of a larger production process. The creation of detail workers necessarily implies that conception and execution be separated to some extent, since some measure of conception must precede the creation of fragmented jobs, and must function in the course of production to coordinate the work of the detail workers. Yet in principle it is possible to imagine a production system in which the workers who perform the fragmented tasks also participate in the initial planning and overseeing of the production process. In Braverman’s schema such an organization of work would be an
impossibility, since he believes that the detail division of labor comes about only because capitalists have managed to assert control over the labor process and have reserved to themselves the job of overseeing and dictating the work of detail workers.

The separation of conception and execution which removes decisionmaking from the shop floor has also swollen the ranks of office workers. As Braverman puts it, "...the separation of conceptualization from execution--the removal of all possible work from the shop floor, the point of execution, to the office--and the further necessity of maintaining a shadow replica of the entire process of production in paper form, brings into being large technical and office staffs." (p. 239)

The effective separation of conception and execution was the central mission of the scientific management movement and its best-known advocate, Frederick W. Taylor. Braverman quotes at length from Taylor’s Principles of Scientific Management. He describes Taylor’s system as consisting essentially of three propositions: first, that managers should gather the traditional knowledge of the workmen and reduce it to "rules, laws, and formulae; second, that "all possible brain work should be removed from the shop and centered in the planning or laying-out department..."; and third, that all workers should be given a detailed description of what they are to do and how they
are to do it, including the amount of time allowed to them. (Braverman, 1974, pp. 112-118)

Taylor's work contains an interesting paradox. On the one hand, he maintains that the workers most capable of competent execution of work are not best able to plan and to comprehend the demands of the work. As proof of this fact he cites the slowness and inefficiency with which he says most workers perform their assigned duties. On the other hand, Taylor acknowledges that workmen have genuine incentive to restrict output. This incentive arises from the fact that their employers are in the habit of reducing the piece rate when the speed of production increases. Taylor himself complained bitterly of the tendency of management to cut his "scientific piece rate" schedule.

On the first point, Taylor "asserts as a general principle... that in almost all of the mechanic arts the science which underlies each act of each workman is so great and amounts to so much that the workman who is best suited to actually doing the work is incapable of fully understanding the science..." (Principles of Scientific Management, 1911/1947, pp. 25-26) Yet in the illustrations Taylor gives, drawn primarily from heavy manual labor such as carrying pig iron or shoveling ore, it is unclear that the constraint has anything to do with the inability of the worker to comprehend the best methods for executing the task. Instead, the increases in production displayed in Taylor's illustrations appear to come largely from increased
work effort, not from an increased efficiency born of Taylor's superior understanding of the work process.

Taylor asserted that his system was intended to create a "mental revolution" which would focus the attention of both management and the workforce on the business of expanding production. He intended that "both sides take their eyes off of the division of the surplus as the all-important matter and together turn their attention toward increasing the size of the surplus until this surplus becomes so large that it is unnecessary to quarrel over how it shall be divided." (Testimony, 1912, pp. 29-30) To achieve this, he proposed a piecerate system whose effectiveness depended on its stability as output increased. Only such stability would guarantee that in fact the gains of expanding production would in fact be shared by both sides. Taylor was taken aback at management's tendency to cut his piecerates, yet this tendency had a long-established history, and was the source of the restriction of output by workers of which he had complained.

Taylor's method for reorganizing and analyzing work resulted in the creation of two sites—one the site of conception, the office, and the other the site of execution, the shop. From this fact and from the fact that office staffs grew as capitalists attempted to reduce direct labor to a minimum, it would be easy to conclude that the number of influential and intellectually demanding jobs was growing as production jobs became fewer and more routinized.
Braverman is quick, however, to point out that the same pressures which lead capital to take control of the shop floor are present in the office, too, especially as office employment becomes a larger portion of total employment. "The functions of thought and planning became concentrated in an ever smaller group within the office, and for the mass of those employed there, the office became just as much a site of manual labor as the factory floor." (Braverman, 1974, p. 316) He elaborates:

"The progressive elimination of thought from the work of the office worker thus takes the form, at first, of reducing mental labor to a repetitious performance of the same small set of functions. The work is still performed in the brain, but the brain is used as the equivalent of the hand of the detail worker in production, grasping and releasing a single piece of 'data' over and over again. The next step is the elimination of the thought process completely—or at least insofar as it is ever removed from human labor—and the increase of clerical categories in which nothing but manual labor is performed." (p. 319)

Thus the growth of the office will not, as some observers have suggested, rescue the workforce from narrow, dull, alienated work.

Technology

"...[I]n the capitalist mode of production, new methods and new machinery are incorporated within a management effort to dissolve the labor process as a process conducted by the worker and reconstitute it as a process conducted by management." (p. 170) For Braverman, technological development must be understood within the context of the
overall development of capitalism. The pressures of capitalist competition cause changes in the labor process, and new technologies are developed and applied to support and to reify changes already made. Like Adam Smith and Karl Marx (whom Braverman defends against charges that he is a technological determinist), Braverman believes that changes in the organization of work are prior to the mechanization of the work process. Once work is reorganized, new technologies are then appropriated and applied to the new methods of production.

It follows for Braverman that there is no autonomous development of technology which by itself stimulates changes in the production process. New technologies are adopted only if they fit into capitalists' plans for how production should be organized. Capitalists' choices about how to organize production stimulate technological development along certain lines and not along others. Thus the path of technological development is determined by the needs of capitalism.

Furthermore, capitalists actively use machinery as a means of control of the labor process, not simply as an aid to increase productivity. The application of new technology is a technique for expanding control over production and for "divesting the mass of workers of their control over their own labor." (p. 193)

The positive potential of technology
In spite of his conviction that the direction of technological development is determined by the needs of capitalists, it is in his discussion of technology that Braverman shows the greatest optimism. While most of his book is devoted to his critique of the status quo and offers little in the way of recommendations for alternative paths (save perhaps, implicitly, for the overthrow of capitalism), Braverman offers a brief glimpse as his alternative vision of how production might develop in the future.

"In reality, machinery embraces a host of possibilities, many of which are systematically thwarted, rather than developed, by capital. An automatic system of machinery opens up the possibility of the true control over a highly productive factory by a relatively small corps of workers, providing these workers attain the level of mastery over the machinery offered by engineering knowledge, and providing they then share out among themselves the routines of the operation, from the most technically advanced to the most routine. This tendency to socialize labor, and to make of it an engineering enterprise on a high level of technical accomplishment, is, considered abstractly, a far more striking characteristic of machinery in its fully developed state than any other. Yet this promise, which has been repeatedly held out with every technical advance since the industrial revolution, is frustrated by the capitalist effort to reconstitute and even deepen the division of labor in all of its worst aspects, despite the fact that this division of labor becomes more archaic with every passing day." (p. 230)

Braverman continues this theme when he writes about the transformation of clerical jobs:

"...as in manufacturing, the office computer does not become, in the capitalist mode of production, the giant step that it could be toward the dismantling and scaling down of the technical division of labor. Instead, capitalism goes against the grain of the technological trend and stubbornly
reproduces the outmoded division of labor in a new and more pernicious form." (p. 328-329)1

Based on a reading of Braverman's book, one would gather that he is nostalgic for the days when skilled craft workers controlled their own individual labor processes. This is our only glimpse at an alternative vision, that of a socialized production process in which workers, rather than reverting to techniques of a previous era, collectively master the scientific, design, and operational knowledge now the province of management and apply it to their own ends.

Braverman-inspired studies of computer programming

Two authors have undertaken to study computer programming as a labor process, and both have reported finding the very changes predicted by Braverman in Labor and Monopoly Capital. Philip Kraft, a sociologist, and Joan Greenbaum, an economist, interviewed programmers and managers in a variety of work settings and concluded that the process described by Braverman--fragmentation and routinization of work, declining skill requirements for most workers, separation of conception and execution, and growing

---

1 These passages are puzzling in view of Braverman's insistence that capitalist's needs determine the direction of technological development. If capitalists did in fact have such control, how could a technology with a "tendency to socialize labor" have developed? And further, isn't the assertion that "capitalism goes against the grain of the technological trend" an assertion that the characteristics of technology and the needs of capital are not always in harmony?
alienation—was indeed taking place in programming workplaces.

Kraft and Greenbaum both describe the transformation of an occupation which, according to Braverman, "displayed the characteristics of a craft" in the 1940s and early 1950s (1974, p. 329). As time went on, managers began to assert control of the labor process, introducing hierarchy, bureaucratic rules, and a division of labor. According to Kraft, in his book *Programmers and Managers*,

What is most remarkable about the work programmers do is how quickly it has been transformed. Barely a generation after its inception, programming is no longer the complex work of creative and perhaps even eccentric people. Instead, divided and routinized, it has become mass-production work parcelled out to interchangeable detail workers. Some software specialists still engage in intellectually demanding and rewarding work—people who are called by such names as systems engineers, analysts, or simply software scientists—but they make up a relatively small and diminishing proportion of the total programming workforce. The great and growing mass of people called programmers (as well as those who do software work but for a variety of reasons are called something else) do work which is less and less distinguishable from that of clerks or, for that matter, assembly line workers. (Kraft, 1977, p. 97)

Similarly, Greenbaum reports in her study, *In the Name of Efficiency*,

The work process was divided and specialized in order to isolate individual tasks that could be given to separate workers. Conceptual tasks such as systems analysis were separated from more routine chores such as programming, and the most repetitive functions were assigned to operations positions. Furthermore, the tools used by each group were simplified so that each step in the work process could become more repetitive. Operating systems, for example, were developed to simplify the work process of both operators and programmers. As these systems became more sophisticated, they assumed
worker functions, incorporating directly into the computer hardware and software decision-making that formerly had been done by people. (1979, p. 160)

Both Kraft and Greenbaum offer brief histories of programming as an occupation, pointing to key events in the evolution of managerial control of the programming labor process. For Kraft, the SAGE program during the Korean war was a turning point: in an attempt to double the number of programmers in a short period of time, the military instituted the first systematic division of labor in the computer field, creating a hierarchical distinction between computer programmers and systems analysts. Greenbaum, a practicing programmer at the time, recalls the organizational changes attendant on the introduction in the middle 1960s of the IBM System 360: the rigid segregation of computer operators from programmers.

It was during this decade also that management problems in business data processing installations became increasingly acute. Labor was a rising percentage of total cost of data processing, and "most computer projects could be characterized as overbudget, late, and ineffective." (Greenbaum, 1979, pp. 57-58) Clashes between programmers and their employers were rampant. Programmers had considerable power over the terms and conditions of their work. Greenbaum quotes one programmer: "In those days we really had control. Management never understood what we were doing and we really didn’t care. It was fun and what we were doing made us feel important..." (1979, p. 65)
It was in this context, according to Kraft and Greenbaum, that employers acted to take control of the labor process. They made changes such as: "the attempt to divide 'hand' from 'head' work; efforts to progressively reduce the tasks of programming into their smallest possible parts; and the parcelling out of these fragments to different people (coders, programmers, analysts, managers) who occupy different 'rungs' in the organizational hierarchy." (Kraft, 1977, p. 67) These steps were necessary, argues Kraft, because the ability to make a profit is "inseparable from the ability of the owner (or his hired manager) to control the manner in which a product is made." (1977, p. 98) Control of the work process, agrees Greenbaum, is a prerequisite for cost reduction.2

Managers use the techniques mentioned above in the search for ways to control the labor process. The fragmentation of work, the separation of conception and execution, and the detail division of labor are all tools that extend managerial control. Technology plays a role, too, in that both Kraft and Greenbaum attribute changes over time in hardware and software to the desire of managers to control the production process and to reduce their

2 In fact, Greenbaum seems ambiguous on this point. On the one hand, she says that cost reduction requires managerial control of the labor process. On the other, she asserts that workers work most effectively when allowed to organize their own work process. Thus the history of the development of programming is a history of reorganization conducted by management "in the name of efficiency" which may in fact sacrifice productive effectiveness in order to achieve hierarchical control.
dependence on costly skilled labor. Thus the changes in the labor process are not the result of technological imperatives; indeed, the reverse is true: the development of the technology itself is shaped by managerial concerns about control.

Both Kraft and Greenbaum stress the importance of a popular software development technique known as structured programming. Structured programming imposes a set of rules for writing code which they argue fragment the work process and bring it under effective managerial scrutiny and control. This technique, says Greenbaum, "completes the process of control that was begun by division of labor and establishment of standards." (1979, p. 81) Kraft describes the changes made in the software development process:

A hierarchy of authority could now be established [using structured programming] by arranging various job fragments in a rank-order on the basis of either skill or an understanding of the task as a whole or both. Managers defined edp problems and gave them to high-level analysts or similar specialists to design the software system. Each component was given to a separate project group (which might have only a single programmer/coder if the task was simple enough) which worked independently of the others. In complex systems, each module could be so narrowly defined that they were essentially coding exercises, making no sense to the people who did the work. The completed tasks or modules were then sent back to the analysts to be combined and tested as a system. The assembled final product could then be turned over to the client for use.

Structured programming, in short, has become the software manager’s answer to the assembly line, minus the conveyor belt but with all the other essential features of a mass-production workplace: a standardized product made in a standardized way by people who do the same limited tasks over and over without knowing how they fit into a larger undertaking. (1977, p. 59)
Like Braverman before them, both authors acknowledge that the managerial reorganization of work takes place unevenly. Within workplaces, there will be variation in the degree of rationalization depending on the type of work being done. Thus Greenbaum:

Most companies have three types of ongoing projects: maintenance of existing programs, modification of existing systems, and development of new ones. By far, the majority of all work falls into the first two categories, which lend themselves to more routine activities and therefore more standard procedures. (Greenbaum, 1979, p. 117)

Between firms there will also be variations in the organization of work. This is especially true of programming as an occupation, says Kraft. "In an occupation so amorphous, whose suboccupations are so vaguely defined and which inevitably overlap, each workplace will define its workers' tasks somewhat differently." (1977, p. 74)

Finally, there are a few passing references to the problems associated with the regularization of this sort of work. As Kraft says (1977, p. 78), "[r]elations between mind-worker and detail worker are not as rigid as in other occupations, however, because the work itself has not been easy to fragment..." While Kraft and Greenbaum acknowledge these variations, the basic trend is very clear to them, as it was to Braverman: a trend toward the fragmentation and rationalization of programming work.

Although for the most part Kraft and Greenbaum are in accord with Braverman's analysis of the transformation of work, there are a few points of difference worth mentioning.
First, both attend to the subject of worker's consciousness, a subject Braverman deliberately chooses to exclude from his analysis. Kraft describes how the socialization of programmers helps to legitimate the managerially-imposed changes in the labor process.

...the training of software workers is organized to impart considerably more than specific or general skills. It is also organized to encourage programmers to accept the segmented nature of their work and the hierarchical structure of the organizations which employ them. They are encouraged to identify corporate profitability with technical rationality, technical rationality with efficiency, and their own individual success and advancement with the success of their future employers...They come to understand that advancement requires they identify with the goals and expectations of the organization, i.e., of the owners and managers. (1977, p. 101-102)

Greenbaum discusses the attitudes and behavior of programmers, describing these in a way that seems designed to demonstrate the similarities between programmers and blue collar workers. In this way she lays the groundwork for the second departure from Braverman's framework. Braverman's critics frequently note that he does not discuss worker resistance to managerial initiatives, and Greenbaum explicitly includes a discussion of worker resistance and compliance in her account of the culture of the shopfloor.

Third, when Kraft and Greenbaum discuss the separation of conception and execution (or head and hand, as Kraft calls it), they generally treat analysis as the conceptual work and programming as execution. Each of them departs from this equivalence, however, on at least one occasion. Kraft describes three basic software occupations, ranked in
order of increasing conceptual content: coding, programming, and analysis. Presumably programming is here regarded as an intermediate form, having some of the characteristics of each extreme.

Greenbaum also drops an interesting hint that there might be three tiers, rather than just the two posited by Braverman, developing in programming as it undergoes reorganization. In a section already quoted above, she says, "[c]onceptual tasks such as systems analysis were separated from more routine chores such as programming, and the most repetitive functions were assigned to operations positions." Thus she too seems to suggest that programmers constitute an intermediate form in the development of a detail division of labor. Because neither Kraft nor Greenbaum elaborates, we do not know whether they view programming as a stable semi-rationalized form of work, or whether they regard it as the temporary manifestation of a job on its way to full rationalization.

If the former were true, this would be a dramatic departure from Braverman, for whom it is crucially important that the polarization of skill creates two classes, not three. Although Braverman discusses briefly what he calls "the middle layers of employment"—professionals and middle managers—which he believes are undergoing the same process of degradation and polarization to which the workforce as a whole is subject, Braverman's model is premised on the opposition of two classes and the polarization of skill and
power which reinforces and replenishes it. Any process which generated three categories rather than two would pose a serious challenge to Braverman's view.

Finally, Greenbaum also departs from Braverman--but returns to a Marxist notion--when she suggests that managerial strategies designed to control the workforce contain problems, or "contradictions," that may ultimately destroy these strategies and force the development of new ones. The division of labor, says Greenbaum, has reduced the number of software workers who can do "a bit of everything," and Greenbaum quotes a manager who complained that he needed such people at a time when the industry was growing fast. Furthermore, this division could cause production problems, as when a computer operator was unable to handle a nonroutine production problem because he or she did not understand the computer's operating system. And finally, she asserts that managerially-imposed hierarchy "pose[s] contradictions that are continually pushing carefully balanced organization schemes into new formations and unraveling the objectivity and "rationality" of the formal structure." (1979, p. 111) The implications of these contradictions are undeveloped by Greenbaum, but the existence of the contradictions themselves is asserted.
Critiques of Braverman

Since the publication in 1974 of *Labor and Monopoly Capital*, scores of scholars and practitioners have focused their attention on the labor process, hoping to test the accuracy and resilience of Braverman's theories about the degradation of work. Most who have written about the labor process in the ensuing years have been generally sympathetic to Braverman's point of view—for example, his conviction that labor and management have opposing interests, and that the capitalist system creates a set of incentives which are antithetical to worker's self-realization on the job—but they have also offered criticisms and modifications to Braverman's model. This section reviews the critiques of Braverman which have circulated in the thirteen years since the book's publication.

Capitalists are neither omniscient nor omnipotent

Many commentators have noted that Braverman writes about management-initiated changes in the labor process as if managers were always successful at implementing their ideas, and as if they always avoided unforeseen consequences. Perhaps the single most popular criticism of Braverman's work is that he neglects the role that worker resistance to managerial initiative play in shaping the labor process. Observations to this effect can be found in almost every commentary on Braverman (see, for example, Hill, 1981; Wood, 1982; Burawoy, 1979; Friedman, 1977; Edwards, 1979;
Littler, 1982; Thompson, 1987) Typical is the remark of Wood and Kelly, "In their transition from conception to implementation, managerial schemes are invariably modified (to differing degrees) by workers' own counterinitiatives, and it would be necessary to appreciate such modifications in any analysis of managerial practices and their consequences" (1982, p. 85).

On a related note, Elger (1982) and others have suggested that Braverman relies too heavily on managerial theorists to make his case about changes in the work process. It does not follow that simply because an idea is expressed in the management literature, it must therefore have been successfully adopted in practice. In particular, Crompton and Reid (1982) suggest that Braverman has inaccurately assumed an unproblematic adoption of Taylorism by the bulk of firms. Wood and Kelly assert that, particularly where there is significant and unpredictable variation either in raw materials or in the production process, the use of Taylorist techniques creates "problems" for capital (1982, p. 87).

Other reasons to doubt that management has unfailing foresight and complete control include the fact that managerial initiatives have unintended consequences that may cause problems and undermine the very managerial control they were intended to enhance (Crompton and Reid, 1982; Hill, 1981). Furthermore, as Wood and Kelly point out, managerial objectives change over time, and different
segments of management may have different interests and therefore differing strategies at any given point in time (1982).

Historical inaccuracy

Braverman counterposes to the current degraded condition of the mass of workers the ideal of the independent craftsman of an earlier era. Much of Braverman's account hangs on the idea that through techniques such as scientific management, employers have taken control of the production process, with serious ill effects for the workforce at large. Elger (1982) says of this that "[i]t tends to portray the craftsmen of the second half of the nineteenth century in terms of the 'artisan ideal', when clearly their positions must be seen as in various ways transitional, marked by a substantial degree by real subordination to capital." Littler (1982) concurrs, arguing that direct entrepreneurial control was present only in a minority of nineteenth-century firms, and that subcontracting and putting-out, which involved more arm's-length relationships between labor and capital, were also prevalent. These challenges to Braverman are persuasive in establishing the importance of conducting historical research on these questions, but do not in themselves disprove the overall thrust of Braverman's argument. It is unclear whether the main debate is simply over the question of the timing and the evenness of the changes Braverman
describes, or whether there is something more fundamental at stake.

A final issue of historical accuracy is raised by Beechey (1982), who suggests that Braverman romanticizes the preindustrial family in the same way that he romanticizes the preindustrial craftsman. She suggests that an analysis of the family and its connection both to the development of the labor process and to the development of capitalism as a whole is a needed addition to Braverman’s ideas.

Shopfloor relations more complex than portrayed

Several writers on the labor process contend that conditions on the shopfloor require a more complex understanding than that offered by Braverman. Burawoy (1979) concluded after close study of a midwestern machine shop that workers and first-line managers collaborate in forming a shopfloor culture which secures the participation of workers under what might otherwise be considered unacceptable working conditions. Braverman, as mentioned above, declines to examine workers’ attitudes and consciousness, but for Burawoy these are the aspects of the labor process most in need of explanation.

Another issue on which Braverman places considerable emphasis is that of skill, and the degradation of skill which confronts the majority of workers as management takes control of work. More (1982) suggests that skill can be thought of as real and necessary for production, or as a socially constructed phenomenon which serves to protect the
interests of those groups who have succeeded in defining themselves as skilled. Lee takes the position that craft skills are real, and that the apprenticeship system has survived because it brings financial returns to employers (1982).

Jones (1982) asserts, based on a study of the impact of numerical control on machining work, that characteristics of the product and of the production process serve as constraints on management’s ability to reallocate and reduce worker’s skills.

Finally, Wood and Kelly (1982) criticize Braverman for holding a zero-sum notion of control and its bases. An increase in control on the part of employers does not necessarily require an identical decrease in control on the part of workers. Production knowledge, in particular, is not zero-sum, and management can increase its knowledge of the production process without necessarily depleting the knowledge of workers.

The importance of other contexts

For Braverman, the wider context in which the reorganization of work occurs is the economic system—capitalism. Other contextual factors are unimportant beside this powerful formative influence. Critics of Braverman, while still believing that the nature of capitalism profoundly shapes work relations and thus the labor process, also believe that to thoroughly explain the variations in the organization of work, one must seek other contributing
causes. Three contextual factors are most often named as lacking in Braverman's analysis. Cyclical and sectoral interindustry shifts, product and labor markets, and broader political circumstances are all important elements in the shaping of the labor process.

Littler (1982) points out that the "deskilling" of the mass of workers can be achieved either through direct confrontation between management and labor, or by the more impersonal avenue of interindustry shifts, which may favor certain skills over others, or favor industries using less skilled labor over those using more. Lee (1982) believes that the reductions that have occurred in the availability of craft jobs are attributable more to cyclical changes in industry structure than to any internal reorganization of work.

Lee also calls attention to the importance of labor markets, which help to determine the cost and availability of labor, in shaping how workers are utilized by employing firms. Because cost reduction is so important a driving force in Braverman's analysis, Lee finds the omission of the role of labor markets from a discussion of the reorganization of work to be striking (1982). Finally, Elger (1982) launches a similar criticism of Braverman's neglect of political factors, asserting that Braverman is wrong to believe that capitalists have achieved domination over labor solely through struggle at the level of the labor process.
Capitalists really seek surplus, not control per se

As Hill (1981) notes, capitalists' primary goal—in Marxist terms—is to accumulate capital through the extraction of surplus value from workers. The detailed control of the labor process is only one technique among many for enhancing accumulation. Braverman may be faulted for concentrating too much of his attention on control, neglecting other important factors.

Even more problematic for Braverman, however, is a criticism offered by Friedman, Edwards, and others: that employers may sometimes find it more profitable to lessen their direct control of the labor process. Friedman suggests that employers adopt different strategies with different segments of the workforce. He calls the two dominant strategies observed in his English case studies (1977) "Direct Control" and "Responsible Autonomy." Direct Control uses "coercive threats, close supervision, and minimising individual worker responsibility"—in short, it is a Taylorist strategy. Responsible Autonomy "attempts to harness the adaptability of labour power by giving workers leeway and encouraging them to adapt to changing situations in a manner beneficial to the firm." (p. 78) The latter strategy tends to be used with the more privileged workers—craft workers professional workers, and so forth. According to Friedman, the use of a Responsible Autonomy strategy with skilled workers is more productive for the firm, because skilled workers have the power to resist management's
attempts at Direct Control, so that a Taylorist approach would backfire with this group.

Still, Friedman is mindful that both the Direct Control and the Responsible Autonomy strategies contain contradictions. Both strategies are impossible to fully realize, and thus encounter difficulties as they are applied. In the case of Direct Control, which treats workers as if they were machines, problems arise because workers are not in fact machines. Direct Control forgoes the positive aspects of human labor while at the same time eliciting resistance from workers. Responsible Autonomy, on the other hand, "does not remove alienation and exploitation, it simply softens their operation...workers behave as though they were participating in a process that reflected their own needs, abilities, and wills..." (1977, p. 101). This element of hypocrisy limits the effectiveness of Responsible Autonomy as a strategy, preventing it from solving permanently the capitalists' problems of extracting surplus.

Edwards (1979) attempts an historically-based account of the evolution of three forms of control--simple control (exerted in small, entrepreneurial organizations, technical control (found in its ultimate expression on the assembly line), and bureaucratic control (characteristic of large firms). These three forms coexist, says Edwards, in the modern economy, but historically each grew out of a contradiction created by its predecessor. In this self-
consciously dialectical account, technical control developed to resolve a crisis in simple control created by such factors as the rapid expansion of corporations and the rebellion fomented by obtrusive and capricious supervision. Bureaucratic control in turn developed to resolve some of the problems of technical control: it reestablished the legitimacy of hierarchy in the corporation, offered a set of impersonal rules as a way of reducing conflict, and affected even those employees whose jobs were not subject to technical control.

Susan Porter Benson's work on department store saleswomen (1981) also supports the contention that management may adopt a strategy other than that of Taylorist direct control when trying to increase profitability. She describes the efforts of department store managers to institute "skilled selling" in the early decades of the twentieth century. Because it was the saleswoman who had direct contact with the customer, she was instrumental in making or losing a sale. Instead of making saleswomen into automatons, unable to react to the uniqueness of each selling opportunity, management tried to teach working-class women selling skills. "The new skilled sellers were to be thinking but also obedient employees; they were to follow store procedures to the last rigorous detail and yet respond creatively to opportunities to sell as they arose; they were to develop independent judgement but yet display unquestioning loyalty to the store; they were to try very
hard to sell but not too hard." (p. 11) In the face of these dilemmas, skilled selling was gradually abandoned after 1940 in favor of self-service following four decades of managerial attempts to make it work.

Wood and Kelly suggest a final example of the abandonment of detailed managerial control over the labor process in favor of a more flexible arrangement. "The significance of sociotechnical systems theory, in this context," they write (1982, p. 87), "is that it articulated this contradiction between...control over the labour process and the production of surplus value, and recommended the creation of autonomous groups."

In addition to these contradictions, Elger (1982) and others point out that the drive for capital accumulation can create conditions that hinder such accumulation. In particular, Elger mentions that the reorganization of the labor process creates new skills on which employers become dependent, just as they are freeing themselves from old skills. Furthermore, rapid accumulation depletes the "reserve army of labour" thus driving up wages and strengthening the hand of workers who want to resist changes in the labor process and the terms of employment.

The new research agenda

Running through much of the post-Braverman literature on the labor process is a call, overt or implied, for a more subtle analysis of the evolution of the labor process and of labor/management relationships under capitalism. For most,
this more subtle analysis must be built on an historical and empirical base, and some of the studies cited above have served to advance that agenda. Littler (1982) and Jones (1982) call our attention to the unevenness of rationalization and mechanization across workplaces, occupations, and industries. Elger urges that scholars attempt to locate the forms and phases of the transformation of the labor process more accurately than has been done (1982). Wood (1982) draws from the collective evidence of the essays in a volume edited by him (many of which have been discussed here) the conclusion that, in the study of the labor process, there is no single direction or process of change applicable to all workplaces. "The implication is that the quest for general trends, such as a progressive deskilling of the workforce, or general conclusions about the impact of new technologies are likely to be both theoretically and practically in vain (Wood, 1982, p. 18)."
CHAPTER THREE

AT THE BANK: PART ONE

This chapter—the first of two which describe the bank—begins with a description of the bank itself and of the methods used in studying it. Following that, a description of systems found at the bank and a discussion of systems work and its elements fill the remainder of the chapter. In Chapter Four the labor market for programmers and other systems people and its effect on the bank will be reviewed, as will the relationship between the systems area and the rest of the bank.

The Case Study

The site of this study was a large commercial bank with headquarters in a major American city. Although it has many offices abroad, the bank has been moving in the last few years to strengthen its position as a regional bank through affiliation with other, smaller banks in the region. The bank is both a wholesale and a retail bank. It employs about 24,000 people worldwide, of whom roughly 800 are in the information systems area of the bank.1

1 Obtaining even approximate figures on number of employees was unexpectedly difficult. These numbers came from the former head of the centralized systems area. The number of "systems" people is extremely sensitive to the definition used. By the most restrictive definition—employees in the systems area doing systems development work—the number is perhaps 500. If those working on communications, on systems developed entirely in business "user" areas, and working in affiliate banks on systems are included, the number may swell to as much as 2,000.
In the 1980s the bank has undergone rapid change. The deregulation of banking, begun in 1981, has caused upheaval in the industry.2 As banking has become more competitive, banks have become more like other corporations in competitive industries, concerning themselves with strategy and with the marketing of banking "products." One of the bank’s responses to increased competition was to undertake a strategic planning process with the help of a major management consulting firm. One outcome of the planning process was a change in the structure of the bank. The new organizational structure, which was being put in place when the interviews for this study began, is shown in Figure 1.

While the thrust of this research was cross-sectional, aiming at an understanding of how work is presently organized in the systems area of the bank, it is possible from the evidence gathered to impute a sketchy chronology of some of the chief events in the systems area over the last twenty years. This chronology could form the basis for conjecture about cause-and-effect relationships between significant events of the three decades leading up to the present moment.

By 1957, more than 170 years after its founding, the bank had begun to computerize some of its operations. The systematization and even mechanization of the operations area of the bank was nothing new: much record-keeping work,

---

2 Sources used in this section on the subject of deregulation are: A. Gilbert Heebner (1985) and American Banker (1986).
fundamental to banking, had been standardized and specified for years, probably since the bank's first beginnings. What was new was the application of the electronic digital computer to the clerical "operations" tasks which are in many ways the heart of banking.

Over the ensuing years computerization grew steadily as more hardware was acquired and more manual record-keeping systems were converted to computerized systems. In the early days of computerization at the bank—probably for the first twenty years or so—the systems being created were "factory-type" systems which automated existing and well-understood transaction-based operations. During this period programmers worked at a site adjacent to the hardware they were using, while the professional bankers worked downtown. They had little contact with the users of their systems, except when analysis work had to be done. Programmers had a culture rooted in their technological expertise and in the counter-culture ethos of the late 1960s and early 1970s, and saw themselves as very different from the users of their products, especially the professionals in their three-piece suits. Both cultural and geographic factors therefore played a role in the chasm which separated programmers and business-oriented bankers, and which caused serious communication problems between the two groups.

In 1970 the ownership structure of the bank underwent a change. Taking advantage of a 1956 law, the bank spawned a bank holding company, which became its parent organization.
The bank was now a wholly-owned subsidiary of the bank holding company. At this time the bank also merged with another local bank. Whether these changes had a significant impact on the systems area of the bank is unknown.

A year later, the bank moved into its brand-new headquarters building in the downtown financial district. The building was larger than the bank’s planners at that time thought the bank would ever need, and the upper floors were leased out. Since then, the bank has experienced rapid growth and the headquarters building is more fully occupied by the bank itself, as are several floors of an adjacent office tower.

In 1976 most or all of the applications programmers and systems analysts moved from their remote location at the hardware site, called Columbia Park, to the downtown location. Today some are in the headquarters building and some are in the adjacent office building. This relocation has had the effect of bringing the systems and business people in closer contact, but at the same time it has reduced the contact between applications programmers and the systems programmers who provide them support services. The applications programmers have shed their jeans and long hair of the 1960s and are now wearing suits to work. Systems programmers, too, are dressing up, and as of 1987 male employees even at Columbia Park were required to wear ties to work.
At at least one point, probably including the late 1970s, problems of labor turnover in the systems area became so acute that there was a serious shortage of experienced personnel. This led to a one-time "market adjustment" in salaries, along with other changes in personnel management and perks. People at the bank still seem mindful of the risk of losing experienced programmers, but the problems with turnover were reportedly less severe than they had been within the last decade.

The Depository Institutions Deregulation and Monetary Control Act went into effect in 1980, phasing out some of the regulations to which banks had been subject. The 1981 McFadden Act report issued by the Carter Administration recommended a gradual relaxation of some limitations on interstate banking. Since that time considerable restructuring has taken place in the financial sector, and there are presumably many more changes to come. Interstate banking, the movement of insurance and other companies into product lines and services which one were the prerogative of banks, and the banks’ reciprocal move into what for them was new territory, and other changes have affected the industry. Banking is now far less circumscribed than it once was, bringing both unaccustomed vigorous competition and unprecedented new opportunity.

In 1981 IBM announced the release of its first personal computer, for which it appropriated the generic name "PC." This was not by any means the first commercially available
desktop computer, but it was the first time that IBM had put the weight of its vast authority in the business world behind the desktop computer. PCs established a major corporate beachhead only after the IBM machine was offered for sale, and by Summer, 1985 there were some 600 PCs at the bank. With the PC came the first opportunity for professionals outside the systems area to own and control their own computer and to oversee themselves the processing of information they needed in the course of doing their jobs. PCs sometimes allowed business people to perform analyses that otherwise would have been done—much more slowly—by the systems area. But they also, and probably more importantly, allowed new sorts of data analysis to be done. Particularly suited to the PC were ad hoc reviews of aggregated information.

By this time a new sort of system had emerged at the bank. In contrast to "factory-type" systems, which handled huge amounts of transaction-based data, "strategic" systems began to be developed. Their development was vastly accelerated by the arrival of the PC, but in fact many of the "strategic" systems were designed for use on the mainframe computer because they drew on very large files of data which could not be accommodated on a smaller machine. These strategic systems are of many types, ranging from those which provide novel banking services to customers to those which provide information for performance tracking and for use in strategic decisionmaking. What they have in
common is that they are systems different from the standard factory systems which are the traditional bread-and-butter of banking and which therefore look basically alike in all banks. Strategic systems are key to the bank’s ability to formulate strategy, and thus to its ability to successfully compete in the new deregulated environment.

In January 1985, after a three-year planning process, the organizational structure of the bank underwent a major change. A decentralization plan called for the creation of five major functional groups and a staff organization. The centralized systems organization was split into pieces and paired with each of the groups. Now, instead of reporting only to a single head of the systems area, the heads of each of the major systems groups is accountable to the group executive who heads the business area of the bank with which they are paired. Like the mid-1970s move of systems people to the bank’s downtown headquarters, this pairing of business and systems and "increased accountability" of the systems area to its client group has increased communication between systems people and their users.

In January of 1986 the outcome of the bank’s most recent job evaluation program was put into effect. Among other things, it specified the existence of two parallel promotion paths. The managerial path had been the standard promotion path, while the professional path was a codification of a system that allowed people with technical or professional expertise to be promoted without having to
become managers. Such a policy is important in the systems area, where technical specialists are needed, and where such specialists may have either no inclination or no aptitude for management.

**The Study Method**

The primary data source for the study was a series of extended, open-ended interviews with 25 employees of the bank. Initial interviews usually lasted about one and a half hours, with a range of from one half hour to two and a half hours. All except 6 of the interviews were tape recorded as a supplement to note-taking, and where extended quotes appear in the text they are transcribed from tape. The interviews were usually exploratory in nature, often with lengthy answers from interviewees and occasional direction from the interviewer.

After the bulk of Chapters Three and Four had been drafted, the manuscript was distributed to a subset of initial interviewees, with a cover letter describing in general terms the content of the other chapters. Thirteen interviewees reviewed and responded to the draft, most during brief followup interviews. No one who reviewed the manuscript believed that it contained pervasive errors, and most said that they found the depiction of the systems area quite accurate. Many suggested corrections of fact or of interpretation, and these have been incorporated into the monograph to the extent possible, except where two informants disagreed. In cases of disagreement the
information was either discarded, noted to indicate disagreement, or the view of one side was accepted if there was corroboration from other sources.

Of the interviewees, 19 were from inside the systems area and 6 from outside. While the outside interviews helped to provide some perspective on the systems area, the account of the bank given here is largely from the point of view of members of the systems area. Further research outside the systems area would yield useful information on systems "users'" perceptions of systems people and their products. Interviewees spanned a range of positions and years of experience, ranging from the then-head of Information Systems to programmer/analysts with one to three years of experience.

The first interviews were conducted in August, 1985 and the last of the followup interviews in April, 1987. However, the picture of the bank is a picture of conditions in 1985-1986. Much had changed for some of the interviewees by the Spring of 1987, and no attempt was made to reflect these changes in the study itself. Even in 1985 significant changes were underway, making research an exercise in hitting a moving target. Flux also made difficult the gathering of some seemingly simple information, such as job titles (the bank was changing from honorific titles like "Assistant Vice President" to functional titles) and number of employees in a given unit. The collection of statistics
and printed material were also hindered by the reluctance of many interviewees to give out such information.

Access to the organization was obtained through a colleague who had as an acquaintance a senior officer of the bank. The acquaintance provided introduction to his colleague, the head of information systems at the bank. After a brief interview, the systems head furnished the names of three others in systems, who in turn provided other names after interviews. Other than describing the sort of position or skills interviewees should have, the selection of interviewees was not guided in any way.

This study probably underreports employee discontent and criticism of the organization. Most interviewees seemed quite candid, but in the face of a stranger armed with a tape recorder who was not probing for criticism of the organization, silence on this point was not surprising. The truly discontented among systems people can likely go elsewhere with relative ease, but a few passing references to "organizational politics" and other matters suggested that there were some aspects of the work situation not fully aired in the interviews.

This study relies for its data primarily on the reports of individuals, rather than on survey data or on independent observation. This method is particularly well suited to exploratory studies, whose object is to create a first map of a new terrain. Since no case studies of programmers in
their institutional context exist, there was little prior work which could be used for initial orientation.

The perspectives used in the analysis of the data thus collected derive from many disciplinary homes. Anthropology provides tools for the investigation of individuals in their complex cultural milieu; psychology offers ways of thinking about individual behavior and how it is influenced by personality; sociology considers the behavior of groups and seeks patterns therein. Organizational structure treats the relationship between the structure of the organism and its characteristics, its behavior, and its effects on members of the organization, while labor economics studies internal and external labor markets, and the economic dimension of corporate life. Finally, industrial organization helps to make sense of the environmental changes—for example, in the level of competition—which influence the institution.

For a full investigation which retains some measure of the complexity of the original object of examination, it is the contention of this study that the multiple perspectives provided by interdisciplinary work are essential. Each perspective contributes to the fullness and dimensionality of the final picture. Each of these disciplinary bases is essential as a jumping-off point for the study, since each provides the ground on which to stand and from which to view the organization. Yet the use of multiple disciplinary bases creates a richer understanding. To the extent that

3 See H. W. Smith's chapter on Triangulation (1975, Ch. 12), which makes an excellent case for this position.
this study succeeds in its goal of portraying the complexity of the programming task and its content, it is through the use of multiple methods applied to a single object of study.

Findings of the Case Study

The "project" is in many ways the basic unit of work organization for people in the systems area of the bank. The existence of projects brings work groups together and joins them for a specific purpose. Projects define and organize the passage of time, and also help those who participate in them to understand that their effort has a beginning, middle, and end, each characterized by a different sort of work and social interaction. Projects generally result in the installation of a new system or the alteration of an existing system, and when systems people talk about their work, they often talk about specific projects and about the systems that those projects created. As an introduction to the work of systems people at the bank, several systems development projects are described below.

The "project" is in many ways the basic unit of work organization for people in the systems area of the bank. The existence of projects brings work groups together and joins them in a specific purpose. Projects define and organize the passage of time, and also help those who participate in them to understand that their effort has a beginning, middle, and end, each characterized by a
different sort of work and social interaction. Projects generally result in the installation of a new system or the alteration of an existing system, and when systems people talk about their work, they often talk about specific projects and about the systems that those projects created. As an introduction to the work of systems people at the bank, several systems development projects are described below.

1) Corporate Banking System (CBS) is a management information system used for tracking the accounts of the bank's corporate customers.

2) Customer Management System (CMS) provides aggregated information on retail customers and is used by account officers for decision support.

3) Massachusetts Consolidation Project converts existing systems so that the bank an its affiliate banks have compatible data processing capabilities.

4) Relationship Planning System is used by loan officers to oversee corporate accounts, and by top management to examine overall performance.

5) Strategic Planning Project provided data processing support to the bankwide strategic planning effort.

6) "ATM" System permits the bank's retail customers to conduct standardized transactions through a network of Automatic Teller Machines.

7) Commercial Loan System (CLN) is used for tracking loan balances and payments.

8) Performance Reporting System (PRS) tracks budgets and actual performance of
individual units within the bank, known as "responsibility centers."

9) "FRS" was a small system installed by a newly-trained programmer using a purchased software package.

A more extensive description of these systems can be found in Appendix 1.

Types of systems

The systems now in use at the bank fall into several broad—and not mutually exclusive—categories. First, there are the large, transaction-based systems which deal daily with a very large number of transactions. The bank has several such systems—the Demand Deposit Accounting (DDA) system, which tracks demand deposits, the Time Deposit System (TDS) for savings accounts and other time deposits, and the Stock Transfer system, to name just three. Second are the Management Information Systems such as the Performance Reporting System (PRS) and the Corporate Banking System (CBS) which help management monitor performance of individual departments and groups. Decision support systems, which provide data to account officers and other nonmanagerial professional employees are a subset of MIS. Small decision support systems might be run on PCs, but they might also be housed on minicomputers or mainframes. Many of the larger systems can be used both for the processing of management information and for decision support. Third are office automation systems, generally used for word processing, preparing mailings to clients or others, and
other similar functions. In the past, office automation has been entirely separate from data processing, using different technology and having different goals. Now there may be some convergence of office automation and data processing taking place. Finally, there are systems which are "customer-oriented." These may be used directly by the customer, as in the case of the "ATM" system, or may allow the bank to offer a new banking product or service.

"Factory" systems and "strategic" systems

A manager at the bank suggested that their computer systems fell into two broad types, which she called "factory" and "strategic." Factory systems are the large, transaction based systems which the bank puts to heavy, daily use. They are, in the words of another manager, the "backbone" of the bank. Strategic systems, which she found more difficult to define and characterize, are those that for reasons of scale, content, or purpose, are not appropriate or feasible in a mass-processing environment. Others at the bank agreed that these two sorts of systems existed. Said one:

"The factory types of systems are the systems that were being developed when I was a programmer and are now undergoing periodic redevelopment. The stock transfer application that is being developed right now is a good example. It's aimed almost entirely at further automation of information flows and paperwork. It's being focused on a particular business unit. It's very large and takes a lot of programming. It's a lot of standard accounting kinds of reports and functions that just help you perform and control the basic operations of the area. Most of those systems that we need have been developed at least once, and most of them more than
once, and so the situation with regard to these factory systems is that we now have to maintain them and we periodically have to redevelop them. Strategic systems tend not to be systems in the old sense of hundreds of programs that function as a unit to automate some large business function. Instead they tend to draw on existing systems and data, and they have a specific purpose that is closely related to some concern of senior managers that may or may not be an ongoing permanent concern, but it might shift priorities a great deal from one time to the next. So generally they're related to decision support for the people fairly high up in the organization who are making important decisions with respect to issues of strategic importance to the organization that cross organization lines, and the challenge of them tends not to be in developing a massive amount of software, but in being clever about developing a small amount. It's clearly an area that we're getting more and more involved in. Factory systems are becoming less as a proportion of our total work."

Another manager put the contrast somewhat differently:

[Processing] Mastercharge is a speed and efficiency issue. Once you write that code for that transaction, that big mainframe can do a million of those a minute...The guy who has to figure out how to price that Mastercharge product only does it once. He may reprice as interest rates change, but you're not going to assign a team of COBOL programmers and make a mainframe available for the guy who prices Mastercharge. That COBOL program is going to have to have a place in its activity cycle that says, 'insert field such-and-such' which is the price, but how that price gets determined is how the bank makes profit. We save money on speed and efficiency; we make money on income generation."

Factory systems

"We process one billion dollars worth of checks per day," said a systems person working on PC applications.4 "You're not going to do that on a PC." Large volume, transaction-based systems from the "operations" area of the

4 According to one manager, the number of checks processed daily is about one and one-half million.
bank were the bank's first systems, and continue to be central to the effective running of the bank. The processing of data is so fundamental to banking that without these systems or their functional equivalent, the bank truly would not exist. These operations-based systems are the prerequisite to all the bank's marketing activity. "You've got to be able to count the beans before you distribute them," said one systems person.

Strategic systems

The manager who originally suggested the distinction between factory and strategic systems described a strategic system as one that was "leading somewhere--and that somewhere might be to the wastebasket." She continued,

[A strategic] system shouldn't be funded in the same way as a factory system because there is R&D implied. The management for the development process for that type of system is different than for the factory. It differs in how hard and fast the specs are, the testing process, and the nature of the lifecycle. I haven't really found a good word for it yet. You're not really following a lifecycle. You're cheating. You're knocking out big parts of it.

Not only are strategic systems constructed using a different development process, but the outcome is different as well. If development techniques appropriate for strategic systems are used in building a system that is later put into a "factory environment"--heavy, frequent use--the system must often be redeveloped.

...when you get to the point where you decide you want to put it in the factory you have to go back and really walk through the lifecycle to make sure that you've taken care of what you should be taking
care of. Because you can’t afford instability and cost inefficiencies and all the rest at that point. But you sure can afford them when you’re in the strategic type of system.

Of strategic systems there are several (possibly overlapping) kinds.

1. Systems which offer customer-level data. These offer data at a more highly aggregated level than do the transaction-based systems. Customer-level systems are used by managers to inform their decision-making. "We’re providing systems for [business] analysts--a product manager or a regional manager--someone who could take the data and wants to play with it, massage it, do things with it to make a decision," says a manager in one of the bank’s Information Centers. This use of data she wouldn’t call strategic, she said. Strictly speaking, she felt that current systems helped managers with individual customers, with assessing profitability, and with managing customer relationships, not with setting bank strategy. "When we have a system that would allow the bank to change its policy, or things are so well organized that somebody could log on and say, ‘Where is all the information I need in order to make a policy change for the bank?...’", then she would say that the bank had true strategic systems. For the moment, she says, there are no such systems at the bank.

2. Management Information Systems which are used to obtain an overview of the bank’s performance in a given area, or as an aid to the formulation of corporate strategy. These systems may be in principle be used by managers at
various levels, although in fact there are reportedly still no systems being used directly by top management. Top managers do, however, make use of the data from reports generated by management information systems.

3. Tools for professionals / Automation of professional work. Increasingly, systems people at the bank are focusing on providing computers as tools for professionals at the bank, and eventually on automating some aspects of professional work. The Advanced Technology Group (ATG) has as its sole mission the delivery of PC-based tools to professional workers. Their target group, according to one member of the ATG, is income-producing professionals—lending officers, treasury traders, investment bankers, and so forth: "Someone that turns a buck for the bank instead of someone that counts them, like me."

[The income-producing professionals] would very much like to have the ability to use technology in, say, structuring a loan pricing deal or a debt placement deal. They can’t wait for feasibility study, functional spec, general design, detailed design, coding, debugging, testing, implementation, training, and documentation...This has to be done next Tuesday or the customer walks.

In fact, this manager says, professionals do a range of things, some of which are suited to the traditional systems environment and some of which are decidedly not. The variety found in professional work is a challenge to the adaptability of the systems organization.

There are a lot of procedural and processing elements in what professional level people do that can be better handled by a more traditional approach to systems development than what we use...If COBOL does the job then there’s no sense in messing around
with it. [On the other hand,] there is a high end that is very much related to AI [Artificial Intelligence] that we haven't made a great deal of progress on, simply because we find that the higher up we go...the more complex problems...are less structured and inherently don't lend themselves to automation...i.e. somebody at an executive level doesn't just whip out a manual that says, "Here's how I solve a personnel problem" and walk through that. It's much more an experiential thing, and based on the number of inputs, the types of inputs, and what they say. Track the solution process, and [you'll see that] they jerk it from side to side and in several directions. It's not something you can lay out on a map. It is much more complex even than a typical decision tree where you just keep getting yes/no's and branching out to different areas. It's softer in terms of what technology is available to help them but it has the promise of being a very rich area for us to try and mine.

4. Systems which offer new customer products.

Finally, there are systems which are the basis for new customer products. These may be for direct use by customers, as in the case of the Monec system, or they may be systems which make it possible to provide new services. A member of the ATG provided an example of the latter sort of system at another financial institution:

...you can take the technology and use it in something that Merrill-Lynch did extremely successfully years ago with their cash management account, where they literally combined your transaction accounts, your brokerage interest, and throw in a VISA card, and presented all that information on a single statement, yet allowed you to do margin buying and selling of stocks. It's a huge, huge development cost, but before anyone else could do it they'd captured four billion dollars in funds.

These customer-oriented systems are in fact a hybrid form, uniting factory and strategic systems. They grow directly out of the bank's strategic moves and tend to be at the leading edge of the bank's growth and development, yet
they are often large, heavily used, transaction-type systems.

Evolution and proliferation

Everyone at the bank agreed that strategic systems occupied an increasing proportion of the time of systems people. Factory systems are not an endangered species, however. No one at the bank foresaw a day when there would be no use for largescale, centralized computing power. Indeed, one manager said that although factory systems occupied a declining proportion of the time of systems people, the time spent on factory systems had not declined in absolute terms.

As these systems get bigger and more complicated there’s more and more software to maintain, so even though there are fewer ongoing development projects, the ones that we do have are more massive and the systems that are past the development phase require more time and attention. As we increase our efficiency, the work is also increasing, probably at least as fast.

To what extent are strategic, small-scale systems replacing factory systems and to what extent are they supplementing large systems? It appears that some of each is taking place. Here are the comments of four members of the systems area:

[The technology is going to the end user, and that’s as it should be]...because they’re tools. You don’t have to be a skilled craftsman like the guilds of the middle ages anymore. There’s a variety of tools that have been simplified to the point that they can be made available to virtually the general public.

User computing hasn’t diminished the traditional centralized operations. Instead, it’s for back-of-the-envelope calculations. But user computing may
reduce maintenance costs, since much of maintenance is report generation.

[The Marketing Research area is doing a competitive pricing study using the purchased software package "Symphony" on a PC.] I imagine that if they did not have a PC then they would either not be able to do it or they would have to go to the mainframe and ask us to write a system for them.

Clearly the business people will do more and more of their own work in areas that have traditionally been left to systems. That is an evolution that is already taking place, it's bound to continue, and it's highly desirable.

Systems Work

It is common for those who work in systems development to describe the development process as having a characteristic lifecycle. Both in interviews at the bank, and in textbooks on systems development, the lifecycle is said to consist of a series of stages through which the process passes on its way to completion. There is considerable variation in nomenclature from workplace to workplace and from textbook to textbook. In Figure 2, the terms used at the bank to describe the stages of the systems development lifecycle are enumerated, and each stage is paired with the task that is dominant during that phase of the development process.

Figure 2

<table>
<thead>
<tr>
<th>Stage of Development Process</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feasibility Study</td>
<td>Analysis</td>
</tr>
<tr>
<td>Functional Specification</td>
<td>Analysis</td>
</tr>
<tr>
<td>Detailed Specification</td>
<td>Analysis</td>
</tr>
</tbody>
</table>
Analysis

It is the job of the analyst to develop written specifications for the proposed system. These documents, the tangible product of the analysis process, serve as a guide for the coding process which follows analysis.

The analyst's task is to transform an often complex and ill-defined process into a set of procedures. Because computers will only accept instructions that are complete and unambiguous, the job of the analyst is to express completely and unambiguously in English prose what the system is to do. The written specification is then translated into computer code during the coding phase.

Five, ten, fifteen years ago we were constantly complaining about, "Oh, those darned users, they changed the spec on us, and they don't know what the heck they want." You know--the usual song and dance that people give. The truth of the matter is, they can't know what they want, and design is an iterative process. If you've ever built a house, you know that until you see these things sometimes it is very difficult to visualize and to understand.

In marked contrast to the immense concreteness embodied in a finished computer program, analysis is a highly varied and nebulous activity. Used in its broadest sense, analysis is also pervasive to the entire systems development process.
The analyst has to do several things:

Understand the work of users and the context in which they operate.

Help users develop and clarify their desires and objectives, tying these to business opportunities and constraints.

Know at least in general terms the characteristics and limits of the technology.

Know about the data used by any given system.

Each of these activities has several components.

1) Understanding the work and context of users.

There is increasing stress in the systems area on spending the time necessary to understand not only the work of users but the context in which they operate. Systems people claimed that this knowledge was critical to designing a good system. A woman from the corporate lending area of the bank went "on loan" to MISD during the development of the CBS system. While there she acted as a user representative to the systems development group. Afterwards, she said,

...I was ready to come back to the lending side. I could have stayed at MISD, [but] the only way you know that it is successful or can guarantee that it will be successful is to get back out there and see--Is it being used? How is it being used? Do people understand what's going on? And as you see things wrong, trying to get them corrected. Or did we totally misjudge an area? You can't be removed for too long and continue to create for lenders. The marketplace is changing too swiftly to sit in a far-removed place and expect that what you're doing is satisfying the needs of people who use it.
Insight into the work processes and needs of user areas comes with familiarity, "working with people on a day to day basis," and with practice in the techniques of inquiry necessary for analysis. As the manager of an information center, who deals with user requests daily, said:

The major thing [in doing analysis] is knowing the right questions to ask. For me that’s something that I never learned in school. I just learned it because I made so many mistakes... I would give the person a report and they would say, ‘Now I want to know this.’ So I just started remembering to ask them. Also, I could start to look at it, too, and think, what would you like to know, how would you go about making this decision, what information would you need to know? I think the main thing is to know the objective. That is crucial. You’ve got to know why they need this information.

2. Developing and clarifying objectives

Knowing the objective is not always an easy task. It often involves many hours of meeting with users, and sometimes a series of false starts or even rejected systems before the objective becomes clear. This is the sense in which the manager quoted above meant that design is an iterative process. An analyst who knows the work context of users is at a great advantage when trying to design an effective and appropriate system, but this knowledge by itself does not insure a clear objective.

The manager who runs the information center said that about half of the requests she receives are from people who have very definite ideas of what they want, and the remainder are quite vague. She gives an example of a request:
A woman wanted to know who her top customers were by total balance for all their relationships with the bank. She really wasn’t sure what she meant by top customers...I thought maybe I should extract every single customer balance and then do a cumulative frequency distribution on that, so then I could see that 30% of the customers had balances of a particular amount or more. So I did this, and it took three lines of code. Once I had that, I told her, "Well, 80% have balances of this amount or less, 20% have more." So she said, "How many have this number?" I told her.

In this way, they arrived at an agreeable cutoff balance and number of customers, satisfying the requester’s idea of what a top customer was.

One of the most difficult aspects of analysis is uncovering and resolving communication problems between user and analyst. Because the systems development process involves making precise what may be vague or tacit, it may demand of systems developers unaccustomed precision in the use of language. Problems of communication are further compounded by the fact that systems people and users are from two different cultures within the bank. These two cultures use different terms and have significantly different points of view. As a result, extra care is needed to avoid systems problems caused by miscommunication.

What we really want to do is make sure we’re answering the right question, performing the right work, before we do the work. One of the standards that we have for the application teams is that they write what we call a function statement back to the requesting department and anyone else that’s going to be using the system, that in our words—and we try not to make them tech-y—parrots back to them what we think they’ve asked for. Even if the user has written a wonderful specification, we want to reshape it and parrot it back to them to make sure we really do understand what we’re doing.
3. Knowing the technology

Because in the process of system design the technology provides both constraints and opportunities, it is important for analysts to have some familiarity not only with computers in general but with the specific language and computer system which will be used in constructing the system they are helping to design. "We're supposed to understand the ripple effects for changes to the system," said one analyst who does not write code. It is not impossible to act as an analyst without such knowledge, but those who are unfamiliar with the technology find that they must work closely with more technically-oriented systems people. An analyst who is part of the CBDG group expressed a strong interest in learning to write code, and said that it was frustrating not to understand the limits of the system. She found herself wanting to try different designs for the system--ones that would do more--but unable to do so because of her limited knowledge of the technology.

Those who do have a more technical background report using it. A manager who writes a lot of programs for ad hoc reporting in an information center said that when she first came to the bank from another programming job she "read a lot of source code." This helped her to quickly understand the structure of the systems she was working with, from which she would be drawing data.

4. Knowing the data
Not only do analysts need to know about the technology and about the users’ context and intentions, but they need also to be familiar with the data files on which the system might draw. To use the data properly it is necessary to know not only what the data cover in general terms, but also many specific features of the data set. Unfamiliarity with the data set’s properties—including transient problems and subtle characteristics—can lead to serious errors, as the account of the Information Center manager makes clear:

We have one manager who wanted to determine how well they had done selling their product... Our file is not set up to tell him how well they did over the last six months. All it has is today’s balances. So first of all I need to know exactly what he needs. Some of the information he needed was, how many new customers did I get? How many old customers went away? So given those information needs, then I did some analysis to figure out how best to read the data. Then after the report was created, there were more meetings with him and some of the other operations people to see, are these numbers reasonable? And if they’re not, what can we do, either on the systems side or on his way of thinking, to make them more reasonable? In this particular case it appeared that there were many, many more new customers than they ever imagined, so they were saying, "Why is that?" And it came up that the operations person said that a lot of social security numbers had been put in wrong, and that had been resolved within the last few days, so perhaps if I just try running the report again on the updated file, then the numbers would be different, and that turned out to be exactly the case. The numbers now do meet their expectations.

This manager said that one of the key qualifications for her job was an interest in the data and in what the data items mean. This is crucial, as her story shows, to producing an accurate report. "If you said, this is how much money our product has earned in this time and then it
wasn’t, not only have you produced a wrong report, but a bad report is much worse than no report." One of the questions she has learned to ask of clients, she said, is what information they already have that she can use to verify her results when she is preparing a report for them.

Although systems people talk about "analysis" and "programming" or "coding" as if they were two distinct activities that occur serially during the systems development process, one finds that in practice the distinction between the two activities is less than sharp. "Analysis" is generally understood to include the activities that lead up to coding but not the writing of code itself, "coding" is generally thought of as describing the relatively narrow, circumscribed activity of translating English-language specifications into COBOL or some other computer language. "Programming" is a more vague term, certainly including coding but often encompassing considerable analysis as well.

These terms are problematic because they are trying to distinguish between activities that are so closely and seamlessly linked as to be virtually indistinguishable. When one programmer/analyst who does a lot of ad hoc reporting, and thus a large amount of both coding and analysis, was asked when she stopped doing analysis and started to code, she replied that even when she was coding she was still doing analysis. Perhaps she was not doing
analysis when she was actually at the terminal with her fingers moving across the keyboard, entering coded phrases; she said, but that was a tiny proportion of her time. When her fingers stopped moving and she paused to reflect on what she was doing, or to rethink what she had just done, she was back doing analysis, she said. Her description suggested that asking a programmer, "when did you stop doing analysis and start coding?" is rather like asking a writer, "When did you stop thinking and start writing?" That is, the question implies a far greater separation of the two activities than is possible, and thus the question itself is almost nonsensical.

A non-programming analyst, who works as a team with a programmer during systems development and modification, said that they go to meetings together and that their knowledge complements each other's. Yet she also said, "We do half the programmer's job because we come up with the specs." This suggests that she believes that programmers should be doing analysis as well as programming. She added that her goal was to become a programmer/analyst, saying that she thought this was the only sensible way to organize the work.

On the whole, systems people saw themselves as following the lead of the "business side" of the bank, serving the strategic, marketing, and production needs of bankers. Rarely did they suggest areas in which they might exert companywide leadership, except with respect to the
application of computerized systems. They explicitly aimed to create systems that served the needs of the users and that reflected the bank accurately. As one manager in MISD said, "Organizations have certain implications." The job of systems people, she suggested, was to find out what the key decision processes were and to use those as the basis for the systems they build. However, despite their efforts to reflect the organization rather than to direct it, there is a real sense in which the systems development process and, after it, the use of computerized systems, changes the organization and the jobs of people in it.

Large systems are built at the bank in ways that impose standardization on the process being computerized. In the case of the large transaction-based systems which were the first sorts of systems at the bank, the standardization came first, even while the process was performed manually. The new systems may have been a step up technologically but did not necessarily represent a quantum leap in the level of standardization imposed on the process. Systems in other areas, however, may impose new rules on what had been a more flexible process. The analyst at CBDG told of her participation in the design of a system which, among other things, incorporated the bank's rules for the conditions under which the bank would "punish someone for nonpayment." These rules had existed long before the system was built, but they were "in transition" at the time of the system development effort. It fell to the designers of the system,
including her, to decide which rules to follow and how to incorporate them into the system. It is possible, she said, to write in functions requiring human input, so that for areas in which officers had been making decisions they could still make those decisions rather than having them relegated to the computer. Still, the system, when built, reflected a set of decisions about how the organization would and should operate. These decisions are generally not made by the systems people acting alone, but neither are they generally made by the entire corpus of users whose work is affected by the new system.

As a generic activity, analysis has a long history at the bank. Long before the 1950s, when the first electronic digital computers were installed, banks had teams of analysts whose job it was to study the work of employees, generally clerks who participated in high-volume, standardized activities. The goal was to increase the efficiency and reduce the cost of these activities by reorganization and automation. Such analysts, the predecessors of the systems people now at the bank, were called by one long-time systems employee "the potential Henry Fords of paper shuffling."

When computers started arriving at this bank, one employee reported, some of the analysts followed a computer-oriented path while others continued to rationalize work, using methods other than computerization. Until very
Recently at the bank there continued to exist a small band of non-computer-oriented analysts in the "Systems Research" department. Systems Research has since been folded into the larger computer systems area.

Today, especially now that the Systems Research area has been absorbed, a manager in systems reported that:

It's...part of our mission not to assume that we're going to computerize everything. That's simply one of a number of alternatives. So when business analysts go into an area what they're basically doing initially is trying to keep an open mind about what the problem is and whether it can be solved by a variety of means including computerization. Systems has never been exclusively associated with computerizing things here at the bank.

Systems people, he reported, are brought in when there are problems which are seen as problems of information flow or information processing. Systems people try to present alternatives to computerization, such as the elimination or "farming out" of work.

I would have to admit that we're highly oriented toward computerizing things, and I'm sure that we miss opportunities for other kinds of solutions because of that orientation, but perhaps with the way things are going these days that's not something we should be overly concerned with, since more and more things these days are amenable to being aided by computerization.

Coding

The actual writing of computer code involves the translation of specifications into a very concrete, specialized, technical language. By contrast with analysis, which requires the analyst to study and dissect a work process and an organizational context, the writing of code is relatively context-independent. Programming languages are
highly standardized, although there may be some variation in
the use of a given language, say, from firm to firm. For
example, one company using COBOL may have an approach
different from that of another company because the first
company uses different hardware and systems software and
requires different techniques of its COBOL programmers.
This would lead to the existence of different "dialects" of
COBOL at different firms.

Most systems people at the bank spend a few years
writing code and then leave it entirely in order to focus on
other activities, chiefly analysis and management. Those
who especially enjoy coding may choose to take a more
technical path, but even then they find that they do not
spend all of their time simply writing code. Even some of
those whose jobs leave them little time for coding sometimes
like to return to it, as one member of the CBS development
project reported. "We were asked to do 18 months of work in
6 months, and consequently we all chipped in and did what we
could." He hadn't written code in four years, he said, but
he enjoyed doing so again. "Rather than trying to bring
someone over to your point of view on why we should design
our specs for the system this way...you just sit down and
interface with the machine. The compiler doesn't give you
any opinions."1

1 A compiler is a piece of software which transforms
the actual code written by programmers (called "source
code") into more streamlined code readily readable by
the computer ("object code.") If the source code does
not follow the correct rules of syntax, punctuation,
spelling, etc., the compiler will generate an error
Testing

During and after the coding process, the testing takes place. Testing is necessary whether the system is large or small. For large systems the testing may take place in several stages, as individual pieces are first tested and then gradually joined into a larger whole, being tested again as pieces are added on.

There are two main types of testing. The first tests for logical consistency of the program, and for the extent to which its logic conforms to the specifications. The second tests the system's performance in practice, as it performs when in use by its intended users.

A programmer/analyst from an information center describes the testing that she does of the small reporting-oriented systems she creates. Before she sits down with the client to look over the finished system, she checks the system by creating mock data that she can use to test "each and every condition, each and every path that a data item travels." She then checks to see whether the output from the system is what she expected it to be, given the data.

While this first stage of testing will find logical flaws, it will not discover problems in the original design of the system, nor in the data sets themselves. A young systems employee said,
We can test the logic before we hand it over to the users. But at some point we hand it over to the users and testing is their responsibility. If you've got a spec in front of you that outlines, yes, this is what they wanted, I guess to that extent you can say, yes, this is what the users wanted, but it never really goes that smoothly.

One of the tensions in the process is caused by the disagreements which inevitably crop up when the system is tested by the users. Especially when the users have not been involved all along in the design process, they find things in the new system that do not work as they would like them to. The users might have in mind the sorts of functions they want, but the systems people can be invested in what they have already developed, and the systems people know the technical constraints better. Sometimes there is conflict over how much fixing of the system the systems people should do: are the changes considered to be part of building the system right in the first place, or are they considered to be "maintenance" of a finished system, and therefore subject to being put into the maintenance queue until a systems person is available?

**Implementation**

"Implementation" is an umbrella term for all the activities that get the system up and running. Testing, which precedes it, and maintenance, which follows it, are thus in a real sense implementation activities also. Testing involves introducing users to the new system and discovering how it performs for them, and maintenance is concerned with the care and improvement of the system after
it is installed; yet since for most systems there is never a time when they are clearly "finished," the line between implementation and maintenance, as with the line between implementation and testing, is very blurred indeed.

One of the major components of the implementation phase is the training of users--the people who will use the system directly and the people who will be affected by it indirectly, through reports generated on the system and so forth. Generally, the systems people involved in the development of the system also do the training. Two major training issues were mentioned during interviews with people involved in systems development.

The first issue concerns the use of systems people as teachers about the new system. In some sense they are an obvious choice as teachers, since they know the system well. But this familiarity can be a liability when someone immersed in systems concepts and language forgets what it is like not to know about computers and is therefore unable to present material appropriately. As someone from the office systems group said,

The problem is that we do this for a living, and we assume other people to be as interested and capable as we believe ourselves to be. So when you walk across the street and talk to an end user about reasonably basic office systems technology, you have to go, "Wait a minute, this person really doesn’t know what I’m talking about and it’s not their fault."

An analyst who acts as a go-between between systems and a user group said that it was hard to remember that she had
to "present" the system to people, since she herself was so familiar with it. "You can’t just dump it on them."

The second issue in training is that trainers report that teaching users is a very labor-intensive activity, and that users learn best when there is someone around who knows the system whom they can come to as needed. The head of the Professional Computer Center (PCC) reportedly played this role for professionals learning to use PCs. A user representative to the CBS development project who now does some training in a user area said that the fact that she was around and on call for users of the system made it easier for people to learn to use CBS. A participant in the CBS project was critical of the implementation process used in the introduction of CBS, feeling that there hadn’t been enough attention paid to the post-development process of making sure the system was truly useable and being used. The CBS people in MISD,

...should be coordinating more closely with people that are using the system to make sure that things are getting corrected, but in effect they have thrown it out to the outside world and said, "OK, it’s your problem, we’ve done the development. There it is. What more do you want?" I’m hoping it’s not going to be the kiss of death for the system...I think you have to hold these people’s hands through this whole process. It’s new. You’re shoving it down their throat. People see no immediate benefit from it.

Maintenance

Maintenance consists of the repair, alteration, and improvement ("enhancement") of systems which have already been installed. Maintenance is necessary when the system is
found to have a problem (often referred to as a "bug"), when a specific change is needed (for example, if the format of a data set used by the system were changed, the system would need to be modified), or when an additional feature or improvement is to be added to a system. The lines separating these three types of maintenance are not always clear. "We had a little research project," said a senior manager, "to try to define maintenance and enhancements...you can't get two people to agree. Some people consider that if it costs less than 'x' dollars it's maintenance."

In many ways, maintenance is systems work in microcosm. Projects are generally small, involving from one to a few people, and they must pass through all of the systems development stages through which new development projects also pass. Although maintenance, which involves dissecting an existing system in order to effectively change it, is different from building a new system from scratch, it requires many of the same skills and can be extremely complex and difficult to do well.

When building new systems, inadequate analysis can lead to the construction of a system that performs poorly and thus creates more problems than it solves. This is even more true of maintenance, in which a seemingly minor alteration can have disastrous consequences. As one senior manager in systems said,

[Maintenance] always was a situation where relatively little coding went on, and it probably
always will be, I hope. You start adding a lot of new code to an existing running system and you’re going to create a lot of problems. So typically in that situation the systems people are spending a lot of time dealing with the users, discussing the needs they have, how the system could be changed. They’re spending a lot of time carefully testing the changes that they make, and relatively little time producing code.

A more junior programmer/analyst had spent the morning before her interview fixing a problem she had introduced into the system the previous day while making a minor change. The problem she created had caused data entry clerks trying to change an interest rate to get an error message which said, "Program canceled due to programming error."

Maintenance is a key activity at the bank because systems are so critical to its daily operation. A bug that shuts down a crucial system can virtually shut down that aspect of bank operations until the problem is repaired. A programmer who had been at the bank for a year said that "production problems," which she defined as "anything that makes the bank lose money, as opposed to simply creating an inconvenience," were top priority for maintenance work. Continued oversight and effective intervention by systems people are necessary to keep the bank’s systems running.

Maintenance work is increasing as a proportion of all large systems work. In 1985 a manager in the systems area estimated that the ratio of maintenance work to new development work was roughly one to one or higher. The ratio is increasing because there is less development of
large systems taking place today, and because the development efforts of the past have generated more systems which need maintenance. One effect of the increasing predominance of maintenance over new development is that the total proportion of time that systems people spend on analysis is increasing and the proportion of time spent coding is decreasing. Because maintenance has a higher proportion of analysis than does new development, its growth changes the overall ratio of analysis to coding activity.

Maintenance can be minimized if the system is designed well in the first place. Rush projects, poorly conceived systems, convoluted code, and a host of other design problems can create a system which badly needs maintenance, yet which by its design makes maintenance more difficult.

In the larger software community, maintenance is looked upon as dull, routine work which should be done by dull, routine people. The exciting projects, in the ethos of this community, are those that involve new development, particularly state-of-the-art technical work. Maintenance workers are the cleanup crew who come in after the party is over.

At the bank, however, the stigma on maintenance work is not pervasive. Some systems people, especially those who feel themselves particularly allied with the software engineering profession or who are wedded to doing new development work, are eager to avoid maintenance work. Others, particularly recent hires who have been trained at
the bank and do not have professional software experience elsewhere, see maintenance as necessary work in which they are willing to participate.

Typifying the anti-maintenance attitude is a manager who has spent 15 years in systems at the bank. At the end of a long systems development project, he said, "all those that can, immediately try to get off it so that they're not left in a maintenance role." He says with evident pleasure that only two of his years at the bank were spent doing maintenance. A member of Misd was critical of the fact that Misd was still involved with the maintenance of the systems it had built. She believed that the people doing maintenance, who were at the low end of the Misd hierarchy, should be split off into a separate group. Even a nonprogramming analyst, who was eager to learn more about technical matters, said that the post-implementation stages were tedious. "You analyze for a year, bring it up, then debug it. It gets kind of boring."

On the other hand, there are systems people who willingly, if not enthusiastically, take on maintenance projects. Entry-level systems employees often start on maintenance work, and continue with it until they are promoted to more prestigious work. The new programmer who reported that her team was split in two soon after she arrived, with the more senior people being relieved of maintenance, seemed to accept the notion that more interesting work was a prerogative that went with seniority.
Presumably junior systems people will have the opportunity
to do work other than maintenance as they become more
senior.

The head of an information center at the bank felt that
the bank had had some success at removing the stigma from
maintenance work because of its practice of training fresh
recruits.

...people come in from school with no experience and
they're not told that production support [i.e.
maintenance of production systems] is really boring,
doggish work. They don't hear that and they see
that what they're doing is sort of interesting; I
mean, you're learning about banking, and some of the
systems are quite sophisticated...They feel what
they do is very important, because it is.

The proof of the fact that maintenance is not a
thankless, lowly job, she continued, is that it holds the
attention and commitment of people who are not at the bottom
of the hierarchy.

I see good people--superstars--who do this kind of
work month in and month out without being tempted to
leave, in a region where there are plenty of jobs.
I also see recognition being given for good work in
this production effort. People are rewarded or
given promotions for being on a team that does
basically production support.

Another manager reported an increasing tendency among
systems people to regard maintenance as interesting and
desirable work. He described this as still a "minority
point of view," but explained the growing status of
maintenance work as the result of three factors. First,
because systems are so important to the organization, their
support and maintenance are extremely important and very
demanding. Second, maintenance can be seen as relatively
high-status work, since it involves more user contact and less coding. Finally, maintenance work is more varied. Because of the shorter turnaround time of maintenance projects, a person doing significant amounts of maintenance work will change projects more often.

Systems people may also end up doing maintenance voluntarily because they become attached to a particular system and the business function that it serves. It is not uncommon for a systems person to become so interested in the business problems of a particular area of the bank that they choose to continue working with that area. Often this means continuing to work on a system whose development they may have had a hand in. Sometimes systems people even cross over to the "business side" of the bank, leaving the systems organization altogether.

With the decentralization of the bank and the reorganization of the systems area into small groups paired with business groups working on particular business functions, the continuity of systems people on a particular set of projects is institutionalized. This helps encourage systems people to become engaged with particular business problems, but may also reduce the mobility of those who want to avoid maintenance.

Additional Tasks

There are aspects of systems work which are not a part of the systems development life cycle as such. These are
managerial and support functions, and a new set of tasks, here referred to as **ad hoc** reporting.

**Management**

Management consumes a significant amount of the time of the more senior systems people. The lowest level of management in systems is that of the project leader; the highest level is that of the senior vice president who oversees the whole systems organization. There are also senior managers in systems who are paired with the five group leaders who oversee the major divisions of the bank. Each group leader has a senior manager who reports to him and whose function it is to oversee the systems work of that particular group. In between the project leader on the one hand and the senior vice president and five or so senior managers are a couple of levels of managers within the systems area.

Managers perform a variety of labor-intensive tasks, including staff development; planning and oversight of projects; checking that projects fit in with the bank’s goals and objectives; meeting with users, especially supervisory and managerial users; cajoling, coercing and bargaining; and handling salary and personnel matters. One manager described her work as follows:

My role in systems is setting policy, methodology at a high level that is carried out by managers. Oversight. Making sure we are marching down the correct road. And quality. My major job is really interacting with the user community...to make sure the communication flow is working, that we are doing the right things...On key projects, much of the
success of those projects depends on folks not under my direct control getting along with one another, understanding their roles, doing their thing. So a big part of my job is the sort of informal mover, understanding what the projects are set to do and what kind of stumbling blocks are in the way.

Another manager reported that as he rose up the hierarchy he found himself "looking more and more at generic, long-term issues."

Every manager interviewed had a systems background and had been a programmer at one time. A large number of these managers had in fact begun their systems careers at the bank, and many had gone through the bank’s training program for programmers. A manager said that managers in the systems area continued to be involved in technical issues, but that management of large teams required considerable administrative work, and this tended to remove the managers more from technical details. Indeed, removal from the day-to-day details of design and execution must be a necessary development for managers who believe it is their job to focus on the bigger picture.

Systems experts and systems programming

Two kinds of work within the systems area at the bank require relatively high levels of technical expertise: systems programming, and expert technical work in the applications area. Systems programming involves the installation, maintenance, and sometimes creation of the systems (made up of both hardware and software) which applications programmers use in building programs and in
running completed programs. Systems programmers, in the words of a former applications programmer, are "usually camped somewhere around the machine," unlike the applications programmers, who are generally located near computer terminals and closer to their business clients but remote from the mainframe computer installation. Systems programmers maintain the "tools" which programmers use, and thus applications programmers are the "user group" for systems programmers, just as the people on the "business side" of the bank are the user group for applications programmers. Just as bankers and applications programmers have communication problems which reflect the differences in the cultures from which they come, so too are there communications problems between some of the applications programmers and the more technically-oriented systems programmers. A programmer who started at the bank 14 months ago when she was fresh out of college said that sometimes the systems programmers "talk over your head." Still, she found them helpful, and said that the communication problems decreased as they talked to each other more frequently.

Although systems programmers are remote and sometimes difficult to communicate with, applications programmers spoke of them in respectful tones. They are respected for their technical expertise, and for the higher salaries their scarce skills allow them to command. When applications programmers compare themselves to the outside software community, they say that the systems programmers are most
like the technical "geniuses" in the book, *Soul of a New Machine.*

In addition to communicating with systems programmers when they have problems with their "tools," applications programmers work with systems programmers on particular phases of projects. On a large project one or more systems programmers may be brought in for a few months. Said an experienced applications programmer,

...and now you're bringing in a systems programmer who, if not in title, at least in terms of income is at least the equivalent of the project manager. In general the systems programmer would not care what they're called or who they say they're reporting to— you still have to treat them with a certain amount of deference. You're not going to push them around by any means."

There are also people regarded as technical experts who work within the applications programming area. According to one manager, large projects really only need one or two

---

1. Tracy Kidder, *Boston: Atlantic-Little, Brown,* 1981. This book, which describes an exciting and high-pressure development project at Data General Corporation, was mentioned by two interviewees. They used the book to exemplify a sort of state-of-the-art development environment not characteristic of systems work at the bank. Even the systems programmers, who of all the systems people are closest to that ideal, have more technically mundane jobs than those described in the book. The head of an information center at the bank said of the systems programmers, "But they're not innovators, whereas the people in *Soul of a New Machine* are creative geniuses. They are thinking of new ways to do things better. The people here are constrained by our environment, and we work well within that framework...Although we do have to be creative and be able to change and innovate within the environment." Although she may underestimate the constraints on programmers even at state-of-the-art development sites, it is clear that she feels that the limitations imposed by the requirements of serving the banking community on a daily basis prevent certain sorts of freewheeling creativity.
technical "gurus" in order to function smoothly. These one or two, however, are essential, and it was the manager's perception that these experts are the most difficult for the bank to recruit and hold. Furthermore, he said, these experts generally don't want to act as managers—and thus they generally serve as "individual contributors" who have a relatively high status but have no one reporting to them.

**Ad hoc reporting**

*Ad hoc* reporting is a task which combines the tasks that make up the systems development lifecycle. It is mentioned separately here because its goals, its context, and its cycle are often quite different from those of the traditional systems development project.

*Ad hoc* reporting, as its name suggests, is an activity whose goal is the generation of data to be used for a specific one-time purpose. At the bank, there are hundreds of reports generated from large transaction-based systems on a regular schedule: daily, weekly, monthly, quarterly, and so forth. By contrast, *ad hoc* reports are created to inform decisions which are either unique or rare.

This sort of reporting is most often practiced in information centers, but in principle it can take place anywhere. The reporting effort, described above, for the strategic planning project was *ad hoc* reporting on a large scale. Systems people may execute the report, or a "user" may produce a report using a PC (and even on rare occasions the mainframe computer). Frequently a fourth-generation
language or database program will be used. Because the software and hardware are generally less unwieldy than COBOL on a mainframe, there is more possibility for multiple iterations. This in turn allows for a more flexible development process than the traditional third-generation-language-with-mainframe sequence.

"Interpretation"

Interpretation, or mediation, consists of acting as a go-between or translator between a user or users and some part of the systems community. Interpretation is necessary because of the very different languages spoken in the two communities, and because the two groups often have very different ways of approaching problems.

Often, the interpretation or translation function is performed by systems people or users themselves, as they conduct business with each other. The more interaction the two groups have, the more possible it is for group members to do their own interpreting. There are, however, some individuals and groups who have come to specialize in interpretation. Users who become user-representatives to systems development projects often end up acting as interpreters or mediators. Members of the CBDG, who are mostly nonprogramming analysts, may spend much of their time doing mediation and translation work. According to one member of the CBDG, they consider themselves "interpreters." At meetings, "our heads go back and forth between the end users and the programmers" as if they were at a tennis match, she said.
"Technical" work

Systems people distinguish "technical" work from the work that involves communication with others and the ability to understand business problems. Technical work seems to include writing computer code, but it refers to other activities as well: Designing the system as a whole, deciding on optimum machine usage, etc. Technical specifications are very different from business specifications, said a member of the Advanced Technology Group who had been an applications programmer for more than a dozen years, and "Very many people who start out in coding head in that [business] direction. They never would code again and it wouldn't be long before they weren't able to. But if you're involved in the technical side and how the whole thing fits together, you still are at least in touch with it and if called upon, as I was [during the CBS project], you'd at least be able to do it."

Skills

After seeing the range of tasks described above, it should come as no surprise that technical and analytic skills are only some of the abilities required for effective work in the systems area of the bank. When asked to describe the qualifications of a programmer trainee, a manager in the corporate systems group said that trainees should have "good analytical ability, good communications skills, and the personality to work as part of a team."
Contrary to the stereotypcial notion of computer programming work, only one of the three abilities he mentioned could be considered "technical." Trainees are given a short training course in coding, operating systems, and so forth, and they learn much more on the job, but in terms of their underlying characteristics, the so-called "people skills" receive at least as much emphasis. This is reflected also in a remark by the head of the Advanced Technology Group, who joked that "I wrote terrible code, so they promoted me."

A member of the ATG argued that systems people develop a set of skills that are recognizably their own but that are also very useful on the business side of the bank. "In a systems job you get big intellectual muscles in organizing, segmenting, designing, implementing. All of these skills are very much applicable to the business side." 3

Division of Labor

Systems work is divided among the pool of systems people in a variety of ways. First, the systems group is itself a subset of all bank employees, being divided as it is from the "business side" of the bank. Within systems,

---

3 There is a literature on the psychological aspects—and, by extension, the skill requirements—of computer programming. For the purposes of this study, I treat skill in its psychological sense very briefly. The issue of skill is surely an important one, and any study which claims to test Braverman’s hypotheses about the transformation of work must come to terms with the issue of skill. I would argue, however, that the sense in which Braverman speaks of skill is not the sort of skill that the psychological literature is investigating, and that therefore the conventional psychological approach can be sidestepped, as I have chosen to do here.
there are units paired with the various groups and departments of the bank, so that, in effect, each bank function or product team has a corresponding team of systems people paired with it. In this way a shadow replica of the entire systems side of the bank is created within systems. Before decentralization, systems people say that their organization was unified, and that people moved more freely between different business functions than they expect to do now. In its present form, the systems organization is broken into several pieces, each reporting to a group head. This is the major division of labor within the systems organization.

There is also, as in most organizations, a hierarchical division of labor. Higher level managers tend, as discussed above, to look at issues with broader scope than those below them are able to see, since the people who are "at the front lines" are often focused on knowing intensively about a particular set of concerns. Also, entry level people tend to do the work that some consider less glamorous and desirable, like writing code and doing maintenance work. To the extent that such work is broken off and given to a particular set of people, it is given to entry level people. Unlike some organizations, however, where the more routine and less desirable work is given to a "permanent underclass" of detail workers, to the extent that there are detail workers in systems they are simply the new recruits. Most of these new recruits seem to move up through the ranks,
generally leaving behind all coding (unless they follow a technical path) within three years. They use their time as programmers to learn skills which they seem to draw upon in future positions. According to one manager, "coding and maintenance tasks provide the experience needed to give a solid technical foundation to build upon. The people who are good [at these tasks] usually become the best of the analysts and designers later." In this sense, the entry-level programmers are rather like apprentices in a bureaucratic setting: they are given what is considered to be the more routine or inconsequential work, but they are also being groomed to move up the ladder.

Even low-level systems workers seem to feel engaged in the mission of the entire systems organization. Even when they receive only a piece of a system to work on, they do not seem to feel alienated from the system as a whole. A lower-level programmer describes a project she was involved with:

My team leader did some of the assembler code and edit routines involved, and I did a piece on a COBOL program that does the dispersal of the transactions to the different applications, and somebody else did some other part. There I had a really myopic view of what was going on, and the big picture isn't that important as long as you know what your little piece is and you can cover that base...[but] you're always coming back to the center...It's not necessary for you to know what the next guy is doing. Which doesn't mean that you don't know. It means that you are given your part to do, and it is woven in with the next guy, and in the final analysis and in the final installation, yes, everybody does come together and fit their pieces together in terms of testing and verification of data and verification of logic. But it's not like I say to my project leader, "It's done" and he takes it and runs with
it. We’re all involved in the end piece...In the testing, the braid starts to get woven, and everybody who had a piece gets involved in the chain.

Within project teams, there are divisions of tasks based on individual abilities and preferences. The division of work is decided by the project leader, or in some cases by collective discussion among team members. Over time, a team member may come to specialize in a certain function and other members of the team may develop complementary abilities. Said one programmer,

There is a man on my team who is very bits-and-bytes-oriented and makes it a point to know the inner workings of things, but that is his preference. That is not what I do. He doesn’t like to deal with our users. He doesn’t like to write memos. Doesn’t like to write spec’s. I do that kind of stuff. I enjoy doing it. So we all fill different needs.

The cyclical nature of projects also brings about what is in effect a sequential division of labor. Because projects proceed in phases, there are times when there is more coding and less analysis, or vice versa. A given individual, then, may be focusing on analysis because that is what the project requires at the time. This does not, however, make this person a true detail worker (if anyone doing so broad a task as analysis could be a detail worker), since their focus on one activity to the exclusion of others is a function of the stage the whole project is in, and the content of their work will change as the project progresses. Furthermore, most programmers are working on more than one
project at a time, and there is variety in passing from one project to another in the course of a day or a week.

Systems people seemed to believe strongly in the importance of seeing and understanding the full scope of a project, even when one is working directly on the creation of only one piece. Typical were the comments of one systems man, now in the ATG, who had 15 years of applications programming experience:

If you can let somebody see the [whole] project; ...if they can follow something [quickly] all the way through; it can be a lot more gratifying than getting involved in a two or three year project. Particularly when you're first starting out. From the point of view of the kid that's just out of school, to be at the lowest level on one of those projects, even a development project with new technology--because you're at the low end of it, you can spend two or three years when you're just coding program after program. If the project is successful and on time and high priority you're being well rewarded for that financially and getting the pats and making a name for yourself, but still...

The same man, however, was convinced that a division of labor was essential for large systems. Before he started in systems he was working as a nonprogramming analyst in the Factoring department. He would have meetings with programmers in which he felt hampered by not knowing programming. This led to his decision to learn programming and move into systems.

I was naive. I thought that if I could program, I could do the whole thing. In reality, there aren't enough hours in the day to serve both functions. You can't go into the user area, understand what they're doing, think about how you could do it better, and then go into the process of how you're
going to do that technically, write the programs, test the programs, implement the system. Five years would have gone by. It’s a lot better to divide it up between individuals, except on some very small systems.

A member of MIS said with relief that MIS was reorganizing to narrow somewhat the range of bank functions that team members were expected to understand. Being a "generalist across the whole bank," she said, is "too big a job." Even the narrower jobs would still be very broad, she asserted.

One manager believed that the division of tasks had some advantages which showed up in the quality of the product. Although he advocated the use of small and one-person projects, he also believed that errors were more likely to be found if different people did the coding and the analysis. "If they miss something in the analysis, they’ll also miss it in the coding," he said.

Separation of coding and analysis

At the bank, interviewees described a significant level of integration of the tasks of coding and analysis.

One sometimes reads [said a manager in the Corporate Systems area] about the possibility that people who create code will become even further removed from the real problem-solving. They’ll just be handed specifications and be expected to churn out code as a sort of second-class citizen. But I don’t think that that’s really a process that’s occurring. I think the process that’s occurring is that there is less and less work that can be described as coding as a proportion of the total amount of work that we do at the bank.

One of the striking features of the systems area at the bank was that there was general agreement among systems
people that there was no one who was exclusively a coder. Everyone who wrote code—with the exception of temporary contract employees—also did analysis at one time or another. For a variety of reasons, enumerated below, they believed that the wholesale separation of coding and analysis, with the creation of "pure coders," as one manager called them, was untenable. "In a financial institution, you can't write code all your life" was a typical comment.

Many reasons were given for the fact that there were no people solely occupied in coding. These reasons, along with some elaboration, are given below:

1. Shrinking number of new development projects

In order to permanently separate coders from analysts, according to systems people, all projects must be of a certain minimum size. On a small project, the separation of coding and analysis becomes unwieldy and ludicrous. Only large projects offered the possibility of such separation.

If it was a strictly development environment, I could see [having pure programmers]. Give someone very detailed specifications and they could code it and that would be no problem. But in the support environment, a problem comes up, and if you assign someone who is a pure programmer to handle it, they would not know the implications of the problem or the solution. They would not be able to query the user to see if this was really the solution, if this was really the problem. So if the work really was big enough that you needed two people working on it, then I could see room. But typically support involves assignments, so I would consider it more valuable to have programmer/analysts.

Today there are fewer large development projects, and there is more maintenance. Maintenance ("production
support") takes place in small projects performed by individuals and small groups, and therefore does not provide an opportunity for the separation of coders and analysts. A systems employee who had spent two years in systems at the Coca Cola company contrasted that company's organization of systems work with that of the bank. At Coke, there was much more new development taking place.

Production did not have much prestige...Their company was not dependent on their systems, and so production support was not as crucial. So you had a very few people doing production support and a great deal of people doing new systems development. Therefore you did need a team of programmers who could come in during the implementation stage of a project, once the requirements had already been established, and do programming. Here, where the production is crucial because this company is running on a system where it's strictly dependent on its systems, there's a lot more production...and a lot less development, because you don't just do a new DDA system. There is some development--I don't really know how much--and to the extent that there is, you could have pure programmers.

This same systems person had also worked at the accounting firm of Arthur Andersen and Company, and argued that the separation of coding and analysis were also more feasible there.

In a consulting environment like AA, people are assigned to a specific project for a specific length of time. You can structure your team however you want. Everything is very fixed. Whereas in this environment people are assigned to support a particular type of production environment forever, so you don't have the same fixed period. You have crises come up, you have things happen. You have to respond. So I think it takes somebody who is multifaceted to be able to handle problems, to be able to come up with solutions. If you had an environment where you just had an analyst, you just had a programmer, and you were trying to provide that sort of support, then the programmer would always be lacking the knowledge to understand how a particular problem happened or where a solution might best fit in, and an analyst might have trouble implementing the solution. I think there is room for pure analysts because as long as you understand
the problem and understand the solution, you can have somebody else do the implementation, but I don’t think there’s much room for pure programmers, because you sort of live on an island when you don’t really understand the business aspects of it.

2. Having programmer/analysts is more efficient

One manager suggested that a software vendor who runs large development projects and keeps a stable of programmers might be able to produce software more inexpensively because the vendor could hire low-level people to do nothing but coding, while the bank spent more paying programmer/analysts to write code. He believed, however, that the advantages of merging the two tasks in one person more than outweighed this disadvantage. Another manager expressed a similar sentiment, saying

I suppose that a person who coded all the time might be a little more efficient at it. Somebody had to put down the word processing tools and pick up the COBOL again at a certain stage in a project, but I think that’s more than offset by the fact that you don’t have the need to communicate at that transition stage as much. You have some of the same minds doing the coding that were doing the analysis.

The head of an information center believed that a separation of programmer and analyst would seriously decrease the quality of reporting work.

If I have to translate everything into specifications for a programmer, then I’ve really lost something, because the person there wouldn’t be able to look at the output and say, that’s stupid. You should have thought about this, or, now you’re going to have to run it again, because this data tells you nothing. Even sometimes after I’ve thought through the whole process, I run the report, I look at the report, and I say, this cannot possibly help you make this decision.
This level of integration of work was made possible by the fact that a fourth-generation programming language was being used. Such languages dramatically simplify and expedite the coding process in situations for which they are appropriate. Even for programmers in a systems environment more conventional than the information center, however, there is considerable variation in the work. To have separate analysts and programmers would constrain managers' ability to allocate systems work. Said one programmer/analyst:

Sometimes [my project leader's] information is real specific. Sometimes he'll come over to me and say, "See right here in this program? This code needs to be changed to do this, this, and this." And that's a very small thing. And in other cases he'll say, "Look at [that program] and find out this," and in that case it requires more analysis of the source because it's not a program that I know start to finish. I know different sections of it, but it's a 10,000, 15,000 line program and I don't know the whole thing. I just know what I need to know and what that affects.

3. Coding and analysis are not really separable

An additional reason why coding and analysis are not separated is that, as discussed above, they are difficult—perhaps impossible—to separate in practice. This was reinforced by the description given by a programmer/analyst.

I think even in the most minor changes you need to identify other areas of a program that may be affected, and I would call that analysis. Maybe other people would call that programming. I guess I'm using a characature of a programming shop where someone comes over to you and says, "Add these lines of code here." That doesn't happen an awful lot here...You're making this change and it looks like you're only affecting this one area of code, when 5,000 lines down the code you're changing the whole
report because of this variable that you’re going to add.

This unity of programming and analysis is one of the factors making the wholesale separation of coding and analysis not only difficult, but also counterproductive in many cases.

4. Improves bank’s ability to recruit and retain

Finally, managers in systems acknowledged that one of the reasons why they did not create a caste of "pure programmers" was that it was easier to attract and keep bright, capable people in systems if coding were not allowed to become an occupational ghetto. According to one manager,

I don’t think there are many good people that we could easily recruit into a programming role if that’s what they saw as their entire future...People who want to develop [writing] software as a career, and who are just interested in that process for its own sake, do not tend to come to a bank.

The people who do tend to come to a bank are those who are at least halfway interested in learning about banking, and those who have little enough experience with computers that they are not candidates for sophisticated software jobs at technology-based firms. Such people tend to feel, in the words of one manager: "I don’t want to be an analyst all my life. I don’t want to be a coder all my life. I like that variety of being able to analyze, develop, and code."

Some long-tenured systems people reported that there had been a time when programmers and analysts were substantially separated, although they dated the
disappearance of pure coders differently. One who had been 15 years at the bank said that he believed there had once been pure coders, but they were before his time. Another manager who came to the bank in the same year remembered an era of pure coders, but said that programmers had officially started doing analysis as well in the early 1970s. A third manager, who had been around for 18 years, most of them in the systems area, also recalled having systems staff who were strictly coders.

While there are no pure coders at the bank today, there are many analysts who do not write code. Many of them, probably the vast majority, were once programmers and programmer/analysts. As they have been promoted they have ceased to code but continued to do analysis. Others, like the analyst from CBDG, have never coded.

We are solely analysts. We don’t know a lot about programming. I always understood that you were a programmer and then you became an analyst, but we got to it through knowing the industry [that is, banking]. I’d like to learn programming. It would make my job a lot easier.1

Nonprogramming analysts expressed a strong interest in learning programming. The comments of a manager who has spent his entire working life at the bank are typical. He described why, after a short time in Factoring as a

---

1 At a followup interview a year later, this analyst had just completed the training course for programmers. She was very enthusiastic about her newly-acquired knowledge, and said that she finally understood the criticisms that the programmers had been making about the specification she wrote.
nonprogramming analyst, he decided to join the systems side of the bank.

The programmers, the technical people, would tell me that I had the space of 300 bytes to store a piece of information, and I'd worked it all out, each character being one byte, and I had four bytes left over, and they'd tell me, "Oh, yeah, you can put a seven-digit date in there." I'd ask, "Well, how can you get seven digits in four bytes?" and they'd go into an explanation. But from that I thought, how can I do my job unless I know these kinds of things? Why should there have to be the interpretation taking place? Also,...[I had] a fascination with the machine...

By contrast, some analysts who had been promoted from positions in which they had had to write code showed no nostalgia for coding. They all, however, had coding experience and continued to know the rudiments of coding--and to use this knowledge in their work--long after they had stopped writing code.

**Shift over time from coding to analysis**

Systems people agreed that one of the most striking changes over the last decade and a half is a distinct shift in the relative emphasis given to coding and analysis. Systems people are spending a far greater proportion of their time and attention on "business issues" and a smaller proportion on "technical tools," according to a senior manager in the systems area.

There has been a general shift towards the notion that you ought to do everything you can to get the definition of the systems right, concentrate more time on that, do some small development to give feedback to the users, do your design work very carefully and conscientiously and make sure it's coherent and robust and that everything is as it should be. There's less emphasis on the programming phase itself, which is now considered to be,
relatively speaking, short and unchallenging and unimportant. It isn’t just that our coding has improved over the years, it’s just the realization that if you’ve got the systems properly designed, first of all, properly designed to meet the real user needs and secondly properly put together from a structural, technical point of view, then the coding comes very easily, and the testing also is much less time-consuming.

Another senior manager agreed that this change of emphasis was taking place, and supported it very strongly, saying, "[Systems problems] tend 99% of the time to be organizational or political. Very few of the major systems disasters are ever technical. They’re usually people-related: systems not built for the right person, not the right level of user involved, that sort of thing."

The declining emphasis on coding as a proportion of all systems work is not solely the result of an increased emphasis on analysis. Software and hardware advances have helped to reduce the proportion of time spent on the coding aspects of systems development projects. Fourth-generation languages like SAS, Focus and database languages, coupled with hardware and software changes which provide online programming, better machine response times, and faster and more available facilities for testing, have reduced the duration of coding and testing activities. Increasing outside purchase of software dramatically reduces the amount of coding but does not reduce the need for analysis. Further reductions in coding could be achieved, according to one project manager, if the bank were to make more use of
software and hardware already developed within the bank and elsewhere, which he said was underused by most of the bank’s systems people. As these new technologies are exploited, their chief effect should be to reduce coding even further compared to analysis.

Technical work is more amenable to automation. It’s more amenable to being described as a set of procedures that can be programmed. I don’t think we’re at the point yet to foresee how that might happen with requirements generation, for example. I think that’s still too much a unique set of activities in each case. Or even if it isn’t unique, it requires an interaction with the business people that is not easily automated or reduced—in fact, we’d like to increase it.

Thus, for a given development project, the absolute amount of coding can be reduced by providing tools that expedite aspects of the coding and testing process. One high-level manager said that she believed that, through a combination of improved tools and better communication with users, productivity had been very substantially increased in the systems area.

I...see so much more productivity. It’s so hard because we don’t measure productivity very well in our business. Lines of code doesn’t even [come close.] Function points are so darned hard to measure. But if I just look at the numbers of requests for services that we’re pumping out and what types of major changes we make and how quickly we do it, I just think we’ve become so much more productive. Adding a management audit report to the DDA [Demand Deposit Accounting] system was a standard 140 hour task. Now we do them in a day.

One senior manager was so struck by the change in job content for programmers at the bank that he said, "Programmers aren’t programmers anymore...Programmers died eleven years ago and people haven’t understood this." The
job of programmer was so changed, he said, that he speculated that 40% of the bank’s programming staff had never seen a computer (as opposed to a computer terminal, which all programmers use). "Today," he said, "the computer is more of an abstraction." While he did not find fault with the change in focus of systems people, he believed that titles should be altered to reflect the shift. Programmers dealt so much less with technical issues and so much more with business issues, he said, that perhaps they should be called "Business Analyst" or "Manager’s Assistant" or "Information Specialist" instead.

Managers in systems foresaw a continued decline in the proportion of time systems people spent on coding and testing, but a distinct and continuing role for "technical" work alongside the work of analysis.

I think we’re moving to a time when we’ll have fewer and fewer [programmer/analysts], and we’ll have a lot more business analysts who happen to have tools called SAS, Focus, whatever, to help service their end users, and on the other hand we’ll have highly technical folks who will be servicing the databases, tuning the online systems, understanding the interfaces between online monitors and database management systems and VTAMs and NCPS and MVS’s and all the operating systems: highly technical folks who will understand assembler language and be down in the bits and the bytes and really know how it all hangs together, and then there will be the business analysts, and I do think that COBOL programmers will be less.

Another senior manager echoed this, in a statement already quoted in part above:

[T]here is less and less work that can be described as coding as a proportion of the total amount of
work that we do at the bank, and I would expect that within a few years that will be further diminished by our move away from procedural languages and towards things that require much less time in the actual coding...And the technical end will continue even as the coding withers and dies away because there will still be high-level technical work in areas such as database design, system design, and that type of thing.

A woman from MISD made the same prediction, saying that less applications coding would be needed in the future, and the large groups remaining would be the "real" systems technical people (the systems programmers who design and maintain the hardware and software) and the "people in the middle" who do analysis and teaching.

Machines and Languages

Taken together, the hardware and software available at the bank constitute the "technology" of systems work. From the point of view of systems people, this technology has a dual role: it is the basic material they work with as they manipulate computerized instructions and configure computer hardware, but it also provides the tools systems people work with in the performance of their assigned tasks.

For the most part, the bank is an "IBM shop," using IBM and IBM-compatible hardware, and running software designed to work in an IBM environment. The IBM equipment handles the bank's data processing needs, while most word processing and some other office automation tasks are done on Wang equipment. Recently, as the Wang machines have become more powerful and more versatile, Wang users (who are mostly clerical and professional staff) have begun to request
spreadsheet software and other data processing tools, and some convergence of office automation and data processing has taken place.

The bank’s large mainframe computers are kept at a location remote from the bank’s downtown headquarters and the satellite office building adjacent to it. The "production" computers are operated out of the Columbia Park computer center in a low-rise commercial and light industrial area inside the city, while the computers used for end user computing, ad hoc reporting, and development work are located in a western suburb. The suburban facility provides computer time for programmers who need to test programs during development, and for systems people and others who need to use the data extracted from production computers on a periodic basis. This keeps the production computers from being slowed down by end user and reporting programs, which often make heavy demands on hardware resources. When working on the large, transaction-based production computers, systems people use the urban facility. Also, when writing a report that requires information not held in the extract files in the suburbs--information such as daily balances for accounts--the urban data center is the source programmers use.

A variety of programming languages and database systems are in use at the bank. Of these languages, COBOL stands out as by far the most heavily used. Over the course of the history of computers at the bank, COBOL has been the
language of choice, as it has been at most large business installations. Only in the very recent past has COBOL suffered significant challenge, but even today it is the language every systems person at the bank must learn. Although it is likely that the bank will continue gradually to reduce its reliance on COBOL, there is still new development work taking place in COBOL, as well as much maintenance and enhancement work still to be done on existing COBOL systems.

The first language written primarily for business applications, COBOL has survived for a quarter of a century in a field where quick obsolescence is the rule. COBOL is ponderously slow to write compared with most other languages now available, and takes many lines of code to do what some other languages and database software can do far more briefly. Still, COBOL remains because of its versatility, because the bank already has an enormous investment in existing COBOL systems and in the COBOL coding skills of its employees, and because COBOL programmers are the easiest sort of programmers to find in the local labor market. (For an example of a COBOL program, see Appendix 2.)

A senior manager at the bank says that he doubts that any future systems will be "coded line for line in COBOL. That's pretty much a thing of the past now. We look for more efficient ways of doing things." COBOL is appealing, he said, because it's familiar and you know you can take care of it, but in the long run it makes no sense. He
characterized it as a security blanket for "those who object to strange products."

The programmers interviewed for this study reported no great affinity for COBOL. Some analysts who no longer wrote COBOL code expressed an active dislike of the language. Fourth-generation languages were interesting to many, but many programmer/analysts who were enthusiastic about these higher level languages in principle found them to have shortcomings in practice. Still, some strongly endorsed the use of fourth-generation and database tools, as the head of an information center did:

I'm a big proponent of using higher level tools to do things that are ad hoc that don't require being run every night. But even if they do, since people time is much more expensive than machine time I say let's use them anyway. But I would listen to reason for applications where efficiency is a concern...I love [SAS and Focus] both dearly, especially SAS, because I have a management science background, so I enjoy both doing the data management in SAS and using a lot of the statistical procedures.

Changes in software and hardware, managers argued, were a "prerequisite" to the development of certain products or services, but the existence of a technology which would permit a particular approach did not mean that such an approach would necessarily be taken. Several high-level managers mentioned that the bank would like to move in the direction of having some of its corporate customers take bank computer terminals into their own offices and do some of their own data entry on things like letters of credit. This would save the bank a lot of money by allowing
management to reduce the clerical staff. This approach was not yet possible, however, because of problems with data security which had not yet been solved. Conversely, a senior manager suggested that all the technology was in place which would allow the bank's service representatives to work out of their homes. Yet even though this was technically feasible, the bank had not moved to encourage home work.

Although the absence of a technological solution can prevent the bank from pursuing a particular avenue, a senior manager felt that the biggest changes in the systems area were independent of technology.

If we didn't have on-line systems, then putting in a system in a customer's office would not really be a feasible approach. I don't think the technical advances are particularly essential for the overall transition toward strategic types of systems. If we were still totally in a batch mode I think a lot of that would [still] be going on.

There are some pieces of hardware and software which do not become part of working systems but are reserved for use as tools by programmers. The terminals used by programmers as they write code are the main technological assist they receive. The compiler, in addition to translating the "source code" written by the programmer into "object code" used by the machine, checks programs for syntactic and structural flaws and reports them to the coder. Higher-level languages, of course, can significantly speed the coding process.
There are still programmers at the bank who do not have their own terminals, and must write code at a terminal in a common area. A programmer/analyst and a manager both mentioned that having ones own terminal speeded the process of software development. The programmer/analyst, who had been at the bank just over a year, had recently gotten her own terminal and said it improved her concentration to have it, but added that she was glad to have learned to code at a terminal in a common area, because there were people closer at hand when she had a question.

A programmer/analyst who had recently been assigned to study productivity in his area said that one of his chief aims was to improve the response time of the system, which in some cases was as much as eight seconds. This meant that programmers sometimes had to wait as much as eight seconds after doing one operation before they could proceed to the next. When operations are small and frequent, this slow response time can be enormously frustrating and detrimental to work momentum. "We've concentrated for years on the productivity of user departments, improving their productivity through systems, but we have neglected our own area. My goal is a productivity increase of at least one-third with no increase in staff," he said.

A manager in another part of systems agreed that the systems area was lagging in its level of automation. "Fifteen years ago, we might have been automating things, but our work was still entirely manual. We were still using
[punched] cards. Five years ago we were more automated ourselves [that is, they had on-line systems] and that's getting more true, although we're still pretty much like the cobbler's children." In some cases, tools have been developed but have not made it into general use. One systems person described two pieces of software that had been developed as programmer's tools by two bank employees. One generated automatically the "skeleton" common to all COBOL programs, and the other was a tool to introduce structure into unstructured COBOL code. He asserted that many of the people who might otherwise be using these tools didn't know they existed. The two developers of the programs have since left and started their own software company.

Data and Data Problems

In addition to writing and maintaining computer programs aimed at particular banking problems, the systems area establishes and oversees large databases with data about various aspects of the bank's activities. Additions to and deletions from these databases may be made by systems people or by members of user departments, depending on the nature of the database and whether the system it is paired with is past the startup stage or not, but the technical design and control of the database are systems responsibilities.

When building a new system, it can be as difficult or more difficult to get the right data properly organized and into the system as it is to write the software which will
manipulate the data. With systems such as the Performance Reporting System, built by MISD, getting the data into the new system was difficult because it required bankwide cooperation from departments which had other demands on their time. Some departments had to recast the data they kept in order to put it into PRS format. The data also had to reconcile to the bank's general ledger, and this did not always happen. In August, 1985, eight months after the PRS had first been put into operation, a member of MISD reported that people were still having to "fudge the numbers to get them to come out."2

Software Vendors

There was evidence in reports by several managers in systems that the bank was moving more toward the purchasing of software from outside vendors. This can be done in two ways: first, through the purchase of software "packages" which are self-contained products that may be used just as they come "off the shelf," or modified to suit a particular application; or second, by hiring a software development firm to write a piece of software to fill a particular need.

To the extent that PCs have become a significant presence at the bank the amount of purchased software increases, since it is rare for PCs at the bank to use anything other than purchased packages. But there are now

2 Another systems person familiar with this effort said that the numbers weren't being falsified, as this quotation implies, but that adjustments were needed as data were drawn from non-automated systems and introduced into the PRS.
whole packaged systems available for minicomputer and
mainframe applications, too, and the bank is making use of
these.

There are also some cases where outside vendors are
engaged to write custom software. On some of the largest
development projects this is especially likely to be true.
The bank's staff may collaborate with the vendor's staff to
produce the finished system, with the bank's systems people
playing a particularly important role in supplying the
specifications and the firm-specific information.

The outside purchase of software significantly reduces
the amount of coding that needs to be done within the bank
to install a working system (one manager called purchased
software "the last nail in the coffin of the programmer
role"). In many cases, coding is not eliminated entirely
because programs must be modified, or because other pieces
which connect to the purchased package must be designed and
built. Outside purchase of software does not, however,
reduce the need for analysis, since the analysis forms the
basis for any intelligent decision about what software
should be purchased. Still, the outside purchase of
software substitutes some labor from sources external to the
bank for labor that would have been performed internally if
the bank had decided to build such a system itself.3

---

3 There must certainly be cases in which something that
is purchased as a package would not have otherwise been
built internally, because of the expense or the lack of
know-how. In these cases outside purchasing is not a
substitute for labor from sources internal to the bank,
but rather is a supplement to it.
A senior manager in systems, although admitting that there was controversy on this point, argued that the bank should buy software from outside where there are appropriate products on the market, especially for the basic transaction processing and accounting functions that tend to be similar [to those of other banks] because they have to do with the basic nature of the business. Creating those applications is very time-consuming and not necessarily very interesting, and therefore to the extent that we can buy those parts of that software portfolio we get away to some extent from routine work, uninteresting work. I think we’ll be doing more buying in the future than we have in the past. The best place to use our own programming resources is in areas where we want to do something different and better than the competition, and those are precisely the areas that tend to be most interesting and least routine, for obvious reasons, and get us interacting in the most intense way with the business people.

Reports from the bank suggest that in fact the ratio of bought software to built software is increasing, and that the systems area of the bank is increasingly concentrating on using the skills of its members to focus on the bank and its business problems, rather than on the technical details of software designed to solve generic banking problems.
CHAPTER FOUR

AT THE BANK: PART TWO

Chapter Three described the nature of work at the bank, as well as the some aspects of the technology as they relate to the work content of applications programmers. In this chapter, the labor market for systems workers is discussed; following that, the relationship of systems people to the rest of the bank is examined.

The Systems Labor Market

In August, 1985 a middle manager in the systems area described the job ladder for entry level employees. The progression of job titles for entry-level people was: Programmer Trainee, Systems Analyst/Programmer, Senior Systems Analyst. The next step, Business Systems Specialist, was usually attained about three years after starting work as a trainee. Job titles seemed to be in a state of considerable flux, however. One woman, who had recently been promoted to Systems Specialist, reported that it was her third job title in about six months, but that the changes in title had not been associated with promotion. She added that the job ladder had formerly been split into much smaller increments (for example, there had been a Junior Programmer title, she thought) than it now was.

Her promotion to Systems Specialist had made some changes in her job, she felt.
It's more analysis work than I think the typical programmer gets to do...[Programming was] much more structured, whereas I think I have a little more independence...I'm able to look at systems and say where can we improve this.

As a systems person moves up in the department's hierarchy, job content changes.

The mix of work here depends more on the seniority of the individual than on anything else. Obviously the more junior people do more programming and less systems analysis, requirements definition, and systems design.

A young programmer/analyst said that the bank had a handout on career paths, and that they encouraged people to move up. She contrasted this attitude with that of some other large employers of programmers where, she said, they just make you "program 'till you drop."

A more senior systems man described how he had made himself a specialist in some new technologies as the bank adopted them, and used his technical expertise as a way of winning assignments to new and interesting projects.

For systems people above the entry level, the bank established in 1986 its own version of the dual career path found in some engineering settings. The standard promotion path is known as the "managerial" path, while the non-supervisory career track is the "professional" path. The professional path contains a series of steps parallel to the managerial path, reaching at least to a level equivalent to that of the manager of a large department. Those with a strong technical bent and no interest in acting as a manager will presumably follow the professional path.
Other systems people generally become managers of some sort. "You leave coding [altogether]," said a longtime member of the systems area, "or you're directing [the coding of] others. Helping them to design their programs. Doing program walkthroughs with them, where they go through their code." A manager in MISD said that this sort of promotion path was still the most common and best-understood within the bank. I know how to promote a manager to a higher level, she said, "just give him more people."

(2) Lateral mobility

Sometimes the sort of mobility that systems people reported as attractive was the ability to move from one project to another, even though this movement might not involve a promotion.

You can stay within the bank and maintain all your business connections and your networking that you spend years establishing, but have a fresh outlook, be on a new application with new problems, new issues, new projects... said one systems specialist. A middle manager, however, expressed concern that the splitting up of the systems organization during decentralization would limit lateral mobility. "Before, when you had one systems group that was responsible for the whole bank, if something appealed to you in terms of the application it was addressing, and you were any good, you could get yourself on that project."

Turnover

Turnover seems to have been a serious problem in the systems area of the bank at times. One systems person tells of being hired at the bank one March in the late 1970s. By August, he said, 30 or 40% of the people around him had left, and he had become--inside of five months--the senior person on his applications team. Similarly, a manager
reports that in the late 1970s or early 1980s the job market was really flourishing, and a great many people left the systems area. "At one point, 80% of our staff [in our section] had under three years' experience, and that's a very precarious situation to be in." 1

The high turnover rate, he said, led management at the bank to "take a harder look" at salaries, as well as at career development, "hard core benefits", job challenge, and environmental comforts and the trappings of success (like carpeting and nice desks). As a result, employment conditions have improved for systems people. Turnover is not the acute problem that it once was, although it is not clear whether this is entirely due to internal changes at the bank, or whether there might be contributing factors in the external labor market.

Salaries

Problems with labor turnover, coupled with chronic problems attracting and keeping experienced technical specialists, have led the bank to significantly increase the salaries of systems people. According to one middle manager this happened once in 1978 and possibly once again since then. The salary increase was called a "market adjustment," not a raise, he said, and was explicitly designed to bring

---

1 In reviewing the first draft of this chapter, two managers disputed the claim that the turnover rate was quite this high, but almost everyone agreed that turnover had been--and in some case still was--a very significant problem.
systems salaries more in line with pay for programmers and analysts in the local labor market.2

The bank is located in a metropolitan area where there is a high concentration of employment in the science- and engineering- based industries. As a result, the bank has stiff competition for technical specialists from area companies doing highly technical, state-of-the-art research. Generally, the bank pays less for systems people than do leading edge software and hardware development firms, according to systems people at the bank. This undoubtedly contributes to the difficulties the bank has had attracting and retaining technically-oriented specialists, although it is not clear that the bank would do dramatically better on this score if its salaries were raised only to parity, since for those intrinsically interested in the technology, high-tech employers are likely to offer more interesting work.

One feature of pay in business systems departments, reported by employees both at the bank and at a large local insurance company, is that it can often be extremely profitable to programmers in business application jobs to change companies. Once they have a couple of years of work experience, programmers can get overnight salary increases by moving to another company within their industry. Because

2 Attempts to confirm the existence of such a market adjustment with personnel representatives failed. After three attempts, a member of the personnel area said because of high turnover in the personnel area it would be difficult to find someone who remembered actions taken in the late 1970s. Records of this adjustment may exist but they were not offered by personnel representatives.
experienced people are so sought after, a programmer who starts as a trainee at one company can move to another with relative ease after a relatively short period of time. One man told of a woman formerly at the bank who got a $5,000 raise by changing companies. Such stories are not uncommon.

**Conditions of employment**

Systems people generally work full time, and are subject to working overtime when the need arises. When a project is urgent and overdue, or when a production problem arises with a needed system, systems people generally work overtime. "One of the less interesting parts of the job," said a veteran systems employee now in the ATG, "was phone calls at 3 AM, or at 8 AM Thanksgiving morning." Now when a programmer is subject to being called while off duty, that person may keep a terminal at home, which sometimes makes it possible to diagnose a problem without actually coming in to the office.

Systems people spoke with evident pleasure of the bank's flextime policy, which allows them to set their own arrival and departure time, provided they overlap with their colleagues for a specified block of time in the middle of the workday, and provided that some systems people were present at all times when the "bankers" were at work. Some systems people also praised the bank's maternity leave policy.

For the first six to fifteen months, new systems employees are subject to the same rules as the bank's other
nonexempt employees. Once they become exempt, the rules change. For example, they cease to be paid for overtime work. One programmer said that the only record kept of their work hours was an on-line project tracking system which bills programmers' and analysts' hours to particular projects. Programmers' comments suggested that managerial oversight of their work hours was casual, and one younger programmer was so struck by this fact that she wondered aloud whether anyone would catch someone who took an extra week of vacation.

**Backgrounds of systems people**

Systems people came from a wide variety of backgrounds, most of them not technical or scientific. Here is a sampling of the career paths of several systems people, up to the point at which they entered the systems area at the bank.

* A college business major whose only interest in business was that she thought it would help her get a job. In her freshman year she took a required Fortran programming course and said it was the first thing she had really enjoyed.

* A woman with a master’s degree in English was looking for a way to make a living. She was working at a local college when word processing was introduced, and "got in on the ground floor." After a time as an office systems analyst at another financial corporation, she joined the bank in the office automation area.

* A man who joined the bank 18 years ago as a programmer trainee straight out of college.

* A college German and Economics major with an MBA who went to work as a business analyst at a large insurance company. She decided she wanted to learn programming just as the company moved its
programming operations out of state, so she came to work at the bank as a programmer trainee.

* A woman who worked as a secretary, then as an administrative assistant on the business side of the bank, then became a nonprogramming analyst who worked as a liaison between systems and a particular business area.

* A Vietnam-era veteran who came back from overseas and applied to the bank for a summer job to tide him over while he looked for a permanent job. The bank gave him an aptitude test and suggested he try programming when they found he had a good math aptitude. He said with a laugh that his worst courses in school were computer systems and financial institutions.

* A piano major from a music conservatory who spent six years in the bank's check processing area, first running a check sorter, then settling the tellies coming off the sorting machines, then working as a shift manager.

* A former high school teacher who had taught for seven years before leaving teaching. As a result of cutbacks after a tax-cap measure was passed in his state, he saw the best people leaving education and decided to get out himself. He joined the office systems area of the bank.

* A woman who started at the bank 18 years ago as a programmer trainee with a degree in biochemistry. Early in her career her husband was transferred to another city and she went with him, returning to the bank four years later.

"I've got a strange background," said this last woman, referring to her biochemistry degree, but then she admitted that many people in systems have strange backgrounds. "It makes for a real fun place to work. I think it actually is healthy because we attempt to come at problems from very strange directions depending on our prior lives. It makes lunchtime conversations fun too."

Recruitment
Not all new hires in the systems area are entry-level recruits who begin in the bank’s training program, but many are. When entry-level people are sought by the bank, they are increasingly sought through college recruitment. Many of the systems staff interviewed mentioned participating in college recruitment efforts.

As the discussion of systems people’s backgrounds reveals, most systems employees do not have a computer science degree. While no one suggested that the bank actively avoids hiring computer science majors, one manager said that it is her practice to seek out people with other backgrounds.

When we do our college recruiting my philosophy is: First of all we are a bank, and if I go recruit...at MIT and topnotch colleges, I’m not going to be able to compete against the software firms, the IBM's. So what I tend to do is look for the smartest person I can find with the best record of performance, and I tend to find those in what I call "starvation majors."

She has lots of people on her staff, she says, with PhDs in arts, romance languages, music, and other fields. She describes some of them as "brilliant" and "fantastic," and says "they’ve taken to the world of systems just great." A computer science degree might be helpful for the first year or two, she says, but after that it "all evens out."

Although they have reportedly increased their college recruiting in recent years, entry-level people are the easiest sort for the bank to find. Managers report considerably more difficulty recruiting systems programmers ("systems people prefer to work for a systems house") and
technically-oriented people with considerable work experience in a systems environment. One manager reported that the bank would prefer to supplement its staff through hiring experienced people, partly to avoid the expense of training and partly because he believed that things went better when there were systems people with a variety of systems experiences around.

Bank's programmer training program

The bank's programmers training program is a self-paced course lasting about 10 weeks. According to one woman who had gone through the program a year before, the training program starts with old video cassettes which she found extremely tedious because she had already had some exposure to computers at college. This was followed by more videos on MVS (Multiple Virtual Storage) and an IBM workbook on JCL (Job Control Language). Finally, there was self-paced COBOL training. In addition to the course materials, there were people available to answer trainees' questions.

Trainees may finish the course more quickly if they have prior computer exposure. Today, most new college graduates have had computer exposure in school, but older trainees who are changing careers have not. When trainees are finished with the course they are assigned to a work team and given a "mentor." The trainee's mentor is the person they go to with questions about things encountered on the job. Said one programmer of her early experience with her mentor: "There's always a million questions that can't
be solved in a manual—things like, 'How does the bank handle this?' It's not something you can look up in an IBM manual."

A manager contrasts the bank's present training program with her experience when she came to the bank 18 years ago.

When I came to work at the bank—remember, I had a degree in biochemistry—I didn't even know what a computer was. I was sat down with a bunch of manuals and told that I had to do a conversion of a bank to the correspondent banking system in two months. I mean, training was nonexistent...I got how to code out of a manual, but no one ever taught me to be a programmer, which is the thought process. It was ludicrous. I don't know how I survived...I wasn't alone, [however]. That's what we all did."

**Contract programmers**

The bank uses temporary, or contract, programmers when necessary to supplement permanent systems staff. "We bring them in for projects with particularly pressing deadlines where it's difficult to stretch our staffing...It's a very useful tool because it allows us to run with a leaner, smaller staff," said one manager. Another manager guessed that about 2/3 of the time when they use contractors, it is because they are simply short on bodies. Most projects of any size will encounter this problem at one stage or another. The other 1/3 of the time, contractors are hired because they have a specific skill the bank is short of. Sometimes contractors are hired to fill a slot while a permanent staffer is being trained to perform the necessary duties. One manager said that where possible, however, the bank uses contractors to do the duller, more routine work,
reserving the interesting, skill-building work for permanent employees.

At peak times, contractors make up no more than one-quarter to one-third of the total staff on a given project, according to a senior man in systems. A special case of the bank/contractor relationship arises when an outside software firm is engaged to work jointly with the systems staff to develop a very large system. In these cases, the "contract" workers (that is, the software firm’s employees plus temporary help) may outnumber the permanent bank employees.

A senior manager who oversaw several contract employees said that some of the contractors they’d hired from "bodyshops" (temporary software employment agencies) had been excellent and some had been terrible. The good ones were not interested in becoming permanent bank employees because they "wanted their independence."3

Professionalism and identification with larger systems community

When systems people think of themselves, and when they present themselves to the outside world, they tend to think of themselves as systems people first, and only secondarily as bankers. They appear to be more identified with their

---

3 Verbal reports from a few employers and software placement agents suggest that the skill level of "bodyshop" programmers may be bimodal. The programmers who end up working for temporary agencies are those with few skills who are trying to break into the software business, and those who are so skilled that they have no concerns about employment security and are more interested in being able to set their own work schedules over the course of a year than they are in having an ongoing relationship with one employer.
work and its content than they are with the industry they work in. One programmer/analyst said that if someone asks her what she does she says, "I'm a programmer." She said that if she worked at IBM she would probably say, "I work at IBM." At the bank, however, naming her job describes her better than naming the company she works for.

Systems people may be more willing now to describe themselves as working for a bank than they were in the past. One senior systems man said, "Up until five years ago...I would avoid being described as a banker."4

Although systems people at the bank identify themselves most strongly with their occupation, they are also quick to point out that they are not batting in the same league as software engineers at leading technical companies. Their references to *Soul of a New Machine* were intended to show how much more technically advanced and intellectually

---

4 Sociologist Alvin Gouldner distinguishes between "cosmopolitan" and "local" identifications among particular occupational groups. The extent to which a person identifies with their occupation as against identifying with their employer is an indication of who they see as their reference group. It seems clear that as systems people have less of a sense of themselves as crack coders (thus are less identified with their general skills) and their expertise and pay levels are more tied to what they know about banking and this bank in particular (firm-specific skills) they will become more identified with their employer and less with the software community at large. This process already appears to be taking place. It should be hastened by further de-emphasis of technical skills, since it is these skills that make systems people feel they are a part of and can "make it" in the larger technical world. To the extent that systems people feel inferior to the members of the larger software community, they are likely to turn to identification with the bank for a sense of accomplishment and self-worth.
challenging work at a company like Data General was. As one programmer with about three years' experience at the bank said,

In casual conversation with people who I graduated with who are in social work and nursing, they are awestruck by what I do. They say, "You work with COMPUTERS!" Anything that I may say is to them ladders above their heads, whereas when I talk with my friends who are systems programmers and bits-and-bytes kinds of people, I am awestruck. So it's all relative.

Women at the bank

Compared to other technically-oriented workplaces, the systems department at the bank employs a striking number of women. One manager put the percentage of women in systems as a whole at 51% at the time of decentralization.5 This figure is far higher than the percentage of women in other occupations perceived as "technical."

Women in systems at the bank are not evenly distributed across all positions. Women are said by people in systems

5 If this figure is correct, it means that the bank employs women in systems jobs at exactly the rate prevailing in the banking industry as a whole, according to the 1980 census. Here are the figures for all computer specialists and for computer specialists in the financial sector.


<table>
<thead>
<tr>
<th></th>
<th>Men</th>
<th>Women</th>
<th>% Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>All industries</td>
<td>371,649</td>
<td>142,214</td>
<td>38</td>
</tr>
<tr>
<td>Banking &amp; Credit</td>
<td>15,473</td>
<td>7,924</td>
<td>51</td>
</tr>
<tr>
<td>Insurance</td>
<td>18,718</td>
<td>11,833</td>
<td>63</td>
</tr>
</tbody>
</table>

to be overrepresented in analysis work and underrepresented in the more technical jobs such as systems programming. Further, although one male manager said that there was less discrimination against women in systems than on the business side of the bank, women are overrepresented in the lower ranks in the systems area. It is unclear whether this simply reflects their lower seniority, or whether other factors, such as discrimination or differences in job performance, are the cause.

When asked to speculate about why there were so many women in systems at the bank compared to other technically-oriented workplaces, some systems people suggested that because computer-related occupations were relatively new, there were no gender stereotypes to contend with. This contrasts with banking work per se, which is considered to be highly traditional in its culture and hiring policies.

Programming and systems analysis have also been high demand occupations for some time, which suggests that these occupations might be more accessible to a group which faces employment discrimination in other areas of the labor market. Furthermore, the bank, like many large employers, has an Affirmative Action program, and this may have helped to swell the ranks of women in the bank’s systems area. For these reasons, systems work at the bank may be more open to women than some similar occupations and workplaces.
Black workers at the bank

In marked contrast to the significant representation of women in all but the highest-level jobs in the systems area stood the apparently low representation of black workers in the systems area. The director of Affirmative Action had no figures for the number of blacks or other minority workers in the systems area, but the subjective impression of the researcher based on many hours spent in the systems area was that there were very few black workers in systems. The highest concentration of black workers in systems appeared to be in the lowest-level jobs in the data-processing area at Columbia Park.

The representation of black and other minority workers in systems is thus seemingly out of proportion to their representation in the bank as a whole (18% in 1986) and even out of proportion to their representation among officials and managers and among professionals (7.5% and 11.1% respectively in 1986). The low representation of black workers is surprising because the same factors which make systems work at the bank relatively open to women also have the potential to make such work accessible to black and other minority workers. Relatively high demand, little requirement for advanced training beyond a college degree, and the existence of an internal training program: all these make systems work a very plausible avenue of entry for minority workers. In an urban area with a large population of black and other minority workers, the underrepresentation of blacks in the systems area is a surprising finding.
The Rest of the Bank

The internal structure of the bank had recently undergone a substantial reorganization at the time that this research was conducted. After about three years of extensive planning, a new structure was put into effect on January 1, 1985. Like any significant organizational change, the actual transformation took time, and although the interviews were conducted one to two years after the reorganization, the bank could still be said to be in a transitional state at the time of the interviews.

Before the reorganization, the bank had 11 Executive Vice-Presidents, according to one report. After "decentralization," there were five major groups, each headed by a senior executive, plus a staff "corporate center." The five groups are the New England group, which handles retail business, the Corporate Banking group, which deals with the bank’s corporate clients, the Real Estate group, the International group, handling international branches and business, and the Treasury and Investment group.

At the time of decentralization, the systems organization was split up into six pieces, which were then paired with the five major groups and corporate center. Before decentralization, according to one manager in systems, the systems area was centralized and had a complement of about 250. After the reorganization, the head of systems and about 50 systems people became part of
corporate center, and the remaining systems people were distributed unevenly among the five groups.

In its new form, the systems organization is meant to mirror the structure of the six units with which it is paired. Instead of being directly accountable only to the head of systems, the people in the five systems organizations are responsible ultimately to the business head of the group with which they are paired. Before, the organization of the systems area reflected the structure of the bank's computer systems and the structure of the operations area of the bank. The bank's reorganization attempted, on the recommendation of the consultants who were hired to help with the strategic planning effort, to make the systems areas reflect instead the structure of the user areas they were assigned to serve. A manager in systems describes this change and the changes in the relationship of systems people and business people that accompanied it.

Traditionally, we've been organized by computer application. We've said, because it was convenient to us to put a bunch of programs together into applications, that we're going to make you the user talk to us in our terms, according to our workflows. So many users were serviced by multiple application teams or multiple application teams serviced many users. That's still sort of what happens but we're focusing in according to the business unit mapping. It makes the communication work a little bit better.

To some extent, the variation in the job content and work organization of systems people can be accounted for by the differences in the business organizations they are assigned to serve. For example, some areas of the bank are
very heavily dependent on systems and have been for some time. Others have had relatively little systems exposure. A manager in the Corporate Banking Group Systems Division described some of the variation between groups within the corporate area, saying that some areas (such as cash management, stock transfer, demand deposit accounting, and freight management) are heavily cost-driven, while commercial lending and commercial finance are not. Some business areas--stock transfer and freight management--are particularly "information intensive," while other may be less so. Finally, the "level of penetration" of systems into a particular user area varies, with commercial lending "relatively undeveloped" and the rest of the areas more affected by systems. There are also differences among the five groups, he said, with "the retail banking world...driven more by cost and less by decision support and strategic systems." These differences affect the work of the paired systems organizations because they determine what sort of systems are needed, how crucial systems are to that area of the business, and the degree of user sophistication about systems.

Systems people had varying reactions to the decentralization of the bank and the breakup of the centralized systems organization. A manager in the retail banking area expressed reservations that were typical of those voiced by systems people. She said that she had originally been opposed to the breaking up of the systems
area because it would inhibit the ability of managers to do "peakload management," keeping systems people fully occupied as some projects started up and others ended, and because it could truncate career paths and make the construction of promotion ladders more problematic. More than a year after decentralization, she was more optimistic, feeling that they had managed to maintain the "critical mass" in systems while achieving increased closeness and communication with the business unit. In another area of systems, two analysts were less content with the effects of decentralization, saying that it had separated them from their comrades who were doing similar work, thus interrupting their ability to compare notes on problems. They were, through their own independent efforts, continuing to meet with their former colleagues, but it was not clear whether these meetings would continue indefinitely, since there was no structure to reinforce their continuation.

The systems-business relationship

At the bank, a clash of cultures prevailed in the relationship between the systems people and those on the "business side" of the bank. The gulf separating the two groups, characterized by use of different languages, mutual misunderstanding and sometimes hostility, and difficulty appreciating the worldview of the other group, is
reminiscent of the tension between the "two cultures"
described by C.P. Snow.1

According to several veteran systems people, the
differences between systems and business people, while still
pronounced, are not nearly as acute as they were some
fifteen or twenty years ago. According to one man, who
joined the systems area in the early 1970s,

It was to some extent two warring camps...The
programmers had long hair, dressed casually, lived
with their girlfriends. The bankers were uptown in
their grey flannel suits. In general. It's just
that there were more of us, the same age, together,
making good salaries by bank standards. Maybe it's
also a reflection of the 60s generation, too. Anti-
establishment. Some of us had been in the service.
We were anti-war...If systems went out and hired
someone with technical experience, they'd be
bringing them in at a salary level that may have
been equivalent to a Vice President, and yet this
individual would show up in dungarees. We were all
placed down here [the urban data center] rather than
uptown, so it was a little looser in terms of dress
code, but that was anathema to many bankers.
Particularly the old-line bankers, many of whom are
[now] retired. I'm sure they viewed us with a
jaundiced eye. "Who are these young upstarts who
think they know all the answers?"...I think those
lines really started to erode when they took our
whole group of programmers and moved us uptown in
1976...you were more involved with the user and
became more of a team with the user...We quickly saw
that if we were going to continue to rise at the
bank we certainly better not show up in jeans. But
I think it's probably the first time that the
mainline bankers, the grey flannel suit types, were

1 In 1959, Lord Snow delivered the Rede Lecture,
entitled "The Two Cultures." In it, he asserted that
"a gulf of mutual in comprehension" separated literary
intellectuals on the one hand and scientists on the
other. The two groups differed in temperament,
training, interests, outlook, language, tradition, and
intent. The differences and tensions described by
people at the bank bear a strong resemblance to those
which Snow outlined almost three decades earlier. C.P.
Snow, The Two Cultures and A Second Look; Cambridge,
exposed to a dungaree-clad population. While they had always had the clericals, that was a different caste. Now you had people who could be making as much money as you but didn’t conform to the banker image. That’s pretty much by the boards now. They’ve come to accept us.

Although systems and business people are much more similar now, a young woman who has been in systems only three years observed that there are still differences in dress, although these are decreasing. "Now with decentralization we’re forced a little more to conform to bank standards, but before people would dress a lot more casually and we were definitely a satellite kind of group. We weren’t in the navy blues with the red bow ties and white shirts." With respect to dress, at least, there has clearly been considerable convergence, achieved entirely by the adjustment of systems people to the norms prevailing among bankers.

Some users saw systems people as more interested in technical elegance and intellectual challenge than in solving business problems. Said one user who had participated in a large development project:

A lot of the systems people that I have met are very good at what they do but they don’t understand anything about the way the bank does business. I’m not sure they’re all that interested in really knowing. It’s fine if the project is interesting and it’s fine if it’s intellectually challenging and creative for them, and if it’s not, they’re not quite as interested even if you need the information.

A manager in systems described the change going on in his organization. "I think there’s a greater awareness now
of the purpose of our existence. It is no longer, for example, to come up with neat little coding techniques to perform certain things, but rather to create these neat little things to meet a business objective."

One of the most striking manifestations of the gulf between the worlds of the systems people and the bankers was the difficulty that the two groups had in communicating with each other. Communication problems, and the lack of understanding which underlay them, lead to extensive frustration and much wasted effort and expense.

Fifteen years ago systems did projects and didn’t talk to users. They had specifications tossed over a wall, people did things, tossed systems back, and everybody hated them. That’s an exaggeration, but communication wasn’t the strongest. Five years ago we had areas where communications were really strong, and other areas where it was as bad as it had been fifteen years ago.

This manager, who had started at the bank eighteen years before, has tried to find ways to describe to users what the development process is like and how difficult it can be to change systems after the fact.

...if I can use a lot of analogies with factories and automobile production and the like, they can really understand how expensive it is to retool a production line, and that if after you’ve gone through the retooling process you decide that you don’t like the shape of the car that popped out the other end it’s a total waste. And we do that all the time in systems.

In one extreme case, a programmer complained that while working on a high-level project he had had to rewrite one program more than a half-dozen times. He claimed that each
time he had given the users what they had asked for and they had rejected it.

While the systems people and the users share the bond of working for the same organization, one systems manager said that the bond becomes strained when problems arise.

It's just human nature. You can't help but choosing sides, so to speak. It's always been all for one and one for all, and we all work for the same bank. That's great in theory, but when systems are not developed on time, when systems that are developed don't work, and it comes time to assess why they don't work--were they poorly written, was the technology used just not applied properly, were the functional specifications incorrect--then I think it's very, very difficult to say that we all work for the same place. Now [with decentralization] you truly are working for the same manager and it's a different story.

When relations between systems and business people become strained, said one systems manager, one finds "the business folks saying, 'Oh, those systems guys, don't tell them anything. They're always trying to influence us.' And the systems guys saying, 'Oh, those dumb users don't know what they want.'" A young programmer/analyst said that "fairly often" she heard users saying, "You programmers! What have you done!"

The bankers and the systems people each have a large vocabulary of terms used in their work which are incomprehensible to the uninitiated. Explaining these terms can be a laborious process, making communication with those unfamiliar with the terms difficult and frustrating, and erecting barriers between people who need very badly to talk
to each other, if successful systems are to be built. A non-programming analyst had picked up a book on programming for novices. She said, "Now I know what 'compile' means. It's easy. We have to know these things, because otherwise easy things will sound like a big deal to us."

A veteran member of the systems area recalled the early 1970s: "High priest is a mantle we enjoyed wearing. We loved to use buzzwords to try to impress [the bankers]." He said that today, however, he tried to avoid using buzzwords. People on the business side, he said, would see his use of jargon as "a negative thing."

While the use of jargon can enhance the prestige and apparent knowledge of the user, there are clear costs in quality of communication, effectiveness of systems, and extent of goodwill. These can translate into months of lost effort, and people at the bank seem to agree that communication is better now than in the past, although there is still much room for improvement.2

2 According to James Martin, a leading consultant, in his book on Fourth-Generation Languages, the use of different terms by systems people and users is often a source of serious problems in the systems development process. "The end users fail to comprehend things that are obvious to the computer staff, and vice versa. A user may have signed off on a data field in month-day-year format and cannot be expected to realize that records cannot be sorted into time sequence with this field. A programmer cannot be expected to know that "benefit effective date" is different form "benefit posted date," although this is obvious to the end user. A DP professional may not realize that an oil well has many different definitions or that the oil it accesses spreads underground to areas with different ownership. A user may have signed off on a document referring to "rating basis" without understanding that the systems analyst meant something entirely different by that.
Systems people believed that most users didn't understand what systems people do, and that as a result users sometimes made inappropriate requests. One woman said that users called her with all sorts of systems questions, saying, "You're in systems, why can't you do this?" even when it was not her area of expertise. She acknowledged, however, that there was sometimes confusion within the systems organization itself about who was doing what. Although she was helping users with PC-based applications, she hadn't known about the information center in her area, even though the head of the information center had for some months been working on the same floor as she did.

When users work closely with systems people, especially when they are assigned to systems projects as user representatives, they learn a lot about systems and what systems people do. Said a user representative to the CBS project,

> It was definitely a rude awakening. I was somewhat amazed...I had just never really thought about how any of this worked. It's easy to be on the user side and be critical of what systems people do. I think for the first time I was forced to look at what the users asked them to do, and [the systems people] have no idea of what we do for business, how we do business.

The DP professional might read phrases like the following over and over again without knowing what they mean: "indicates the date on which a given qualification was verified in the context of the structure within which it existed." (This is a real example! Its meaning is obvious to the end users in question.)" (James Martin, p. 209)
Systems people also readily admitted that they had a long way to go in understanding the business, too.

**Accomodations and adjustments**

At the bank, there were both formal and informal mechanisms for adapting to the communications problems and cultural barriers affecting the two groups. The formal adaptations were official organizational responses—assigning mediator roles to particular systems or business people, or giving systems people training in business matters—while the informal adaptations were those individuals used when they needed to bridge the gap.

Users are sometimes assigned on a temporary basis to the systems area to act as official user representatives to systems development projects. This has only started to happen in the last few years, but it seems to be accomplishing the desired result of increasing systems/user communication. The danger is that, as at some companies, some users become "professional user representatives," preferring the excitement of systems work to their old jobs. They stay on and lose touch with the people whom they are representing. There are also some full-time mediators, like the staff of the CBDG, who spend all their time trying to promote communication between the two groups. One member of the CBDG says that when she meets with users, she is accompanied by a systems person, and she acts as interpreter between systems and user personnel. As to other formal measures, one programmer/analyst reported attending a couple
of sessions put on by managers in her area whose subject matter was the business practices of the area they were linked with. Finally, the bank management has some requirements, such as the new requirement that loan officers use the CBS system for certain tasks, that force users to come to grips with systems whether they want to or not.

Informally, people on the systems and the business sides develop a network of people on the other side to whom they can go on an informal basis when they need information. A user with a systems problem may eventually wind up with a helpful systems person on the other end of the telephone, and will keep that systems person's name and number in case of future need. A user and a systems person might find themselves in the same meeting and talk later about other business if the need arises. In this way people in each area meet people in the other, and start to call each other with questions. These questions may not be addressed to the right person, but users and systems people may start with the people they know or the people who have been helpful before. One user said that when she had a problem, a colleague gave her a programmer's name and she called that person. Systems people report having such interruptions frequently, especially if they respond sympathetically and are able to be helpful. Sometimes these interruptions interfere with other important work, and a supervisor may step in to shield a subordinate so that they can get their work done. Most systems people have had the experience of
panicked calls from users, and "you manage" said one programmer.

Sometimes an understanding is reached between systems and business people because one or more people decide to expend extra effort educating the other side. A user representative assigned to the CBS project said, "To me it was a backwoods. It didn’t make sense to have these people sitting up there thinking about what should be in the system without first talking to users. So we spent a great deal of the first six months [of the project] fighting and arguing over what really should be there." A manager in systems described a meeting she had initiated with senior managers from the business side. She laughingly described how one man canceled just before the meeting, and how she had to sit so that another man who kept getting up was wedged into a corner. He would move to get up and she would laughingly say, "I’m not through!" and push him back into his seat.

The convergence of the business and systems areas

Originally, the systems area was an area isolated from the main part of the bank, tucked away in the back office or "operations" part of the bank. Although it was always staffed with relatively well-paid technical specialists, it was not a highly visible part of the bank. Systems have always been crucial to the day-to-day operations of any area of the bank with large recordkeeping requirements, but systems people and systems work have been remote from the professionals in the front office. Slowly, and perhaps more
quickly now with the reorganization of the bank, this has been changing.

The critical importance of systems to most aspects of the bank’s work in the 1980s has been brought into sharp focus by the dramatically heightened competition in a deregulated banking industry. As one senior manager in the systems area put it,

Management on the business side realizes that certain prerequisites to success are made even more pressing by the change in the competitive environment, and a lot of those have to do with being an excellent systems bank, not just an excellent credit bank. Obviously there’s an incredibly heavy impact on the cost side of the organization if you can be a good systems bank, and now that competition is heating up, the cost side can be the critical factor that is going to determine who’s successful and who isn’t...The fact that competition is heating up certainly means that systems and systems people get more attention, and that there’s a greater disposition to spend time and money on being an excellent systems bank.

In addition to the pressures of competition, the demands from members of the bank’s board of directors and from outside regulators for information about the workings of all parts of the bank exert pressure on management to fund and maintain excellent systems. Regulatory pressures have taken a more prominent role since 1985, when the bank’s failure to report some international and domestic transactions made headlines and subjected the bank to disciplinary action.

1) Increased status of systems

Several managers in the systems area felt that the status of systems within the bank had increased very
significantly since the early days, and that
decentralization particularly had made systems a more
integrated part of the main line business at the bank. In
the words of one manager,

When I first came [about 15 years ago], we were
pretty much relegated to the back shop. We were
housed at Columbia Park where the computers are and
nobody from management ever saw us. We were
expected to deal with the supervisory people in the
operations area, and that was about it. Nowadays
systems people have some contact with all levels of
senior management. Increasing contact. So
certainly for people like myself in a middle
management position, there's a great deal of
increased visibility and a better image in the
organization. This upgrades the image of systems in
the eyes of the business people and increases
contact, thus increasing the prestige of systems.

Before decentralization, another middle manager said,
systems people were only occasionally included in business
planning meetings, and then on an informal basis. She told
of being at a planning meeting where she was the only
"techie:" "Before we were decentralized we participated in
things like that only where we had on a personal basis
established rapport in relationships. Now the organization
sort of mandates that type of participation, so it's good."
A third manager added, "We're real excited...that we're
bringing together two separate entities...The enthusiasm
level has raised considerably, because now it's no longer,
'Well, we're going to go in this direction, OK change, let's
go in this direction.' Now we know, when we take an
approach, which way we're going, and it's with the
commitment of the business manager." It was his feeling
that decisions that were once made unilaterally by the
business side were now made in consultation with systems people. The manager who said that the organization now mandates systems participation agreed that "systems folks feel less outcast and more a part of the organization...A lot of 'we against the world' has gone away."

2) Increased accountability of systems to business side

Along with the increased status and visibility of systems goes increased "accountability," according to systems people and users. Users are demanding more from systems people, and because most systems people now work directly for one of the five group executives, they are under greater pressure to produce systems which meet the user's needs. According to one former user, now in the systems area,

At one time early on in my career when I would work with the DP department I'd say I need this information and you would get a report that would be about that thick with what you needed plus other things. It was maximizing [the effect of] their programming effort to get out the information that you need plus somebody else needs. Now endusers are getting very particular about what they want...

A user who had participated in a systems development project believed that reorganization, and especially the fact that a decentralized systems organization now reported to the various group managers, was likely to have a significant impact.

Up to this point what was happening was that here you had ISS [that is, systems] and here you had the rest of the bank and if I was over here and I needed something from ISS, you could approach them and if it wasn't interesting enough, what difference did it make? I'm not paying their salary. I'm not judging
how well they perform... I think what will happen now is that you will have systems people working for each group, and for the first time there will be some feedback as to what it is their individual group needs. I think if those needs are not met, whether or not it's challenging or interesting enough will have nothing to do with it. It will be, 'Well, gee, we are paying your salary, and this is what we want and need.'

She added further that not only will users have more power to enforce that their requests are met, but also that some users will themselves be under increased pressure to produce because of competitive changes in the industry.

I think that now that you're dealing especially with lending people who are being beaten over the head on "Well, how much money do we make, and service your customer, and can you get blood from a stone," it's going to be a very demanding group. They're going to turn around and say [to systems], "Well, that's nice, but it's not good enough." So it's going to be very interesting to see what happens.

Speaking for the systems side, a member of the ATG described the changing situation in the systems organization. "As far as what the user wants, it's dependent on what he tells you. That's been the armor that systems guys have always wrapped themselves in: that you just don't know what they [the users] need to know. We're saying, 'That's true. But it's not an excuse anymore. Figure it out.'"

3) Efforts at mutual education

People on both the systems and the business sides described experiences in which they had spent time educating people on the other side about the content and meaning of their own work. This informal teaching reportedly takes
place at all levels of the hierarchy at which there are systems-users relationships, from programmer/analysts who interact with first-line supervisors to middle and upper management systems people who meet with their peers from the business side. A middle manager in systems said, "...it's a slow process of engaging them. They have little inclination or ability [for this]."

One environmental factor that slowly helps promote the education process is that young college and professional school graduates who start at the bank have more computer exposure than their counterparts of a decade before. "I think the most useful thing that can happen is that lending people will learn more about systems," said one user. "As we get more new people coming in, they are less afraid of systems...They come in, they have no fears about it, they just treat it as a fact of life. [But] half or maybe three quarters of the people...in this division would not go near a terminal if their life depended on it."

The head of one of the bank's information centers had observed the same phenomenon.

I think there's a tremendous amount of pressure on the users to become computer literate now, that I didn't used to see. It used to be OK to say, "Oh, I don't touch those things." There used to be a guy at Coca Cola who did transportation models on his calculator. To this day I don't know how he did it, but he did not trust computers...Now people have to use PCs in business school. The young hot shots come out of business school, they want a PC, so the other people feel pressured to get a PC.
Systems people endorsed the idea of each side learning more about the work of the other side. Not only would this promote smoother relations, they maintained, but it would be in the interests of the bank as a whole to have some of the knowledge of each group held by the other.

I think the best way to actually bridge [the systems/business] gap is for the systems people to get a little business knowledge—not necessarily tons of business knowledge—but enough to understand basically what's going on, and I think, more importantly, for the user people to learn more about systems, because then we're in a better position to do some of it yourself, or to think more clearly about what it is you will need, and then be able to communicate that...

said a user who had represented her area on a major systems development project. A manager in the systems area thought that not only should the systems and business sides learn more about each other's work, but also that people at every level of the bank should understand the organization's general direction.

...if you want to meet a goal, an objective, then I think everybody who is in a line to meet that goal and objective should know what it is. Everybody from the business guy who is directly responsible for meeting that right down to the coding level, right down to the computer operator, right down to the person who is responsible for enclosing your business statement, licking the envelope, and sending it out the door...I think we've done a real good job over the years of reaching down in the organization and touching everybody. That wasn't the case before. When I first started I was really concerned with doing my job and doing it well, and I didn't know where the corporation was going. I might have been lucky if I'd known where the department was headed.

5) Role of the systems organization

3 One man from systems was also quick to point out that many on the business side would also support learning more about systems.
Although there was evidence that the role of the systems organization was changing at the bank, several key managers still described the role of the systems organization as one of support to the business side of the bank. One manager called the systems organization "an essential supporting actor" and said,

The first function [of systems] is to work with the business people to determine how technology can be used to the greatest advantage to further business objectives. The second function is to see that those plans get successfully implemented, again by working with the business people. So it's very much a support mission, not a mission that has an independent character or flavor...It has always been a support function, but it was originally a support function with a rather narrow scope and a limited set of contacts. It's now a support function with an incredibly broad scope and contacts all over the organization."

While other managers who commented on this subject agreed that systems was, like operations as a whole, essentially a support function, there was evidence of an emerging paradox: although managers maintained that systems' role was to provide support, they also advocated a limited amount of initiative by systems people. The comment of the manager quoted immediately above is typical:

As banking, or parts of banking, become more and more information businesses, the systems people can sometimes take the lead, if we do it in a circumspect way, and become a little more than just support. For example, our corporate customer base--as we move our systems out into customers' offices--I think are going to want and demand a coherent, integrated, terminal-based method of getting into the bank and using a variety of services. This is something that systems people can perhaps foresee and initiate, rather than simply responding to a perceived user need.
Another manager agreed that systems was a "support function," but added another notion of what role systems was playing at the bank.

I don’t think that we have a life of our own...Now that doesn’t mean subservient, it just means that we’re there to help the business make money. I see "system" meaning a discipline, a methodology, and it doesn’t necessarily mean automation...What the bank has discovered, which we knew all along, is that in systems organizations you have people who understand how to run projects. Our business, the widget we do, is run projects. We understand concepts like critical paths, we understand things like defining success, and it’s amazing how many folks, lenders and the like, who are great at what they do, don’t understand how to run a project. It’s just a different discipline.

A manager who reported to this manager felt that the role of systems had in fact changed so much that it had ceased to be a support role and had become more of a partnership.

Because of our decentralization and reorganization we are no longer a central support kind of facility. We are now walking hand in hand with the business managers, leading them down the path of technology, so to speak. Before we were a support service for them...we were consultants, advisors. Tell us what you want, we’ll give you the best way to do that. Now it’s tell us where you want to be, and we’ll show you the best way to get there. Our goals and objectives are now the same as the business units’, whereas before our objectives were to "provide good support."

Even when the systems organization is behaving as a supporting actor, its activities have a real effect on the business side of the bank, sometimes bringing about a significant and unanticipated organizational change. A participant in the CBS project said that one of the results of the CBS project was a corporate-wide re-evaluation of
what information was needed, and a host of changes to the bank's underlying accounting systems. A manager from MISD said that the price of making MIS a top priority was "a certain rigor applied to the operating areas." No longer would systems that could not feed data to the corporate MIS be considered acceptable.

The top man in systems at the bank is co-author of a book on the management of information systems in large organizations. In the book he and his co-author suggest a role for systems that goes beyond that of support. He is a strong advocate for a centralized systems organization and for the appointment of a Chief Information Officer, whose role is far broader than that of simply overseeing the systems area and who works very closely with top management on the business side.4 At the same time, he sees members of the systems organization as very active, even aggressive, proponents of a systems perspective within the corporation. He uses the term "user penetration" to refer to the spread of systems, and of systems approaches, within the business side of the bank. He advocates the sending of "Trojan horses"—systems people who join the business side and then act as internal forces for change—as a way of getting non-systems people to adopt a systems perspective.

---

4 In an interview for this research, he described the CIO as a participant in strategic planning and a participating architect in the design of overall corporate strategy. In the past, he said, technology has been separate from questions of business and management. In the future, information systems will be an inextricable part of the front office, not just the back office.
The fact that not all people in systems agree about the role of the systems organization, and the fact that individuals may hold self-contradictory beliefs, is evidence of some confusion at the bank about what role the systems organization plays there and what its mandate is. Presumably some of this confusion is the result of the reorganization of the bank and of the problems that catalysed the reorganization, but one of the central tasks for the organization will be to bring about agreement on what the function of systems and the role of systems people really is within the bank.

A feature of the organizational structure which, by implication, suggests the role that the systems organization should play is the rules about whether or not users are charged for the services of systems people. In some cases, users are expected to pay for the time of the systems people who work on their projects, and in other cases systems services are free. A programmer/analyst in the Corporate Banking Systems area said, "We charge them back for what we do, so they see us as a service. The ATG, on the other hand, does not charge users for the services performed by its members "because if people have to pay for learning, they won't." The Office Automation group once charged users "44 dollars an hour for analysts' time," but had operated for a year without charge to users when they were assigned to work with the New England information center. The
information center charges users for its time, so the office automation people will have to begin to start charging, too.

6) The future role of systems

"Systems is a bubble rising to the surface of the organization" said a member of the ATG. As the bubble rises, the role of systems is vulnerable to change. Some systems people speculated about the future role of systems, contrasting it with that which still prevails in most areas of the bank.

[Under the traditional approach] you're talking about going into a business area and doing the work for them...It really is a service, as if someone were to paint your house. You'd call me up and say, "I think I need a departmental processer" or "I've got a mass mailing of 4,000 people twice a month," which is not uncommon at this bank at all. How do I do it? Traditionally, you go all the way from creating that product to hand them and supporting it for as long as that product is around...The nontraditional environment is trying to train and help people to do that kind of work for themselves and ultimately providing that support for themselves...

A member of the ATG argued that this "nontraditional" approach was the most effective one in the long run.

"Would it have been better to teach the guy who discovered fire how to cook, or would it be better to have him teach people how to build fires? You get a lot more bang for the buck by teaching all those people once how to build fires and letting them cook whatever they want...You get a lot more benefit from a lending officer being technologically literate than you do from a systems guy who's going to stay a systems guy understanding the lending process."

There are some serious barriers, according to systems people, which inhibit the transformation of systems people from the cook-for-you model to the teach-you-how-to-build-
fires model. Business people must have the time, the inclination, and the resources to commit to what can be a difficult and even threatening learning process. "The biggest task is, how do you teach people to do computer-aided tasks that they have never done before, have not bought into as part of their business plan, have not planned in terms of staff, resources, education, training, and support."

Even with the reorganization of the bank and the increased closeness of the systems and business areas, it remains clear who is a member of the systems area and who is on the business side. Despite the fact that people from one side sometimes camp out temporarily or transfer permanently to the other side, everyone knows which side they are on. Some claim that "the bank's emphasis is on making those lines less distinct", while others say, "It's like boys and girls: they're different, and they're supposed to be." On the question of whether the two groups will merge completely in the future, there was a tendency among systems people to believe that there would always be a role for systems-oriented specialists at the bank. People would always be needed who could maintain the hardware and software, and a manager at MISD suggested that the role her group should take in the future would be that of manager of the data which users draw on as they do their own analyses. Another manager suggested that the long-term role for systems was as
agents of technology transfer to the rest of the organization.

Who are the users?

Systems people speak of the "users" as if they were a homogeneous and clearcut group. In fact, the user community consists of several segments, each with its distinct relationship to the system and to the systems development process. The group requesting the system is often different from the group which will directly interact with the system when it is built. These two groups are in turn different from the groups which may be affected indirectly by the finished system.

In general, when systems people speak of working with users during the systems design process, they are speaking about professionals and managers--often first line supervisory personnel--who participate as user representatives in the process. Even the participation of these groups is relatively recent, however. Five and more years ago, the people who participated in systems design were from the operations area. Now front office personnel may participate in systems design efforts as well.

Systems people distinguish, when asked, between different sorts of users. A young programmer/analyst in the New England Systems group said, "...the people that actually

---

5 I am indebted for this insight to the members of the International Computer Occupations Network [ICON], based at the University of Bristol in England, who since 1982 have been conducting international comparative research on computer occupations.
use our output in a lot of cases are clerical... The users we talk to are professional [shift managers, etc.], but there are a lot of proof machine clerks, who do a lot of the work that is still manual and needs to be manual until they come up with machines that can read handwriting, but they do a lot of the preparatory work for input into our system." A manager in the New England Systems area pointed out that the bank’s customers are in some sense the ultimate users of the system.

The distinctions between segments of the user community are highlighted when two different groups have differing agendas for the system. This is probably a common occurrence. A user representative to the CBS development process describes the result of conflicting agendas.

What [the loan officers] see is that it’s more work for them [with the CBS system]. At this point, I’m sure they feel there’s very little return. CBS was developed not specifically to make life easier for the individual lending officer. It was really developed so that the people on the second floor, the board of directors and whoever, would be able to quickly and easily get information—the big picture type of information—on where we’re making money.

A member of CBDG gave her thoughts on this conflict:

"For the users, a new system can be more work. You should really give them more capabilities and benefits to offset the more work they have to do."

PC’s and end user computing

In addition to the participation of users in systems design and in report writing, users are increasingly doing their own data processing. The amount of computerized
analysis performed by users themselves remains minuscule in
comparison to the volume of work done by the systems area,
but by 1985 the bank had about 600 Personal Computers (PCs)
located in various user areas.

Almost all computing done by users takes place on PCs. The other
mainframe and minicomputer systems used for data
processing can be quite complicated and confusing for novice
users, and few users have the time or are sufficiently
trained and motivated to overcome the barriers to the use of
larger systems. "...it's frankly usually a quite laborious
effort for a common user to get logged onto the trust system
or to get logged onto CIF or whatever it is they need," said
a member of the systems staff who works with office
automation users. In addition to being laborious, it can be
frustrating and expensive.

The development of the PC is a very recent phenomenon,
but its impact was swift and widespread. Although there
were desktop computers before 1980, they were generally the
province of the engineering hobbyist. PCs did not become a
commodity with commercial appeal until the 1980s, and it was
only with the 1981 announcement by IBM of its first personal
computer that the business market really caught fire. The
PC permitted the further extension of what systems people
described as a trend over the last decade or more: the trend
of "delivering technology to the end user."
The head of one of the bank's information centers explained why she thought PCs had become popular with professionals at the bank.

I think the main reason people use PCs is because they have control. They have control over the whole environment. They don't have that with a mainframe. Even to get logged onto a mainframe you have to talk to people, you have to get an ID [identification code], you have to get a terminal, and then you get a blank screen that you don't know anything about and it's very intimidating. Whereas with a PC you have the whole environment right there in front of you. I think another reason people use PCs is because it is more prestigious. Now people coming out of business school have to use PCs in business courses and [other users] want to stay at least with the brightest people who are using that environment. It is probably less prestigious for those people to use a mainframe, and people probably associate you with a programmer or something, whereas somebody who uses a PC certainly wouldn't be considered a programmer because they probably use [software] packages..."

There are other reasons why professionals use PCs as well. PCs may make it possible to do things that cannot be done, or that cannot be done as easily by the user, on a larger machine. Some of the software available for PCs, according to the systems person just quoted, is actually more sophisticated, and certainly more "user-friendly," than much existing mainframe software. For some applications, an important motivating factor is that requests to the systems area for a particular piece of data processing work are subject to delays. Systems areas in this and other corporations usually have a significant backlog which--when added to the time it normally would take for a user to meet with a systems person, explain what they want, test the
output, and so forth—becomes prohibitive when a report or calculation is needed quickly.

PCs can be put to a variety of uses. At the bank, according to the head of the systems area, most PCs are being used for "task automation" and management information/management support applications. Few are being used for newer and more innovative applications, which he dubbed "competitive support."

One of the benefits of the spread of PCs in the user areas of the bank, according to the head of the ATG, is that users develop more compassion for systems people and the complexities and frustrations of systems work. Through their own firsthand experience with a computer system far simpler than that used by systems people, the users come to understand that, in his words, "you can't just open a goddamn box" and have it work the first time.

There is a major technical barrier to transferring increasing amounts of data processing work to users themselves. The data sets analyzed for much of the report writing that takes place at the bank are far too large to be handled on a stand-alone PC. "Our data files for the retail side of the bank are so large that it just doesn't facilitate end user computing. You have to have some pretty sophisticated tools and hardware to accommodate [such files]," said a systems person who writes a large number of reports for users in the New England area. Even if users had easy access to the large computer systems and data sets,
there might be reasons to limit their use by novices. "It takes a long time to read all those records, and if you do it inefficiently you could tie up resources for a long time." What the systems area can do for users, and is trying to do more frequently, is to provide extracts of the data to users when a small subset of the information available in the larger file would suffice.

Another barrier—at least temporarily—to user computing is the recent attempt by top management at the bank to cut "non-interest" expenditures where possible. According to a systems person in the New England area, requests by users for new purchases of PCs dropped to zero immediately after the drive to cut non-interest expenses was announced, and requests had only begun to trickle in three months later.

**Information centers**

The Information Center is an institution, touted by IBM, which acts as an internal catalyst and source of technical assistance for end user computing. It takes different forms in different corporations. At the bank, there were at least two information centers, which assisted users in the New England and Corporate Banking areas. The New England information center provides three sorts of services. First, it produces reports on an ad-hoc basis for users in the New England area. This is by far the most popular of its three activities. Second, it helps promote end user computing, providing technical assistance and
extracts of data files when possible. Finally, it provides tools (not necessarily PC-based) for decision support in areas not covered by the bank's conventional systems.

The work of non-systems people

To people on the business side of the bank, the systems people, who design and build computerized systems, are the facilitators of the new "numbers-driven bank." Although the systems people do not themselves set the policy which mandates more extensive use of computers at the bank, they certainly form an institutional base of support for such a policy. In turn, the growing use of computers at the bank affects not only the jobs of systems people, but also the jobs of those in the business areas of the bank. While the subject of this research was the systems area and its organization, there was some evidence from interviews that changes were taking place in the business areas as well.

Reports from systems and business people suggested that systems were used or contemplated chiefly to increase control of operations, to reduce costs, and to improve the quality and effectiveness of customer service. All of these can affect people working on the business side of the bank.

Computer systems can be used, deliberately or unintentionally, to enforce bank procedures and to monitor the work of employees, including professionals. According to a nonprogramming analyst, the process of implementing a new system may uncover situations in which employees, for good reasons or for bad, are not following the rules in the
course of their work. "With the fee control unit," she said, refering to a system she was working on, "when I'm doing the analysis work, people tell me what they're doing and I find they're not following procedures. I'm the bad guy who makes a system that will prevent that." A user with systems experience from the lending area spoke of the new Relationship Planning System and the things it requires of loan officer.

...officers are actually asked to put in writing a marketing plan. What is your strategy for this particular customer? This is a key customer. What do they do at the bank now? What are our opportunities for going forward? [They are asked to] actually put that in writing. Additionally, indicate what we have made profitability-wise last year, write down where you think you can make money next year. Quantify where you think the money will be made...As time goes on we'll be tracking how well people have predicted what we'll do. For two reasons. One, to track if an account officer is actually performing up to what we think he can do, and additionally, is this customer really worthwhile? For the amount of time we put in, are we getting the return we think we should be getting?

A member of the ATG justified the work of his group by explaining that in a competitive industry, an institution looking to cut its costs should not ignore professional workers.

The office systems [automation effort] was very, very important. We did a wonderful job of leveraging the clerical staff.6 They probably

---

6 Several members of the ATG used the word "leverage" when they talked about increasing the productivity of bank professionals and clericals through automation. By "leverage" they seemed to refer to the process by which one extends the scope or the usefulness of an original investment (whether in people or in systems) by adding a small incremental investment. The head of the ATG tried to convince me that they were not trying to increase productivity--since that was the mission of
represent 20-25% of salary costs. Once you start attacking higher levels in the organization...if I save a lending officer an hour a week I’ve saved sixty bucks. If I save a typist an hour a week I’ve saved ten."

What effect do these changes have on the user community? This research generated scant evidence on these questions, but some reports about professional workers pointed to a significant effect. An observer of lending officers said,

"Banking has been very traditional. You knew exactly what your job was, this was what was required, you knew exactly what you had to do. With the industry changing and the bank itself changing and trying to decide where they want to strategically place themselves...In the marketplace, you can’t be all things to all people. You probably had a lot of very fixed, stable type people who worked here in banking and they knew exactly what it was...and now they know exactly what it isn’t. For those people it’s very difficult to deal with change."

Some business people actually leave the division or the bank because they don’t like the changes taking place in their jobs, she asserted. It is important to note that these changes are not a result of computerization per se, but rather of larger competitive changes in the industry. Top management at the bank has responded to these changes in ways that affect the jobs of many people in the organization. Computerization may best be understood as an outgrowth of managerial strategy, rather than as an

---

another group at the bank--but that they wanted to expand the effectiveness of income producing professionals and thus increase the bank’s income. I was unconvinced that this was not the same as increasing productivity.
independent phenomenon with a significant impact on the nature of work at the bank.

Systems themselves can, of course, affect the work of business people, as the quotation from the nonprogramming analyst indicates above. A user representative to the CBS development project reported that when loan officers use the CBS system, they can change data in some parts of the system but not in others. Data about customers themselves—such as address and telephone number—can be changed, but the information on customer loans can only be altered if the officer sends an update to others, who then revise the numbers in the underlying accounting system. This feature of the system may not represent a significant change, however, since even before computerization officers may not have had unfettered ability to make such changes.

A member of the ATG spoke about the changes he saw taking place in the work of bank officers. While on the one hand the ATG is trying to deliver computerized tools to professionals at the bank—lending officers, for instance—on the other hand the work of some professionals is becoming in some ways less routine. Since it is routine work which is most easily automated, this poses an apparent contradiction for the members of the ATG who are trying to bring computerization to a professional domain. This contradiction underlies the ATG member’s comments on the use of Artificial Intelligence (AI) techniques at the bank. The "expert systems" now on the market he finds
...not very meaningful at a professional level. They have a lot of potential as point-of-sale items...where the environment can be fixed...within a fixed domain. The rules of the game have been set beforehand. Corporate lending, and now to a greater extent retail lending too, is a much more individual-deal, independent process that nowadays requires, because of competition, a lot more of the experiential and innovative.

Since in his view lending work is growing less routine, it is therefore harder to automate—harder, that is, to replace with a machine. Yet the resolution to this contradiction comes from the fact that, at least at present, the members of the ATG do not see their mission as one of replacing professionals with machines. Instead, they want to supply tools which will make professional workers more effective and productive.

Yet these two goals amount to the same thing. By increasing the productivity of professionals one is decreasing the size of the professional staff needed for a given volume of business. In a sense, then, some aspects of the jobs of professionals are being replaced by automation. There is, of course, much of professional work that is nonroutine, and these functions remain to be performed by professionals at the bank. The ATG member seemed to see his group’s role as one which works for the rationalization of effort—the reduction of legwork, reduction of duplication—for bank officers. For example, he suggested that an expert system could help an officer to screen potential clients for other departments.

If I have a lending officer, for instance, that is out selling funds and he goes to a customer who
might be marginally credit-worthy given our standards, he might be a good candidate for leasing. Now that lending officer, because he might not know a damn thing about leasing, generally has to call the leasing department, tell them he has a lead, they hear from the leasing guy, he goes out and spins his wheels, and it turns out they never should have been considered for a lease at all. If I could have the leasing guy's model--his work, that we developed with him--on the network, the lending officer could come in and say, "There is that menu that has a leasing pick on it." If he was to hit that, and be prompted, even on the surface level, with questions about the customer and what the equipment was, and he could have some routine presented to him to make that evaluation. That's better than calling the leasing guy, having to wait for him to get out there, getting back his assessment, things of that nature.

With this optimistic statement from a member of the ATG touting the possibilities of computer technology at the bank, this two-chapter discussion of the nature of systems work and its role in the larger organization at the bank comes to an end. In the next chapter, the same evidence is reviewed more briefly, in a form which contrasts the evidence with the theories presented in Chapter Two.
CHAPTER FIVE

COMPARING THEORY AND FINDINGS

As the evidence presented in the previous chapter suggests, an investigation of the organization of work in the systems area of a large bank found little to confirm the predictions of Braverman, Kraft, and Greenbaum. The previous chapter described the evidence gathered at the bank and organized that evidence into three categories, corresponding to the three dimensions of Braverman's theory: fragmentation and the detail division of labor, separation of conception and execution, and degradation of skill for the mass of jobs. The present chapter reviews the same three categories, but unlike the previous chapter, which described the bank in the language used by its members, this chapter casts the evidence in the language of the theories under discussion. Under each category heading is a review of the evidence for or against the predictions of Braverman and his followers, and a discussion of the implications of the findings for the meaning and viability of the theory.

Fragmentation and a detail division of labor

Although work in the systems area was organized hierarchically and most projects involved multiple participants and a division of labor, there was no generalized evidence of the sort of fragmentation and detail
division of labor which makes individual tasks meaningless for those who perform them. Several findings support this conclusion.

1. No occupational ghetto for coders or maintainers

Writing code was not considered a dead-end job, and entry-level people could expect promotion, often into jobs involving little or no coding. On some projects senior members of the project team also wrote code, so that coding was not just an entry-level task. Furthermore, systems programmers, who were highly esteemed for their technical ability, wrote code as part of their jobs.

Maintenance, considered by many to be the most routine work done by systems people, contained many challenging elements, including the necessity to solve technical puzzles crucial to the continued functioning of the bank. A growing number of people, including experienced systems people, were said to consider maintenance important and attractive work.

2. Assigned to a particular business unit, which limits the extent of the market

Recalling Adam Smith’s dictum that the division of labor is limited by the extent of the market, we find that in the bank the market for the services of systems people is fragmented and that the opportunities for the detail division of labor are therefore restricted. Although it may seem startling that a commercial bank operating in a mass market does not afford adequate opportunity for the detail
division of labor, the market for the bank's **products** is not the market which determines the extent to which systems labor can be divided. Instead, systems people face a different market: the bank's internal market for their services. Systems people construct software systems to serve mass markets, but their own customers are the employees in the business areas of the bank. These employees are divided by function into groups, and it is these groups that systems people serve. Furthermore, since decentralization, the systems area is also divided into small groups, which are paired with client groups. As a result, the opportunity for large-scale division of labor is minimized.

3. **Movement from detail work to "whole" work and back, depending on the job**

   On the whole, even entry-level systems people seemed to have jobs with significant variety in them. While a person could be assigned to a large development project and work on the same system for a matter of a year or even considerably more, systems people more commonly worked on a variety of projects of different sizes, often tackling more than one project at a time. On one project their role might be larger or require more of an overview than on another (perhaps because the projects were in different phases), so that a systems person would experience a shift in the scope of work when going from one project to another. This gave
considerable variety to the daily schedule of a programmer or analyst.

4. **Fragmentation is not equivalent to simplification.**

Implicitly or explicitly, Braverman, Kraft, and Greenbaum treat fragmentation and simplification as intimately linked. While it may be the case that by any meaningful measure, the fragmentation of work also leads to its simplification, the reverse is not true. The history of the computerization of work offers many examples of situations in which work was simplified but not fragmented. At the bank, the use of higher level languages by programmers allows programmers to write simpler code more easily, but it does not fragment the work of programmers. In some cases, in fact, use of such languages may lead to more unified work when it allows the size of project teams to be reduced. Fragmentation and simplification, then, are separate concepts, and should be treated independently when used for the analysis of the changing organization of work.

**Separation of conception and execution**

The idea that the managerial control of work entails the separation of conception and execution is as central to Braverman's theory as it was to that of Frederick Taylor, on which Braverman bases much of his own work. In order to test the predictive power of Braverman's framework, then, it is crucial to investigate the extent to which conception and execution are separated in a given labor process.
Braverman, Kraft, and Greenbaum all identify analysis as the conceptual element of programming, and coding as execution. Analysis, in their view, encompasses the planning work required for the production of a program, and coding consists merely in transforming what is already planned into computer-readable form. In drawing these parallels, Braverman, Kraft, and Greenbaum also affirm the conventional view that there is a hierarchical relationship between analysis work and coding work: the analysis is done by upper-level workers and the coding by less skilled detail workers.1

Little Separation of Coding and Analysis

In fact, this research uncovered no systematic and wholesale separation of coding and analysis. Some who did analysis work did not code, and some who wrote code were involved in projects for which they did not have primary

---

1 As noted in Chapter Two, Greenbaum does briefly discuss the separation of programmers from computer operators, suggesting that this separation could be considered a separation of conception and execution. Under this alternative model, it is the operation of the hardware, not the writing of the code, which constitutes execution. This model seems inappropriate for a variety of reasons. The systems programmers and operators, a much smaller group than the group of applications programmers, stand in a service or support relationship to applications programmers. They are more like the maintenance mechanics who keep the programmers' tools in order than they are like routine manufacturing production workers. There may have been a larger element of routine production work like the printing of reports prior to the early 1980s, when a concerted attempt was made to "professionalize" the operations staff. Still, it seems more plausible to look to routine work in other areas of the bank rather than to point at the work of operators if what is sought is routine execution work.
responsibility for overall analysis, but there were no cases found of regular bank employees who did coding but no analysis.2 There was considerable talk, in fact, of the importance of having the person who was doing the coding also do the analysis where possible.

Because this finding is so critical to a test of the theories of Braverman, Kraft, and Greenbaum, it is appropriate here to ask about the reliability of the evidence that there are no "pure coders" at the bank. This conclusion is based on reports from all of the interviewees in the systems area, from recent hires to upper middle managers. Every systems person interviewed said that programmers did both coding and analysis, and offered several reasons why this was the case.

This conclusion is not based on independent examination of the tasks performed by programmers, although such a study would certainly add extra weight to this conclusion. The evidence from interviews seemed strong enough to be convincing, especially because some of those interviewed were among those most likely to be "pure coders," if such a group existed. Based on the work of Kraft and Greenbaum and

---

2 The closest thing at the bank to a job involving only coding is the work of contract programmers, who are often given tasks involving maximum coding and minimum analysis. Managers in the systems area justify this by saying that they want to give the most interesting and varied work to their own people, whom they are trying to develop. Another reason for perhaps wanting to minimize the amount of analysis done by contract help is that these workers are least likely to know or care about what is being done in the business areas of the bank.
on the evidence from the bank itself, "pure coders" would be most likely to be found in the lower echelons of large factory-type systems development projects using primarily COBOL. The fact that systems people who had participated in such projects reported that there are no jobs which involve solely coding strengthens this finding.

Why is there no Separation?

There appeared to be no single reason why execution and conception were not segregated. Instead, there were many factors contributing to this outcome.

1) Most large systems already developed.

The proportion of systems workers occupied in the construction of large systems has fallen as smaller-scale projects and maintenance work come to dominate the time of systems people. In the 1980s, it appears that fewer development projects for large transaction-based systems are taking place, since many of these systems have already been built. Smaller projects, like maintenance work, offer fewer opportunities for the division of labor, as discussed above, and therefore also for the separation of conception and execution.

2) Labor market constraints.

Managers in the systems area seemed concerned about recruiting and retaining capable and motivated systems people. Past experiences with uncomfortably high rates of
turnover and difficulty finding technical specialists for positions above entry level have prompted the bank to show increased concern about salaries, working conditions, and intrinsic interest and challenge in systems work. One of the consequences of this is that no one in the systems area is expected to spend full time writing code. Coding, especially COBOL coding for banking applications, is considered by most to be unattractive as a fulltime preoccupation. In general, systems people said, the smart liberal arts graduates from "starvation majors" whom they recruit for entry-level positions do not want to spend the rest of their careers writing code.

3) High level of ambiguity in work.

Several systems people spoke of the great ambiguity which sometimes characterized their assignments. It is certainly true that programmers, because of the nature of their work, never do the same thing twice. Unlike workers on an assembly line or in a routine clerical work situation, the work of programmers is not standardized. Given this fact, it seems a reasonable conjecture that the separation of conception and execution would be impossible, or at least seriously counterproductive. Kraft says that routinization and fragmentation require "(1) an unvarying good or service and (2) an unvarying way of turning it out." (1977, p. 52) Programming has neither. This does not rule out the possibility of at least a degree of separation of execution
and conception, but it does suggest why programming work at the bank is not fully segregated.

4) Contextual knowledge needed for good systems work.

   Because the vast majority of all systems failures at the bank were caused by poor communication between users and systems people, more stress is now being placed on careful analysis and improved understanding of business problems. Many systems people emphasized in interviews the importance of knowing the business context when working on system development. Such knowledge is useful even at the detailed and technical level of writing the code, so it is useful for coders to understand the uses to which their system will be put. This suggests that coders will do a better job if they are also privy to the analysis work involved in system development.

   A related advantage of having programmers who are both analysts and coders is that the same people can follow the project through from beginning to end. While this does not always happen, it does seem to be the norm. More than one manager argued the advantages of having analysts who also do the coding, reducing potential communication problems.

   One manager saw the increasing emphasis on context and on understanding contextual factors as serving to increase the skill required of programmers in the systems area. "Deskilling is doing a job with no context," he said after reading an early draft of the study. He believed that the
systems area of the bank was moving in a direction opposite
the one which the term "deskilling" suggested to him.

Definitional problems with terms "conception" and
"execution"

Braverman and his followers discuss conception and
execution as if the meaning of these terms were clear and
unambiguous. As the evidence from the bank reveals,
however, this distinction can be quite difficult to make in
practice. Programmers found it difficult to say where
analysis left off and coding began, and when pressed one
programmer seemed to suggest that analysis was endemic even
to the writing of code itself. Any reflection that took
place during coding she considered to be analysis, and
reflection seemed to be a common component of the coding
process. This suggests a degree of overlap—even an
inseparability—of conception and execution for which
Braverman’s theory does not allow.

Bryn Jones makes a similar criticism of Braverman in
his study of numerically controlled (NC) machining in small-
batch British engineering firms.

The conception-separation thesis leads [Braverman]
to describe programming positions as devoid of
metal-cutting experience, and machining jobs as
stripped of a knowledge of tool control. Yet these
lacunae in job knowledge must, in his terms, also be
considered conducive to production efficiency. It
is difficult to see how this could be achieved by
such a double deskilling [of programmer and
machinist] since...some tool knowledge is required
on the shop floor during machining operations and
metal-working experience is required in advance of
machining by part-programmers. (1982, pp.188-189)
In light of these difficulties distinguishing conception and execution, it might be more useful to think of conception and execution as two poles, with a spectrum of tasks arrayed between the two extremes. Some tasks clearly have a higher level of conceptual effort, while others may require more dexterity and concrete production knowledge. Such a revision of Braverman’s model would address the empirical problem of classifying the many tasks which appear to have both a conceptual and an execution component. In a systems development project, is there a "moment of conception" in which the decision is made to build a particular system, and all that follows is simply the execution of that decision? Or is all the work of systems development conceptual, except for the short period when the programmer is sitting at the keyboard or the pad of paper and visibly writing? Between these two extremes is a large grey area which is problematic for Braverman’s model.

Degradation of skill for the mass of jobs

Certainly the most familiar of Braverman’s predictions is his assertion that the managerial reorganization of work aims at the reduction of skill requirements for the mass of jobs. This phenomenon has been dubbed "deskilling" by Braverman’s disciples. It is a measure of the powerful appeal of Braverman’s notions that the term’s use has now gone far beyond the academic literature and become a staple of popular writing on the changing economy.
"Deskilling" is a misleading term, however, because it focuses only on one half of Braverman's prediction about the skill content of work. What Braverman was in fact predicting was a polarization of skill, as managers increasingly succeeded in following Taylor's dictum that the planning and control of the work process be removed from the shop floor. The consequence of this, Braverman saw, was increasing skill in the hands of managers and decreased skill possessed by the majority of the workforce.

This study cannot provide a conclusive answer to the question of whether programmers are less technically skilled than their counterparts five, ten, or twenty years ago. Further research is needed to resolve this question, but the path to such research is fraught with difficulty. Several impediments hinder research on whether polarization is taking place in software work at the bank.

First, it is very difficult to measure programming proficiency. It is part of the oral tradition in the software community that there variations in productivity of as much as 20:1 between the best programmers and the worst, and it may be difficult for an employer to gauge where on the spectrum a particular programmer stands. This problem would certainly hinder an investigation of whether levels of technical skill were rising or falling in particular parts of the software population.

It certainly seems to be true at the bank that applications programmers, who are the majority of
programmers in the systems area, know less about the hardware with which they work than was true when they worked in an area adjacent to the machines. From a technical point of view, it appears that applications programmers can be effective with a lower level of technical proficiency than was true in the past. While there may have been little change in the language used by programmers, a variety of aids have been developed to expedite the other tasks associated with producing applications software.

For the most part, programmers at the bank are still using COBOL, which applications programmers have used for 30 years. The language is still the most prevalent one for business applications in the country as a whole, and the bank continues to rely on it very heavily. Applications programmers at the bank generally do not know assembler, a language which tends to be the province of the more technically oriented software people, but it is not clear to what extent their predecessors did know such first- and second-generation languages.

Applications programmers are constrained in how they use COBOL, in that they are instructed to use structured methods when writing a program.3 They are told not to use

---
1 Structured programming is a set of rules and principles said by its advocates to represent "good practice" in programming. Rules prohibiting the use of GOTO statements, for example, are intended to avoid the production of "spaghetti code": that is, code which has a tangled and convoluted structure. Spaghetti code is said to be extremely difficult to de-bug and to maintain. Since both of these functions are crucial to a robust and long-lived system, structured methods are often enforced by management. At the bank, structured
GOTO statements and nested IF's, and code may be reviewed for good structure. Since maintenance is a large part of systems work at the bank, techniques of this sort which expedite maintenance are considered highly desirable.

Systems programmers are more likely to know first- and second-generation languages than are applications programmers. One could make the case that a skill polarization has developed which puts systems programmers in the upper tier and applications programmers in the lower, and there is considerable anecdotal evidence to support this. Thus, from the point of view of technical work alone, Braverman's prediction of skill polarization appears to be confirmed. Such a polarization of technical skill does not, however, truly confirm Braverman's prediction, since technical skill is only one aspect of the total spectrum of skills required to be effective in systems work at the bank.

Finally, the complexity of computer systems has increased dramatically over the years. On the one hand, this increased complexity need not mean increased skill requirements for all programmers. Indeed, it has made code has only been required for about six years, according to one manager. Since that time, trainees have only been taught structured methods, and they are rarely exposed to the use of GOTOS or other unstructured practices.

Kraft (1977) cites structured programming as the primary tool used by management in its effort to deskill the workforce. Since many programmers are among the advocates of structured methods, and since even programmers working alone may choose to write structured code, this approach clearly has other benefits as well. Thus Kraft's contention remains to be proven.
possible programming by those with less technical skill, since greater memory capacity can allow the use of higher-level, less complicated programming languages and productivity-enhancing tools. Softening this effect, however, according to some in the systems area, is the fact that increased complexity leads to a proliferation of technical "niches" for people who become expert in the use of a particular database system or some other particular systems technology.

A shift in skill requirements

In the larger community of software specialists, work in the systems area of banks and insurance companies is regarded as routine and relatively unskilled. The software engineer who referred to these workplaces as "sweatshops" and compared them to the New England textile mills of the nineteenth century was suggesting that a significant decline has taken place in the skill requirements for these jobs—if indeed high levels of skill were ever required. The likelihood that this view is accurate with respect to technical skill alone has been discussed in the preceding section. Far more important, however, is that this view overlooks some of the most important aspects of software work at the bank.

To measure the skill required to be a successful programmer at the bank simply by examining the amount of technical skill required is to use the wrong yardstick.
Certainly technical skill—proficiency in coding, ability to understand the hardware, and so forth—continues to be a key element of systems work. What the interviews at the bank document, however, is the large and growing component of systems work which is not strictly technical. Systems work has always been conducted within the context of a business objective, and everyone in the systems area stressed the importance of the ability to understand the business of the bank and to communicate with other systems people and with the user community about how a system should be designed and built.

The ability to analyze a business situation and the ability to communicate with others about the design and maintenance of a system are clearly skills which are crucial to an effective systems development process. They are not, however, generally considered technical skills by systems people. Instead, they are a different sort of skill, requiring a different set of abilities and a different outlook on the process of program production. As such, they are measured on a different scale than that which measures technical proficiency. Both technical and non-technical abilities are required for good software work.

The increased emphasis on analysis and communication as opposed to strictly technical work has evolved over the course of years at the bank. Many factors have contributed to this shift in emphasis. Each of these factors is discussed briefly below.
1) Reasons for systems failure largely not technical

As one manager emphasized, the vast majority of systems failures are the result of problems in communication, not the result of technical faults. For this reason it has made sense for managers in the systems area to put increased stress on analysis and communication, the areas of greatest weakness in the process of systems construction. Many systems people said that time spent getting the specifications right saved time at every subsequent stage of the systems development process, and that such time was therefore well spent.

2) To retain many employees, must provide interesting non-technical work

The sort of people the bank hires for entry-level programming work have non-technical backgrounds. While they may be interested in the technology, they are unlikely to be satisfied for a long period working at a job which is purely technical in its preoccupations. Increasing the exposure of systems people to the context of their work serves the purpose of expanding their horizons to include analysis and communication.

3) Fewer new development projects, more maintenance

Because there are a growing number of systems which need to be maintained, maintenance has become a larger part of the total workload in the systems area. Since
maintenance is said to involve a higher proportion of analysis and a lower proportion of coding than does new systems development, the increased importance of maintenance increases the share of worktime spent on analysis rather than on coding.

4) More outside purchasing of programs. Systems supplemented by PCs, packages.

The bank is increasingly turning to the outside purchase of packaged software, and is building proportionately fewer systems in-house. All the software used on PCs has been purchased outside the bank, and many minicomputer and mainframe systems are also purchased packages. These packages may be adapted for use by the bank, but much of the coding that would otherwise be necessary to build an entire system is eliminated. Outside purchase does not, however, eliminate the analysis work involved in system design and implementation. Much analysis work is required as a prerequisite to choosing a software package, and user training and support are not eliminated simply because an outside vendor has supplied the software. Packages purchased for PCs represent the limiting case of this trend, since their installation requires analysis—that is, an assessment of the job to be done and the tools available to do it—but no coding.

5) Programming tools are becoming easier to use.

The bank makes some use of new software tools which expedite the process of coding. Although still heavily
invested in COBOL, systems people have also used at various times some of the database packages and fourth-generation languages available on the market. These tools can have the effect of significantly reducing the time spent on technical issues, particularly coding. As such, they contribute to the shift in emphasis away from technical concerns and toward analysis and communication.

In addition to these tools, the use of aids which expedite the aspects of the work which are not simply involved with the production of the code itself—that is, for example, the areas in which programs intersect with data sets and operating systems—have reduced the minimum requirement for technical knowledge among applications programmers.

6) **Falling hardware costs make optimization less important in some situations.**

In the past, one of the activities programmers engaged in was the process of trying to write a program so that it used machine time and space as efficiently as possible. This process of optimization could be enjoyable for those who liked to solve technical problems, but it was also time-consuming and therefore expensive for the employer. Today, hardware costs have fallen so significantly that machine efficiency is a much less important issue.

7) **More emphasis on computerizing professionals and other poorly understood subjects, less on automation of the already-rationalized**
When work processes that are rationalized, fragmented, and well-understood are automated, the analysis process can be relatively straightforward. Many of the early bank system were computerized versions of work that had been done manually by detail workers. The analysts generally saw themselves as simply the translators of a paper process into computer terms.

At the bank today, most of the large, rationalized, transaction-based systems have been computerized. More recently, systems people have been turning their attention to more conceptually difficult problems: the construction of systems which have not previously existed in paper form, and the automation of some aspects of professional work. This new kind of work takes more analysis time for two reasons. First, it is conceptually more complex, and requires more analysis, and second, systems people generally approach professional users differently than they do the clerical workers who work in the rationalized environments. With professionals, systems people feel they have to "sell" their product, while clerical workers are generally faced with a "take it or leave" situation.

The Debate About Skill

The evidence from the systems area of the bank makes it possible for us to reexamine the debate over whether "deskilling" is taking place, and recast it in new terms by challenging some of the current terms of the controversy.
The debate over the changing skill requirements of industrial settings has three sorts of participants. There are those, like Larry Hirschorn and Paul Adler, who believe that on balance skills are upgraded as technology progresses; there are those, like Braverman, Kraft, and Greenbaum, who believe that the skills of the majority are downgraded; and there are those, like Wood and Jones, who believe that both upgrading and downgrading take place, and that it is currently impossible to say on balance which takes precedence.

When examining closely the change in systems work, one of the features of the change which is most apparent is the transformation of the work from more technically-oriented to more analysis- and communication-oriented. There is little discussion in the literature of the methodological problems raised by such shifts in the type of skill required. Adler, for example, claims to have found skill upgrading accompanied by a change in the sorts of skills required of bank tellers, but does not offer a convincing argument for why the old skills make a teller less skilled than the new skills do.

The most careful and convincing account of skill is given by Maryellen Kelley (1987), who suggests that skill is really multidimensional when studied in an applied industrial setting. Based on her examination of the labor process in metalworking machining, Kelley distinguishes three dimensions of machining skill: conceptual demands,
skill in executing a given course of action, and the scope of knowledge which a set of tasks requires. For any given set of tasks, skill may vary along all three of these dimensions. Kelley's model offers an opportunity to liberate the debate over skill from the narrow, one-dimensional approach, which holds that skill must either go up or down.

If a systems person were to learn new analytic and communications skills at the expense of coding skills, would an upgrading or a downgrading of skill have taken place? To answer this question it would be necessary not just to determine whether skill had increased or decreased along two dimensions--coding/technical and analytic/communicative. It would also be necessary to develop a basis for comparing these two quite different sorts of skills, in order to say whether there had been an overall increase or decrease in skill. The task of making such a comparison is exceedingly thorny. Braverman, Kraft, and Greenbaum, for whom skill is one-dimensional, do not even recognize the existence of the problem.

The "user friendly" controversy

What are the practical implications of stressing the existence of a shift in skills, not just of a raising or lowering of skill? Because the problem of comparability is a difficult one, an attempt at comparison highlights the difficulty of measuring and evaluating skill. It also focuses attention on the notion of skill itself and on the
question of why we care about skill, which will be discussed in a separate section below.

An example from a software setting of the practical implications of thinking about skill and deskilling can be seen in the controversy over "user friendly" software. This euphemism is usually used to describe software which is easy for a novice to use. Ease of use comes from several factors: legibility and comprehensibility of the screen, tolerance of user mistakes, ease of understanding what to do next, the opportunity to correct problems and recover from unexpected events, and the ability to perform in a relatively small number of steps, and without complicated jargon, the work that brought the user to the computer in the first place.

Some software experts criticize "user friendly" systems because they limit the options of users and do not teach users how the computer really works. A programmer writing COBOL or another high level language has more flexibility than a novice using a software package. The novice will learn how the package works by using it, but will learn little about the underlying makeup of the computer. In one sense, these things are disadvantages. Seen in another light, they are what make enduser computing possible.

To see user friendly packages as deskilling, as some do, you would have to believe that 1) programmers who might otherwise have used a real computing language are forced instead to use these packages, or 2) that users who use the
packages would have learned to program if the packages were not available, or 3) the packages are putting programmers out of business. At the bank no cases of the first were found during the study, although they were not explicitly sought. The second might possibly have happened if there were no PCs, but it is likely to have been so rare as to be negligible. As to the third, there seems certainly to be some truth here, but its extent is hard to judge. Certainly no one has been laid off from the systems area for lack of work because of enduser computing. Indeed, the systems area has continued to grow. Whether systems would have had more work to do if there were not PCs is difficult to say, but much of the work being done on PCs at the bank would never have been done if it had had to go through the normal systems development channels.

All of this points to the conclusion that there is presently no basis for claiming that a reduction in skill requirements is found in the bank's systems area. Further research would be needed even to test whether there has been a drop in the technical skill levels of applications programmers. When the increase in emphasis on analysis and communication skills is added into the equation, the net effect on programmer skill over time is very difficult to guage.

Invisible or hidden skills
The stereotype of data processing departments in the financial services sector as "sweatshops" misses the extent to which very real, non-technical skills are in use in these work settings. How is it possible that these skills are overlooked by some members of the larger software community? Why are some skills better recognized and more highly rewarded than others? One possible answer to these questions is that certain skills become "hidden" or "invisible."

Braverman himself focuses on skills which involve operations on objects to the exclusion of other sorts of skills. He does not consider the ability to communicate or to interact with other people as skills necessary on the job. Even when writing about service occupations, Braverman uses a manufacturing model, treating only those aspects of service work which require the manipulation of the physical world. Customer service, sales, and other tasks which require personal interaction never appear in his model. As a result, he misses entirely the existence of phenomena such as the shift away from technical preoccupation at the bank, and can dismiss the work of managers as geared entirely to control, rather than seeing that they also serve a

2 The fact that Braverman's model is so grounded in Taylor's work may account for this error. Of course the ability to communicate and to work with others are important in a manufacturing setting, too, but they may not be as central as they are in the service sector. In stressing the similarity of office jobs to production jobs in the manufacturing sector, Braverman has overlooked some of the genuine differences between the two settings.

3 Frederick Taylor makes the same error when he claims
communication and negotiation function within the organization.

Examples abound of skills which go unrecognized while others are rewarded. Kusterer (1978) is widely known for making the case that even so-called "unskilled" workers have extensive working knowledge on which the smooth running of the firm depends. These skills may be unrecognized and unrewarded by managers, but this does not mean that are not important to the operation of the business. Lucy Suchman suggests that in office settings, even when there are detailed procedures which workers are told to follow, fitting the real world to the procedures is not a straightforward matter (1983). Likewise, she says, the instructions for seemingly simple office machines such as copiers can never be complete and unambiguous, and their users must supply the missing knowledge or information. In both cases, knowledge and judgement are required to perform what might at a distance appear to be routine work.

Braverman himself launches an impressive critique of the classification of occupations with respect to skill established by the U. S. Bureau of the Census in the 1930s. Operatives, who work with machinery, are taken in this system to be more skilled than laborers, regardless of the precise content of their work. "It is only in the world of census statistics," says Braverman, "...that an assembly line worker is presumed to have greater skill than a fisherman or oysterman, the forklift operator greater skill
than the gardener or groundskeeper, the machine feeder greater skill than the longshoreman, the parking lot attendant greater skill than the lumberman or raftsman." (p. 430)5

To demonstrate that there are skills which are hidden or invisible in particular employment situations is not to explain how or why these skills become hidden. Rubery suggests that literacy is no longer counted as a job-related skill because it is thought to be universal in the population and is therefore "fails to differentiate workers or provide them with a basis for bargaining." (Wood, 1982, p. 19fn.) Other skills, like the skill of driving a car, could be thought of in almost the same way. In general, skills for which no wage premium is necessary may go unrecognized. This phenomenon may particularly affect women and minority workers, who have more limited job opportunities and who are generally found in the lower-paying jobs.

Skills may also become hidden because other skills have been the focus of great attention. Research in the sociology of occupations shows that occupation-based organizations, from craft unions to professional organizations, act to maintain and increase members' incomes by identifying and protecting a set of skills thought to be characteristic of the occupation. Skills which are not thus

3 Frederick Taylor makes the same error when he claims that his system represents upgrading for all. The "green hands" whom Taylor saw hired in were likely to have been farmers in their country of origin.
protected may, by contrast, come to appear not to be skills at all.

Why is skill important?

The literature on Braverman has focused more heavily on his predictions about skill than on other aspects of his model. The term "deskilling" has become synonymous with Braverman in the minds of many social scientists. Skill, and its increase and decrease under a variety of circumstances, have become hotly debated issues not only in the academic community but among policymakers and in some groups of workers as well. Despite its popularity as a subject of investigation and controversy, no satisfactory definition of skill, in the broad craft sense in which Braverman uses it, has ever been offered. The problem of measuring and comparing skill is even greater. Nevertheless, if progress is ever to be made in the debate over whether skill demands are rising or falling, these problems of definition and measurement must be solved or circumvented.

The problem of defining and measuring skill seems virtually insurmountable, at least in the terms in which the present debate is framed; what follows, then, is a proposal for circumventing, not solving, this knotty analytical problem. Perhaps if we were to ask ourselves why skill was important, why we cared about it so deeply, we could link it with another variable that might admit of easier definition
and measurement. To begin this process, here is a list of possible reasons why skill might be important:

- Using skill gives intrinsic satisfaction
- Workers with skill have greater bargaining power, leading to increased wages
- Skill puts workers at the center of the labor process and allows them a sphere of control
- Maintaining and expanding skills increases the capability of the national/world population

If any of these reasons is central to the concern about skill, it becomes possible to pose questions about the fluctuation of something other than that intractable quantity, skill. If we are concerned about workers' psychological wellbeing, or about their ability to command high wages, those factors are easier to measure than skill.

The disjunction of skill and organizational power.

Braverman, in his discussion of the polarization of skill, equates the possession of job-related skill with organizational power. The evidence from the systems area of the bank shows that this equation can be misleading and sometimes even mistaken.

In Braverman's framework, the polarization of skill leads to the organizational and social domination of the less skilled majority by the highly skilled minority. The minority not only control production knowledge, but also
hold the reins of organizational power and command high salaries. Thus there is complete congruity among all three dimensions—skill, income, and organizational power.

At the bank, skill does not necessarily translate into organizational rank. Particularly in the case of technical skill, the most technically able are often not in the positions of highest status in the managerial hierarchy. Indeed, bank management has moved explicitly to separate skill and organizational rank because they found that, as in many other firms, skilled specialists do not necessarily want to be managers, nor do they always make good managers. Since at least early 1986, and perhaps earlier, the bank has had dual career ladders which permit those with specialized skills to be promoted along a "professional" career path distinct from the "managerial" path taken by those who become supervisors of the work of others. People with technical expertise may follow either path, but some of the most skilled technical experts take the professional track. They retain the authority and influence that is born of their technical expertise, but they do not have managerial authority over others in the organization.

Although the status of systems work has increased at the bank over the years, top management is still not drawn

---

4 Kraft and Dubnoff concluded from a study of more than 650 programmers in the Boston area that "the key to financial success in software is not to specialize and not to get too technical." (1986, p.192) To what extent this is true at the bank is unclear, but it appears to be the norm in the labor market in one large metropolitan area.
from the systems ranks. Certainly those who become chairman or president at the bank have skills, but it is not likely that these skills are technical or even specialized skills. What, then is the relationship between skill and organizational power?

To answer this question requires further study, but this research suggests that in this case there may be three dimensions of skill/authority. First, technical expertise; second, business expertise; and finally, managerial rank or authority. How these three variables are correlated remains to be seen, but evidence from the bank suggests that the connection between technical expertise and managerial rank is not absolute.
CHAPTER SIX
CONCLUSION AND SPECULATION

Harry Braverman has predicted that under modern capitalist conditions the mass of jobs will become degraded: fragmentation and a detail division of labor, the separation of conception from execution, and the destruction of craft skill with a polarization in skill requirements will prevail. Both Braverman himself and his followers, Philip Kraft and Joan Greenbaum, have applied this prediction to the work of computer programmers and have concluded that programming, too, will follow the course of other occupations. The evidence from this study, however, suggests that Braverman's predictions for programming do not hold true, at least in one large commercial bank. Where Braverman, Kraft, and Greenbaum predict fragmentation, this study finds a division of labor which undergoes considerable modification from project to project but which never relegates programmers solely to the position of detail workers. Where they predict the separation of conception and execution, instead it appears that coding and analysis are generally not separated, and that no one spends all their time simply writing code. Where they predict reduction in skill requirements for most jobs, a shift in skill requirements--emphasizing analysis and communication and de-emphasizing coding--is found.
These findings do not by themselves disprove Braverman’s theory about the transformation of work. In the recent history of industrialization there are many instances of the transformation of work following the pattern laid out by Braverman. The automobile assembly line is certainly the classic example, but there is evidence from other occupations--clerical work and printing, for example--as well.1 This study does, however, call into question the applicability of Braverman’s theory to programming work in financial services, to programming work in general, and even perhaps to work in all non-production settings.2

As with any set of observations based on a single case, the representativeness of the case and the generalizeability of the findings drawn from it must be questioned. Among the many avenues for further research suggested by this study, the generalizeability of these findings is certainly among the central issues. Is this bank an anomaly? Does systems work have parallels in other occupations? What do these findings suggest for the future of work in advanced industrial society?

The goal of this study has been to ask these questions, not to answer them. A single-case approach was chosen as a vehicle for questioning one compelling set of theories about the future of work, and as a means for mapping the terrain

1 See, for example, Evelyn Glenn and Roslyn Feldberg (1979); Andrew Zimbalist (1979); Maarten de Kadt (1979).
2 For a discussion of Andrew Friedman’s notion of Responsible Rutonomy applied to programmers, see Appendix 4.
within which further research might take place. The remaining section of this final chapter is devoted to framing a set of questions about the study’s implications and to speculating about those implications.

**Can the findings be extended to other parts of the bank?**

Programmers have a role at the bank that is different from that of many of the bank’s other employees, suggesting that the conditions which have prevented the fragmentation and routinization of systems work may not apply to other employees at the bank. A brief discussion of the role of systems people at the bank may clarify some of the differences between systems people and many of those on the business side of the bank, especially clerical workers.

From a structural point of view, systems people serve the same function in the corporation as Frederick Taylor’s engineers. They are not single-mindedly devoted to the removal of working knowledge from the shop floor, as Taylor himself was, but it is their job to study and to automate many of the functions performed at the bank. They are in charge of some of the aspects of the "design, planning, calculation, and record-keeping" which Braverman says were the province of the office staff under Taylor’s system (1974, p. 124).

Although their role in the bank thus places systems people in Braverman’s upper tier in the skill polarization, Braverman says that this privileged status does not protect them from the processes of occupational degradation which he
describes. Many office functions have already been routinized, he points out, transforming them into work virtually indistinguishable from that on an assembly line. He was already asserting in 1974 that much of programming was following suit, becoming more simple and routine.3

Again, the findings of this study do not confirm Braverman's predictions. Rather than reject the notion that programmers stand in the same relation to the production process as did Taylor's engineers, however, it seems more likely that there are some limits to the routinization of engineering-like work which Braverman did not take into account. These limits are discussed in the section below, which reviews the analogy to engineering and design workers.

The evidence from interviews raised other questions about the role of the systems organization in the bank. Most systems people described systems as a "service," while a few also suggested the possibility of a leadership role. George Mitchell, an economist and former governor of the Federal Reserve Board, describes a bank as "a counting house."4 If this were true, the systems organization would

---

3 This raises the question of what it means to be in Braverman's "upper tier." Does it signify standing in a particular relationship to the production process, as those who study and automate the work of others would? Or does it mean that one has escaped the downward pressure on skill as the skill structure polarized, and thus that one retains membership in the class of highly skilled? Or does it reflect a difference in organizational power, such that those in the upper tier direct the work of those in the lower? Braverman seems to equate all of these, yet there are certainly cases in which these three dimensions are not fully congruent.

4 Quoted in Martin Mayer, (1984, p. 6).
be synonymous with the bank, and the rest of the employees could be discharged. There are two reasons why this will not happen. First, the complexity of the systems necessary to run a large diversified commercial bank would be so great as to defeat all foreseeable efforts at full automation. Second, and more important, the bank, especially in today's deregulated environment, does far more than count. In particular, the formulation of a competitive business strategy for the "strategically managed, numbers-driven bank" is key to its success, and this function is far from automatable. What role computer systems and the systems organization will come to play in strategic decisionmaking remains to be seen.

Can the findings be extended to other banks?

Without investigating the practices of other banks, it is difficult to say to what extent this bank is typical of firms in its industry. One piece of evidence suggests that this bank may be unusually backward in its systems area--at least with respect to the use of Management Information Systems--while several other factors suggest that the bank is at least not a major outlier in the banking population.

According to a manager in one user area, the consultants from the large consulting firm which studied the bank as part of the strategic planning process were highly critical of some parts of the systems area and gave it a low rating compared to other banks for its performance in the

6 On the other hand, one could argue that under
MIS area. In one respect, then, this bank is an extreme case and not a typical organization. On the other hand, a Salomon Brothers report on technology in banking cites the bank as being in the top third of the 35 banks it follows with respect to its use of "high technology" (Hanley et al., 1985).

In a general way, the bank is unlikely to be entirely offbeat in its organizational style: it is a very large institution, it can be conservative and slow-moving, it uses the standard big consulting firms for its strategic planning effort, and it is not one of the very small number of banks (such as Citibank) known for its innovation in the systems arena. All of these factors suggest that while the bank may not be typical, it also does not represent a total departure from received practice in the industry. Furthermore, to the extent that the decisions about how to organize work are governed by factors outside the bank's direct control (such as the state of technology, the generic nature of programming work, and the competitive conditions in the industry) it has been affected by the same influences to which all other banks are subject. These influences could tend to produce similar responses.6

---
6 On the other hand, one could argue that under competitive circumstances advantage will accrue to the innovators who take a novel path. The firm of Morgan Stanley, for example, decided to build its entire systems operation around top-notch liberal arts graduates using fourth-generation languages. This strategy is reported to have been very successful for the company, partly because no competitors are doing anything this novel and as a result the firm is able to pick the cream of the graduating crop.
Can the findings be extended to other systems workplaces?

The bank was chosen as the site for a test of the applicability of Braverman's theory to programmers because the conventional wisdom holds that systems work in banks and insurance companies is among the most routine. For this reason, a financial services company seemed the best place to test the limits to the routinization of programming. While systems work at the bank seems to be less technically challenging than that at state-of-the-art high technology workplaces, the variety of the work and the elements of the job which require analysis and communication save it from being routine.

Does this mean that it would be impossible to find elsewhere a more routine systems workplace? The evidence from the bank suggests that the routinization of systems work is most possible in workplaces at which:

1) The product is relatively standardized and conceptually straightforward.

2) The projects are large with extensive division of labor and top-down control of the development process.

3) There is little or no contact between programmers and users.

4) Managers see programming as routine work amenable to assembly-line management techniques.

5) Labor market constraints do not create problems attracting and keeping qualified workers.

In scientific and research settings there may be extensive division of labor, but the product may be
relatively unstandardized, which could limit the level of routinization. In business settings, as at the bank, there may be increasing use of software purchased from outside vendors. This would decrease the need of any given company for a large pool of applications programmers, but it would ensure that those applications programmers who remained were heavily occupied with analysis and implementation tasks. Programming work performed for software vendors has the potential to be comparatively routine, since projects may be large and contact with users scarce. Whether this is true in fact is a matter for investigation.

One-person projects

Among those who write about the management of software projects there is a school of thought with interesting implications for the future organization of software production work. Members of this school suggest that the single most important productivity increase in systems development comes from reducing the size of the development project. The ultimate goal is to make as many projects as possible into one-person projects.

Frederick Brooks, a prominent software engineer and the project manager for the IBM System/360, wrote in a 1975 book on software development that "the sheer number of minds to be coordinated affects the cost of the effort, for a major part of the cost is communication and correcting the ill effects of miscommunication (system debugging). This...suggests that one wants the system to be built by as
few minds as possible." (1975, p. 30). T. Capers Jones, an expert on productivity in software development, argues similarly that the best thing that a manager can do to improve productivity is to make a project a one-person project (1986).

When a project cannot be reduced to the scale necessary to make it a one person project, Brooks suggests that the key is to reduce the amount of communication needed while maintaining the "conceptual integrity" of the product. To this end he suggests various techniques, including the establishment of "chief programmer teams" organized on the model of the surgical team. He believes that a hierarchical arrangement coupled with a well-defined division of labor is the best way to reduce the expense of communication and the risk of miscommunication. This recommendation can certainly be read as an endorsement of fragmentation and a detail division of labor, but such a reading overstates the case, and misses entirely Brooks' endorsement of project teams of the minimum necessary size.

Can the findings be extended to other design and engineering work?

Why has standardized, routine work developed in some workplaces and occupations, but not in programming at the bank? Assembly-line-like programming work has not developed at the bank, nor does it show any sign that it will in the future. The presence of the assembly-line in manufacturing and clerical work but not in programming may be explained by
distinguishing between two different phases of the production process: development and reproduction.

The development phase of production comprises the design work and strategic and tactical decisionmaking that are the prerequisites for manufacturing or clerical production. In the development phase the product is designed, the policies set, and the infrastructure constructed which allow the actual production of a product or service. Reproduction is the phase which follows the development process, during which the actual products and processes conceived during the development phase are brought into being. In the production of automobiles, for example, the work of the engineers who design the car and the production process is part of the development phase, while the shop floor workers who manufacture and assemble the autos are reproducing the product design over and over as they perform their jobs.

Programmers are like design engineers, not like automobile production workers. They are not involved in the reproduction of a product; instead, they are designing programs which remain as one-of-a-kind objects. 7

7 The production of software has the character of a development process regardless of whether the software is intended for a company’s internal use or for reproduction for sale as a commodity in the marketplace. The matter of whether the program will be reproduced or not may have only minor effects on the nature of programming work in a given setting. The size of the market for a particular program—be it a market of only one customer or a market of thousands—has no systematic effect on the nature of programming
Programmers and engineers must both develop specifications and detailed designs, manipulating information in order to come up with a final design—a program, in the case of programmers; a blueprint in the case of engineers. 8

There is always a significant element of "custom" work in each programming project. Although there are similarities among programs, no two are ever identical. Instead, programs exhibit a "family resemblance" in that they may contain common elements, use a common language, and share similar purposes. Because of these common elements, tools can be developed which expedite the programming process. While these tools may simplify the technical aspects of programming work, they do not routinize work, since it is development work and not reproduction.

The fact that the organization of programming work is independent of the size of the market for the software being developed seems to contradict Adam Smith's assertion that "the division of labour is limited by the extent of the market." (Smith, 1776/1979, p. 121) In systems development, the most widely circulated software may be written by a single person, while program developed for a single application may require the work of a very large team. Smith's aphorism may hold true for reproduction work, but it does not hold for development work. In development work, in effect, the size of the batch produced is always one, regardless of the size of the market.

8 There are some dissimilarities between a program and a blueprint in that the program is executable without human intervention and the blueprint is not. The dream of the automatic factory rests on the notion that it is possible to make a blueprint as easily executable as a computer program, but this dream is far from being a reality. The intractibility of the physical world and the complexity of most manufacturing processes place an enormous stumbling block in the path of the automatic factory. The computer, in which programs are executed, is by contrast extremely tractable.
programming itself. Even at a lower level of technical complexity, programming still requires analytical ability, insight, and the capacity to respond to a variety of sometimes-ambiguous situations. The judgement of the programmer is still an essential element of the systems development process. Tools that reduce the technical complexity of the work have not led, at the bank, to the routinization of programming, but rather to a change in the job content of programmers.

"The Henry Ford of software hasn't come forth yet," quotes Greenbaum from an article in Computerworld (Greenbaum, 1979, p. 105). Nor will he ever come. The conditions set by Kraft for the routinization of work—"an unvarying good or service and... an unvarying way of turning it out" (1977, p. 52)—are not met. Fordism will never make it to the programming shop.

One of the consequences of the fact that programming is development and not reproduction work is that Direct Control is impossible with programmers, but not because of worker resistance. Programmers are indeed very likely to resist Direct Control (as both Kraft and Greenbaum document), but because of the nature of their work its application would be inappropriate even in the absence of resistance. The development process is not predictable enough, nor its productivity sufficiently measurable, to have Direct Control make sense as a strategy.
What does this tell us about Braverman’s theory? Braverman writes about work situations in which people perform potentially routinizeable operations on things. That is, he writes about the domain of reproduction, not the domain of development work. Design tasks, which deal with the collection, alteration, and transmission of information, are part of the upper tier, the planning function, and therefore not the primary objects of his scrutiny.

Can the findings be extended to other work in general?

The variety of work roles and settings is so vast in industrial society that it is risky, although tempting, to generalize to all jobs from the findings on a single set of activities, let alone a single workplace. The commonalities between programming and other sorts of work wait to be established through other detailed case studies. In time, it may be possible to develop a typology of labor processes, including those in factories and beauty parlors, in hospitals and amusement parks and automobile dealerships.

This study is intended as a contribution to the task of classifying modern work and its properties. Investigation at this level of detail seems necessary given the limited state of our knowledge about the varied work settings in which people today may be found. With luck, some of the findings of this study will be among the building blocks from which a typology of work can be constructed. The distinction between development and reproduction work, for example, forms the basis for a classification of work. But
is machine maintenance work development or reproduction? And what of sales and marketing jobs? Under what heading would they fall? The task of developing such a typology is a major one, but would also be interesting and revealing.
APPENDIX 1. SOME PROJECTS FROM THE SYSTEMS AREA

a) Corporate Banking System (CBS)

The Corporate Banking System stores and compiles information on the bank's corporate customers, providing different levels of information to professional and managerial employees of the bank. Loan Officers may use the system to track individual clients, getting information on such things as the bank's total loan "exposure" to a particular client. Other officers may use CBS customer data to determine which customers might profitably be sold other bank "products" in addition to those they already use. At the top of the organization, senior management may use data from the system, presented in aggregate form, to make strategic decisions affecting the entire organization.

CBS is an on-line system which can be used interactively from a computer terminal, but it is also possible to get printouts on paper of information in the system. Loan officers have access to the system, but cannot change some of the data on it. An officer might change the address of a company, but for all changes that affect the accounting system of the bank, changes must be made by going through the proper channels for altering accounting data.

Of all the computer systems in the bank, this one is said to have the greatest effect on loan officers. "It's like someone standing over them," said one person who worked on the development and implementation of the system.
When the CBS system was developed, the system's requirements were generated by senior management at the bank. The project was overseen by the MISD group, and two systems people from MISD wrote the specifications for the system. One member of MISD described the intense effort of developing the system in a short period of time with six or eight "all-star" systems people as "one of the best professional times of my life." The systems development process was novel for the bank because for the first time true "endusers" were involved in the development on an ongoing basis. In earlier systems development efforts, the users involved would be those from the "operations" area of a user department, not necessarily those who would use the system's output. In addition to acting as internal consultants on the system's use and design, the user group designed the displays as they would appear on the terminals when the system was operating, deciding on what information would be shown and how.

The system was put into "production" in late summer or early fall of 1984. A year later, a member of MISD said they were still doing a lot of work on the data going into the system, as well as making changes in the system itself. "I've been cleaning up bugs for almost a year now," she said. Among other things, getting the departments to send reliable, accurate data promptly had proven a major headache. Often departments have their data in a form other than that needed for input to the CBS system, and those
departments have other things to do that interfere with the change to a compatible data-keeping system. There were other reports of system implementation problems as well, including a suggestion that there was insufficient training and support for officers and others who were now being asked or required to use the system.

b) Customer Management System

The CMS is a decision support system created for account officers. Its purpose is to provide them with information that will help them manage the relationships between the different market segments (upscale, mass market, etc.) It does not provide information about every individual account.

New England Systems Information Center created CMS because the standard bank teller terminals were inadequate to provide this sort of information. The system was put together quickly—it is essentially a prototype—and it had to be available in the bank branches, which is one of the reasons why it is not PC based.

c) Massachusetts Consolidation Project

One of the consequences of the deregulation of banking is an increased rate of purchase and merger involving two or more banks. The bank has joined forces in this way with several other banks in the region. Once joined, the banks have to start using the same computerized systems, so that the systems of one or both banks have to change. The systems people at the bank have participated in many "conversions." In one year, there were 87 conversions.
These conversions are not technically difficult, according to one manager in the New England systems area, but there are lots of details to look after. Also, the interpersonal dimension of the conversion process—the relationship between the personnel of the two banks that are being joined—are very important.

d) Relationship Planning System

The Relationship Planning System is designed for use by loan officers. It requires officers to put a marketing plan aimed at a particular customer into writing, including numerical targets. The system then makes it possible to determine over time whether a particular customer is profitable to the bank or not. It also makes it possible to track the performance of loan officers, and gives "the people on the second floor" (that is, top management) information about loan officer performance that they didn't have before.

e) Commercial Loan System

This system generated written reports for use by loan officers.

f) Strategic Planning Project

During the bank's three-year strategic planning process, a small group of relatively senior systems people was assembled to provide information to aid the strategic planning process. They reported to an executive vice president. According to a systems person who had been a member of the team, they worked under extreme time pressure to "take the entire general ledger and align it according to business units."
For this effort, which was really a major management information reporting effort rather than a project to develop a lasting computer system, they chose to use a fourth-generation language called "Stratagem," which used very compact code. It was so compact that it could be difficult to read. It also creates a system that is not robust enough to run day in and day out.

The workload for the half-dozen or so programmers who were the staff of the project was very heavy, but could also be maddeningly intermittent, according to one team member. "It was either 110 hours per week or you were bored silly." The bank reserved rooms in a nearby hotel for the team members so that they could be at work for a maximum number of hours and not lose work time commuting. The pressure was such that "sometimes the programmers would cry in the middle of the night," according to one participant.

The project could also be frustrating, said one member, because "you were always fielding requests." It was satisfying to be part of an important effort, he said, but team members were frequently asked to redo things they had written, sometimes rewriting them several times. Yet there was also the pleasure of working with a small group of other senior people.

g) Monec System

The Monec system allows the bank’s retail customers to make a limited number of standardized banking transactions using an automatic teller machine (ATM). The system was developed in the early 1980s for use by the bank and its
correspondent banks, and also as a system that could be sold by the bank to other banks for their own use.

The development team for the system consisted of six people from the bank, eight programmers hired on a temporary contract basis from outside the bank, and several people representing the "user" area. The project was based at a site remote from the bank's downtown headquarters, at the bank’s Data Center in a nearby suburb.

The project leader said that he believed it was important to take "rigid control" of the project from the beginning. He was particularly concerned with imposing structure on the design and coding effort from the beginning because he felt that contract programmers write code without regard for how easy or difficult the system will be to maintain in the future. The project leader and two others laid out the entire architecture of the system beforehand. During the project, they had design reviews.

The project leader feels that the system is a great success from a technical point of view. He reports that the system is rarely "down" and that it has been relatively easy to expand the system's capabilities—to support more varieties of ATMs, for example—and offers such versatility and maintainability as evidence that the system was well conceived and well executed from the beginning. He also notes, however, that the system has not been much of a success from a marketing point of view.
h) Commercial Loan System

The system for the commercial loan area was finally put in place in August/September 1984. The data from the system are used by three loan processing units and by the Finance area. Both the Corporate group and the New England group use the system.

According to an analyst from the CBDG group who worked on the project, it had taken ten years of talking about what they wanted and what would work to get the system to the point of installation. In the later stages, she and two others from the CBDG worked for a year with three users and three systems people to hammer out some of the details of the system's design. The analyst who described the project said she worked on designing the specifications for the 60 or so reports the system was to generate; another woman designed the displays as they would appear on the screens of the terminal. According to the analyst, it was a year of vigorous negotiation and asking "what if...?" They would repeatedly have meetings where they went in a room and "pounded heads." After they were done it was coded by others.

A programmer trainee was assigned to the Commercial Loan team (CLN) in the systems area after eight weeks of training. Two weeks after she joined the team, the new commercial loan system went in, replacing a system that had been in place for 15 years. "I fought fires and did little things for the first month." Then her team was reorganized,
split into two, and the more senior group was put in charge of enhancements to the system, which she referred to as "the more interesting work." In the fourteen months since the program was installed, she has worked on maintenance, including overseeing the production of 130 reports regularly generated by the system.

A year after the installation of the system, members of the CBDG team were still at work on the system. They were working on debugging the system, testing new functions, and helping with the conversion of correspondent banks to the system.

i) Performance Reporting System (PRS)

The Performance Reporting System (PRS) was developed to track budgets and expenses throughout the bank. The tracking is done not for individuals but for collections of individuals under a single manager. These collections are known as "responsibility centers." The strategic planning effort, which lead to the announcement that the bank would become a "numbers-driven" company, was the catalyst for the development of the system.

The system keeps records of the budget for each responsibility center, and can generate reports comparing budgeted figures with actual income and expenditures. On a monthly basis, the system generates such reports, and there are quarterly meetings of senior managers in which they are asked to explain discrepancies. This information can also be used for forecasting and for developing budgets for upcoming years.
The system was developed by MISD, which learned in August of 1984 that they were to build such a system. MISD bought a software package called "System W" to use in writing the PRS. In part because System W allowed quick development, the PRS was up and running the following January. Seven months later, a member of MISD reported that there were still problems with the system, but that the reports were in use. Problems included trouble with the data in the system, which have to reconcile with those in the general ledger. There was still, seven months after the system started running, much manual input of data and some "fudging" of numbers, according to a senior analyst at MISD. Also, System W runs very slowly—one entire cycle takes at least 18 hours—and cannot feasibly be run every day, although management would like it to. MISD is still involved in debugging and enhancement, including the merger of some of the input processes for PRS and CBS, since they use some of the same data.

j) "FRS"

One programmer trainee was assigned to do a project by herself when she first completed the training program. The project (FRS) consisted of installing a purchased software package which had been bought one or two years before and which no one had had time to install. The trainee did the entire project herself, "soup to nuts," with help from others in systems and the user areas, especially from the more senior "mentor" to whom she was assigned. She reports that there was a lot of analysis work involved—not the sort
of analysis that determines whether the system meets user
needs, since that had already been done when the package was
purchased--but "analysis in terms of DP considerations." In
the process of the project, in addition to working alone,
she met with a representative of the software house that had
sold the system to the bank, she met with users, and she did
a lot of training of users who would be working with the
system when it was installed. After installation there were
a lot of "finishing touches" to be taken care of. In
addition to her other assignments, she continued to maintain
this system for a year, until the maintenance responsibilify
moved to another area with bank decentralization. Even now
she thinks of the system as "her baby," she says, and when
she gets the occasional telephone call about it, she can
easily get back into the details of the system. Some
larger, more recent projects she finds more difficult to
remember.
Figure 1.21 Annotated EMP-LIST program

APPENDIX 3. THE VARIETY OF SYSTEMS SETTINGS

Systems people all work within the systems area, but in a variety of different settings. The variation in these settings accounts for a large extent for the variety of different conditions within which systems people work at the bank. It would be misleading to treat systems work as generic and uniform, since in fact two people, both working in systems at the bank, may work on very different things and under strikingly different conditions.

(1) Advanced Technology Group

The Advanced Technology Group (ATG) is a small group of ten to twelve high-level systems people who have as their mission the development of PC-based computer systems for use by professionals at the bank. They are trying to increase the bank's gross annual income by focusing their attention on aiding the work of "income-producing" professionals--"Someone who turns a buck for the bank, instead of someone that counts them, like me," said one member of the ATG. The bank had a significant and growing investment in PCs when the group was founded officially in January 1984, and the work of the ATG is intended to "leverage" that investment in hardware, to make the existing capital equipment more useful and cost-effective. Thus the ATG concerns itself entirely with the selection and packaging of software, taking the existing hardware as given. The chief job of the members of the ATG, however, is the "marketing" of software to
professionals at the bank. Most of their time is spent in analyzing, persuading, and implementing.

The ATG forms relationships with individual people on the user side of the bank and packages a system designed to help solve a particular genre of business problem. Members of the ATG may go out trying to sell a particular system they’ve developed, or they may respond to a user request. In either case, they do not write software themselves except when necessary to tie together two or more pieces of purchased technology. For example, if they have a request for a sales tracking package, they may select one from among those commercially available, integrate it with a dialing package, then offer customized versions of it to other areas of the bank in addition to the one originally requesting it. "What we will do eventually is lay a control layer on top of Lotus and Expert Choice and all that...I’m not interested in rewriting the spreadsheets but I’m very interested in making the spreadsheets dance."

The ATG works with users at no charge, and development of a system is an iterative process. They try to respond quickly to requests, coming back to the user within a week or two with something to show them—which may be "only 5% of what they want." By getting the user’s reaction, they can get a better sense of how to proceed with the development effort. "Prototyping is a way of life here," says one group member. "Our systems are always in prototype stage," says another. This contrasts with the more traditional systems
approach in the areas that deal with the large "factory" systems. There prototyping is done only occasionally, and even then only the "presentation layer"--the part that the user will see.

A member of the ATG who had spent a decade and a half in the traditional systems area before coming to the ATG contrasted the two work experiences.

The role down here is completely different. Much more loosely structured. More entrepreneurial. Rather than living by a Gantt chart of tasks that have to be accomplished by a certain time, you've got 6 or 8 months to provide some benefit to some area of the bank by applying technology to it. You're left pretty free to do that any way you choose. Let me tell you, at first it's frightening, coming out of a structured environment. I didn't anticipate that coming down here would bother me, or how much of my life had been dictated for me without my even realizing it because I was a member of a large team with specific goals, fairly well regimented in terms of task-oriented detail. Down here, you have to discover the opportunities, market them, put the packages together to decide what they need, convince them that they need it, persuade them to pay the price, and then monitor it to see if you've really helped them, not just helped IBM's bottom line.

The members of the ATG are deliberately attempting to take an approach different from that of the main systems area out of which they all came--different not only in the sorts of systems they build and how they build them, but also in the structure of the group itself. Although on an organization chart the dozen or so members of the group are arrayed in a "typical pyramid," according to one manager, they attempt to conduct themselves as a group without the usual hierarchy. They say they stress coordination, not supervision, and that that is possible because they are all
experienced, relatively senior people. The ATG, said one member,

...is much more a confederation of individual contributors than it is a finely tuned orchestra. The nice thing is that we have no interest in being a finely tuned orchestra. We would rather be a jam session of good jazz players all recognised for their individual talents that only comes together once or twice in a lifetime and then they go their separate ways.

(2) CBDG

The Corporate Banking Development Group is a small group of about 3 people who act as fulltime analysts and interpreters. In its previous incarnation, CBDG did financial analysis work and other sorts of studies. The members of CBDG did not become involved in systems until around early 1983, when they got involved with the CLN system "and we got too busy to do other things." Now the members of CBDG, who are nonprogramming analysts, spend most of their time on systems work. All the members of the CBDG are women, except for one man, a programmer, who at the time was assigned to CBDG and was learning about their work.

(3) MISD

Management Information Systems Development is a group of about 30 devoted to Management Information Systems projects. The systems people at MISD must, collectively, understand the business of the entire bank, since they build systems which collect data bankwide. Users assist the systems people on particular projects--in August, 1985 there were ten users assigned to MISD--but the systems people also develop their own expertise on banking matters. MISD
continues to be involved in the implementation and maintenance of the two systems they have designed since their inception, but one analyst in MISD said that the implementation and maintenance work was being done, on the whole, by lower level people at MISD, and could be split off.

(4) CBGSD

The Corporate Group Systems Division (CBGSD) is a large organization, paired with the Corporate Group. At the time of decentralization at the bank, when the five major business-oriented groups were formed, the systems area was also broken up into groups. The CBGSD was created at that time, to serve the information processing needs of the Corporate Group. The head of the CBGSD reports directly to the head of the Corporate Group, and the people at lower levels of the systems area and the business side of the bank interact with each other in the course of their work. The CBGSD is in charge of development and maintenance of the large transaction-based systems needed by the Corporate Group, but also handles the reporting needs and the smaller systems requirements of their client group, the Corporate Group. The Corporate Group handles the banking needs of corporate clients, and as a result the information gathered is on corporate clients, not on individuals.

(5) NEIS

New England Information Systems is the systems organization paired with the bank’s New England group. The
New England group handles all retail banking work for the bank. It keeps customers' demand and time deposits, offers credit services, and provides other services to individuals. The New England Group is the largest in the bank, both in terms of the size of its staff and in terms of the size of its customer base. New England Information Systems oversees the systems needs of this group, including services like automatic teller machines.
APPENDIX 4. ANDREW FRIEDMAN'S THEORY OF RESPONSIBLE AUTONOMY

The most influential attempt to rescue Braverman from the incongruity of the empirical evidence offered in this and other studies has been made by sociologist Andrew Friedman. Friedman's argument, discussed in Chapter 2, is that capitalists use two different strategies, not just one, to maintain control of the labor force. In addition to the Taylorist strategy, which Friedman calls "Direct Control," there is a second strategy, "Responsible Autonomy," which give workers a measure of authority and responsibility in the interest of more effective long-term control.

Although it seems at first counterintuitive to give workers increased autonomy and responsibility in order to increase overall control of production, Friedman explains that a variety of factors led capitalists to appropriate a bowdlerized version of the traditional craft worker's ethos as a means to maintain control. Since the 1920s, he says, managers had been searching for solutions to the problem of control of workers whose initiative and creativity were valuable to the production process. As labor markets tightened, giving workers more power; as worker resistance to Direct Control became stronger; and as more highly integrated production technologies expanded the impact of sabotage, a Responsible Autonomy strategy became increasingly attractive to management.

As Friedman asserts, Responsible Autonomy under these conditions may actually strengthen managerial control of
production although it loosens detailed control of the labor process. At Volvo, for example, he argues that the removal of the assembly line and the granting of moment-to-moment control over activity to assembly workers actually increased managers' overall control of production by decreasing worker resistance (expressed in absenteeism and labor turnover.)

A Responsible Autonomy strategy does not, however, change the underlying structure of capitalist relations. Responsible Autonomy may soften the effects of alienation and exploitation, but it does not remove them. Production continues to reflect the needs and will of management, not of the workforce. Responsible Autonomy, in Friedman's view, is not a case of management ceding its power to the workforce; it is instead a strategy for maintaining control in situations where the exercise of Direct Control is likely to backfire. Thus Responsible Autonomy is most likely to be found among segments of the workforce such as craft and professional workers whose market power is greatest and whose initiative is most needed.

Friedman's formulation of Responsible Autonomy is useful because it highlights the fact that control can still be exerted without resorting to the coercive regimentation of Direct Control. The practical significance of Friedman's revision of Braverman is unclear, however. The characteristics of Responsible Autonomy are described in vague terms, and in an empirical sense it is dangerously close to being simply "the absence of Direct Control." That
is, if Direct Control is not used by capitalists, then Responsible Autonomy will be. The risk here is that of creating a tautology: if the workplace is a capitalist one and Direct Control is not in use, then Responsible Autonomy must be. That is, if one were to ask whether Responsible Autonomy were in effect in the systems area at the bank, and the answer was that it must be since the bank is a capitalist organization and the systems area is not subject to direct control, it is not clear that one has learned anything from the inquiry. The existence of Direct Control and Responsible Autonomy is thus proven "by definition"—as something inherent in the definition of capitalism. Within Friedman’s framework, to argue that neither of these strategies correctly described what was going on within a capitalist organization would be impossible.

At the bank, it appears that control of systems people and of other professional and managerial employees is achieved largely by winning their commitment to managerial goals. Along with that commitment go an identification with the organization and a devotion to its success in the marketplace. While other workers, possibly including those who hold the many routine clerical jobs at the bank itself, may be subject to Direct Control, the corporation achieves loyalty among its higher-level employees by getting them involved and invested in the bank’s business objectives.

As the systems people become more like their counterparts on the business side of the bank, they clearly
have become more involved and invested in business issues. As a consequence, they also become increasingly subject to the sort of control which is mediated through a commitment to organizational objectives. Such a change reduces the incidence of conflict of the sort described by Greenbaum and echoed at least faintly in systems people’s comments about bygone days at the bank: that is, the conflict between managers with business problems and systems people who controlled the technical domain. When systems people are brought closer into the organization and educated about the business objectives, they become both more capable of communicating with users and more identified with the overall goals of the organization.
FIGURE 1. THE BANK AFTER DECENTRALIZATION
BIBLIOGRAPHY


Glenn, Evelyn Nakano and Feldberg, Roslyn L. "Proletarianizing Clerical Work: Technology and Organizational Control in the Office." In Case Studies


