360

THE NATURE OF THE CONTROL OF WORK IN COMPUTER SOFTWARE PRODUCTION

Thesis Submitted For The Ph.D. Examination
of the University of London

by

Richard William Holti

Imperial College of Science and Technology

June 1989

Abstract

The thesis investigates aspects of the control of work in software production, as 'professional technical' work characterised by a 'high-trust' form of work organisation. Its aims are twofold: to establish what forms of conflict arise between managers and workers in such work; and to understand the forces that shape the development of such occupations, in particular their prospects for establishing occupational control over the conduct of their work.

A framework is developed for exploring these issues at the level of the firm. Concepts are drawn from literature on organisational adaptation to understand how managers may use a variety of coordination mechanisms to cope with uncertainty occurring in their environment, alongside mechanisms for obtaining desired levels of performance from individual producers (labour control). An examination of literature on professional technical workers gives rise to a view of them as expecting challenge and autonomy in their work, in exchange for commitment to enterprise goals and norms of quality and productivity.

Four case studies of software-producing organisations identify mechanisms of control and coordination adopted under different conditions, producers' reactions to them, the nature of tensions, conflicts and developments in the organisation of work. Data indicate that formalised techniques for the conduct of software production co-exist with the use of discretion and forms of labour control based on trust. Formalised techniques fulfil functions of coordination, and can also be seen as 'professional tools' in assuring quality of work. They are not in general used to define routine areas of work so that less skilled workers with separate career structures can be introduced, and do not contradict 'high trust' principles. At the same time, the fact that desires for challenge and autonomy are largely met suggest that producers have little reason to seek to establish true occupational control over their work.

Acknowledgements

So many friends, colleagues and teaching staff have provided me with so many varied forms of help during the completion of this thesis that it is invidious to mention any by name. The one exception is Arthur Francis, who as my supervisor, has been unfailingly patient and understanding on one hand and immaginative on the other. Many other staff of what is now the Management School at Imperial College have also been full of suggestions and interest. More recently, at the Tavistock Institute of Human Relations, Jean Neumann and Riccardo Peccei have been very helpful in commenting on drafts. I am of course grateful to the Joint Committee of the Economic and Social Research Council and the Science and Engineering Research Council for funding my time as a research student on this project. I am indebted to Maureen Staines and Rosemary Welchman for so much help in the final stages of production.

CONTENTS

	Page
CHAPTER 1: INTRODUCTION	15
The Emergence of Software Producing Occupations	15
The Approach	18
Outline of the Thesis	19
CHAPTER 2: THE EMERGENCE OF THE UK SOFTWARE INDUSTRY AND SOFTWARE OCCUPATIONS	24
Introduction	24
2.1 Up to 1961: The First and Second Generation of Computer Systems	24
2.2 1961-69: The Third Generation of Systems	36
2.3 The Fourth Generation of Computer Systems	45
Conclusion	59
CHAPTER 3: MANAGEMENT CONCERNS AND INITIATIVES IN SOFTWARE PRODUCTION	61
Introduction	62
3.1 Problems in the Management of Software Production	63
3.2 Developments in Production Technology	66
3.3 Developments in Organisation in the 1970s	78
3.4 The State of Software Occupations in the Early 1980s	85
Conclusion	88

CHAPTER 4: PROFESSIONAL TECHNICAL WORKERS: CONFLICT AND THE EVOLUTION OF OCCUPATIONS 90

Introduction 91

4.1 The Sociology of the Professions: Conflict between 'Professional' and 'Administrative' Principles 92

4.2 Marxist Accounts of Professional Technical Workers 110

4.3 Professional Technical Workers and Business Organisations: "Autonomy with Respect to Means but not Goals" 127

Conclusion 134

CHAPTER 5: MANAGEMENT CONTROL CONCERNS AND MECHANISMS IN PROFESSIONAL TECHNICAL WORK 137

Introduction 138

5.1 Conceptions of Management Control in Contingency Theory 140

5.2 The Shaping of Management Control Systems 153

5.3 The Implications of Business Strategies for Task Environments 158

5.4 Mechanisms of Control and Coordination 166

Conclusion 184

CHAPTER 6: RESEARCH FRAMEWORK AND METHODOLOGY 186

Introduction 187

6.1 Research Framework 187

6.2 Research Design and Research Questions 195

6.3 Research Methodology 201

CHAPTER 7: THE WORK-RELATED GOALS OF SOFTWARE PRODUCERS 213

Introduction 214

7.1 Goals in Current Work 214

7.2 Career Goals 217

Conclusion 218

CHAPTER 8: TELEXCO 223

Summary: Telexco 224

8.1 Business Strategy and Task Environment 225

8.2 Organisation, Divisions of Labour and Job Structures 229

8.3 Commitment to Work 251

8.4 The Nature of Tensions and Conflicts 256

8.5 Initiatives and Possible Outcomes 274

CHAPTER 9: EXCO 277

Summary: Exco 278

9.1 Business Strategy and Task Environment 279

9.2 Organisation, Career Structures, Control and Coordination 283

9.3 Mechanisms for Producing Commitment to Work 307

9.4 Issues and Tensions 309

9.5 Management Intentions and Possible Outcomes 323

CHAPTER 10: SYSCO, MANUFACTURING SECTION 330

Summary: Sysco 331

10.1 Environment and Business Strategy 332

10.2 Organisation, Career Structures, Control and Coordination 336

10.3 Commitment to Project Targets 360

10.4 Issues and Tensions 364

10.5 Strategies for Change and Possible Outcomes 379

CHAPTER 11: INFOSERVICES DIVISION 384

Summary: Infoservices 385

11.1 Business Strategy and Task Environment 386

11.2 Organisation, Career Structures, Control and Coordination 392

11.3 Mechanisms for Establishing Commitment to Work 423

11.4 The Nature of Tensions and Conflicts 429

11.5 Initiatives and Possible Outcomes 445

CHAPTER 12: CONCLUSIONS FROM THE CASES 453

Introduction 454

12.1 Business Strategies and Task Environments 455

12.2 Organisation, Job Structures, Control and Coordination 461

12.3 The Nature of Conflict 479

12.4 The Shaping of Software Workforces 488

Appendix 1: Checklist of Data to be obtained from Managers and Project Managers/Team Leaders with Selected Business Unit 499

Appendix 2: The Interview Schedule 502

Appendix 3: The Diary SheetS 503

Appendix 4: Questions used for Initial Analysis of Case Study Data 504

Bibliography 507

LIST OF FIGURES AND TABLES

		Page
Figure 5.1	A Two-Dimensional Continuum of Competitive Strategy	164
Figure 6.1	Management Concerns and Mechanisms for Control and Coordination	190
Figure 6.2	Analytical Framework for the Case Studies	212
Table 7.1	Goals relating to Current Work	220
Table 7.2	Goal Satisfaction and Features of Current Work	221
Table 7.3	Career Goals	222
Figure 8.1	Organisation of Communications Division	229
Table 8.1	Job Histories of Technical Department Staff	236
Table 8.2	Job Individuals expect to be in in Five Years Time	238
Table 8.3	Career Goals of Software Producers	238
Table 8.4	Formal and Informal Reporting Patterns	246
Table 8.5	The Use of Help from Team Leaders and Colleagues	246

Table 8.6	Discussing Interfaces with Team Leaders and Colleagues	247
Table 8.7	Perceived Importance of Various Criteria for Promotion	254
Table 8.8	Views on Overall Goal Satisfaction	260
Table 8.9	Experience of Utilisation of Skills	260
Table 8.10	Experience of Autonomy in Carrying out Tasks	261
Table 8.11	Impact and visibility of Contributions	262
Table 8.12	View on Salary	263
Table 8.13	Satisfaction with Promotion Prospects	263
Table 8.14	Attitudes to Learning Advanced Software Engineering	264
Figure 9.1	Organisation of Exco	284
Figure 9.2	Organisation of the Software Development Centre	288
Table 9.1	The Nature of the Workforce within Exco	290
Table 9.2	Job Individuals expect to be in in Five Years Time	291

Table 9.3	Career Goals of Software Producers	291
Figure 9.3	Job Titles with Exco	292
Table 9.4	Time Spent Receiving Help	302
Table 9.5	Time Spent Reporting Informally to Supervisors	302
Table 9.6	Time spent discussing Interfaces	303
Table 9.7	Perceived Importance of Various Criteria for Promotion	308
Table 9.8	Views on Overall Goal Achievement	313
Table 9.9	Experience of Utilisation of Skills	314
Table 9.10	Experience of Autonomy in Carrying out Tasks	314
Table 9.11	Impact and Visibility of Contributions	315
Table 9.12	View on Salary	316
Table 9.13	Views on Probable Timing of Next Promotion	316
Table 9.14	Satisfaction with Promotion Prospects	317
Table 9.15	Attitudes to Learning Advanced Software Engineering	317

Figure 10.1	Organisation of the Production Department	337
Figure 10.2	Job Titles and Responsibilities	342
Table 10.1	The Nature of the Workforce within Sysco	344
Table 10.2	Job Individuals expect to be in in Five Years Time	345
Table 10.3	Career Goals for Software Producers	345
Table 10.4	Informal Reporting for Programming Staff	355
Table 10.5	Time Spent Discussing Interfaces	357
Table 10.6	Perceived Importance of Various Criteria for Promotion	363
Table 10.7	Achieving Most Important Work-Related Goals	367
Table 10.8	Extent to Which Job Measures up to What is Desired	368
Table 10.9	Experience of Utilisation of Skills	369
Table 10.10	Experience of Autonomy in Carrying out Tasks	369
Table 10.11	Impact and Visibility of Contributions	370
Table 10.12	View on Salary	371

Table 10.13	Satisfaction with Promotion Prospects	372
Table 10.14	Attitudes to Learning Advanced Software Engineering	372
Figure 11.1	Organisational Structure of ISD	393
Figure 11.2	Job Titles in Infoservices	402
Table 11.1	Job Histories of S & P Staff	403
Table 11.2	Job Individuals expect to be in in Five Years Time	405
Table 11.3	Career Goals of Software Producers	405
Table 11.4	General Help from Team Leaders	415
Table 11.5	Discussing Interfaces with Team Leaders and Colleagues	415
Table 11.6	Informal Reporting Patterns	415
Table 11.7	Perceived Importance of Various Criteria for Promotion	428
Table 11.8	Views on Overall Goal Satisfaction	433
Table 11.9	Experience of Utilisation of Skills	434
Table 11.10	Experience of Autonomy in Carrying out Tasks	434
Table 11.11	Impact and Visibility of Contributions	435

Table 11.12	View on Salary	436
Table 11.13	Views on Probable Timing of Next Promotion	437
Table 11.14	Satisfaction with Promotion Prospects	437
Table 11.15	Attitudes to Learning Advanced Software Engineering	438
Figure 12.1	Business Strategies and Task Environments	458
Figure 12.2	Divisions of Labour and Career Structures	462
Figure 12.3	Importance of Problem-Solving and Rule Adherence in Promotion Assessments	478
Table 12.1	Experiences of Utilisation, Control over Work, and Visibility of Contribution	484

CHAPTER ONE

INTRODUCTION

The Emergence of Software Producing Occupations

There is in English no single collective noun for referring to all those involved in the production of computer software, the instructions that make computers able to perform calculations and other types of information processing. They appear to include a variety of occupational groups, including 'programmers', 'systems analysts' and 'software engineers', each of which has rather indistinct boundaries.

These software producers are in many ways the most important new occupational group to have emerged in developed economies in the post war era. The UK market for software products and services was worth over £1 billion in 1985 (Acard: 1986). In terms of their economic role, software producers are the key to the widespread automation of clerical and service tasks that has taken place with the growth of computerised 'data processing' in most branches of industry and public administration over the last 30 years, and to a continual development in telecommunications technology and infrastructure. They are also vital to more recent developments of flexible forms of automated manufacturing technology, using programmable micro-electronic hardware, and the use of computer systems to integrate design, manufacture and commercial elements of manufacturing, known as Computer Integrated Manufacturing (CIM).

Such technological developments, and by implication, the importance of software production, are common elements in a number of diverse attempts to make a case that the 1980s is seeing a transition to a new type of industrial organisation in the developed world. This is generally seen in terms of a shift from manufacturing to service employment, from mass markets, large firms and units, mass production and collectivised industrial relations to niche markets, small flexible firms, and individualised industrial relations. The new forms are characterised variously as 'post industrial', 'post Fordist', and 'the information economy'.

Software producers have a further importance in visions of the 'post industrial' economy and society. They are seen as an exemplar of a new kind of worker, with a new kind of relationship to their employer and their work. An important element in Bell's relatively early account (1968) of the 'post industrial' economy was the idea that industrial conflict would disappear as technical advances eliminated the need for heavy and repetitive work, and workers of all types obtained greater autonomy and responsibility in performing tasks that required knowledge, skill and discretion. The 'post industrial' workforce would become trusted 'professionals', rather than exploited and recalcitrant 'detail workers'. Software producers would appear to be prime candidates as the paradigm for what is supposed to be in store for everyone. They are people, of either gender, who perform demanding mental work associated with highly automated systems of production. They work in clean environments, derive satisfaction from their work, are intrinsically as well as extrinsically motivated, non-unionised, well-paid and middle class in outlook.

These ideas that software producers might be the first of the new 'post industrial', 'professionalised' industrial occupations, who work in organisations without the institutionalised conflict typical of bureaucracy and mass production raises a number of basic questions for industrial sociology.

First, there is the issue of whether software-producing organisations are truly without fundamental conflict between managers and direct producers of software, the 'workers'. If there is conflict, what form does it take, and how comparable is it to the forms of management-worker relationships described by industrial sociologists in more traditional sectors?

Second, there are questions of the nature of the emergence of these occupations and of their future development. What are the conditions under which 'trusted' occupations emerge, and are they likely to stay 'trusted', or do managers seek gradually to establish more direct control over what they do?

Finally, there is an issue as to what 'professionalised' or 'professionalisation' means for these occupations. Various occupational groups involved in software production appear to have emerged, but the senses in which they actually are 'professions' are not clear. In the 'post industrial' literature, the term appears to refer principally to a type of work organisation where workers are given responsibility and autonomy in performing tasks, as opposed to having their work specified and monitored in detail.

Other streams of sociological literature have identified other facets of the concept. In particular professionals are usually identified as occupational groups engaged in complex work which have established some sort of control at a societal level over admission to the occupation and the conduct of work. Practitioners of medicine, law and architecture are seen as having established this sort of control at the expense of other agencies, through the mobilisation of forms of power available to them in an essentially political process. There are debates in the literature as to how far newer 'industrial professions', such as engineers, or accountants are actually professions, in that they share some, but not all of the the attributes of the 'traditional professions.'

Such debates raise issues about whether software producers are 'professional,' particularly in the sense of whether or not there is some form of occupational control over their work. If not, what are the prospects for its emergence?

These are three basic questions which this thesis seeks to answer. The approach taken starts from an assumption, moreover, that these three questions are very much connected. The nature of conflict between managers and workers, the evolution of an occupation and the degree of occupational control it achieves are interconnected phenomena.

The Approach

The thesis starts from the assumption that the evolution of any occupation must be understood in terms of the interaction of processes at two different levels.

First, there are processes at the level of the industry concerned, by which the task and character of employing organisations and members of the occupation are shaped. These include the development of products and the technologies associated with them, the development of technologies for making these products, the state of demand for products and of competition between suppliers, and the forms of training available to members of the occupation.

Second, there are processes at the level of the enterprise, by which managers and workers shape forms of work organisation and divisions of labour, in the context of an environment determined by industry-level processes. This joint shaping of organisational outcomes, and the forms of conflict arising within it, can be understood in terms of the business strategy of the organisation and the priorities of managers stemming from it on the one hand, and the orientations of members of the occupation on the other.

Both levels of processes involve the interplay of interests, rather than the working out of a simple logic of technical advance. New product and process technologies can give rise to negotiation over the nature of change at the level at the enterprise. Equally, the emergence of new product or process technologies can be a consequence of particular interest groups putting resources into particular research and development projects at industry or national level.

The aim of this thesis is principally to illuminate at the level of the enterprise the way that conflict occurs between software producers and managers, and the processes by which divisions of labour are shaped, at one particular point in time. On this basis it will then be possible to gain some insight into the way

that these processes at the level of the enterprise influence the evolution of occupations at industry level.

The rationale for the focus on enterprise-level processes is the diverse forms that techniques and work organisation seem to be taking in different software-producing organisations. Even at the end of the 80s, it is difficult to detect general tendencies. Different outcomes seem to be the result of different combinations of factors at the level of the firm. A clearer understanding of these factors is needed.

At the most basic level, the approach taken in studying enterprise-level phenomena is first to understand management concerns in controlling software production as stemming from the demands of the firm's business environment, but mediated through a business strategy. Together these shape the organisational structure and a system for organising and controlling work. Once management control concerns have been analysed in these terms, their demands can be compared with the preferences of software producers for control over their work. This provides a basis for analysing conflict, developments in work organisation and the division of labour.

Outline Of The Thesis

The first task of the thesis is to set out the sorts of business environments and workforces typical of software-producing organisations. Accordingly, Chapter 2 is concerned with the emergence of software-producing organisations, their nature, the sorts of business environments typical of them, and their workforces. The chapter describes how developments in computer hardware and software technology have been shaped by social and economic forces, how the demand for computer software has grown, and the sorts of organisation that have come into being to meet it. It also describes the growth in numbers and the broad categories of computer software producers. It provides the context in terms of industry-level developments over a thirty-

five year period, from around 1950 until the early 80's, for understanding the range of concerns faced by managers in software-producing organisations, and the broad nature of software-producing workforces in the early 80s.

Chapter 3 sets out available data as to the sorts of issues experienced by managers within software-producing organisations, and the way these have shaped the development of work organisation, the division of labour and software production techniques. It concludes that in recent years there have been a plethora of initiatives in each of these areas, but that no clear picture emerges as to how exactly occupations are being shaped, or the nature of conflict involved.

It describes how, according to managers, trainers and academics serving the industry, software production has in fact been characterised by serious problems and tensions throughout its thirty-five years of existence. In particular, there have been problems of labour shortage associated with rapid expansion, and problems of project control associated with the uncertain nature of software production. Chapter 3 describes how these problems have led to a series of initiatives in terms of new techniques and practices in software development, including the introduction of formalised techniques, such as 'structured programming' and the idea of establishing engineering rigour in the sector, under the banner of 'software engineering'. There is however no clear picture of the impact of new techniques, of the extent to which they are associated with developments in work organisation, divisions of labour or conflict. Some have seen them as leading to a 'proletarianisation' of software production, with increasing management control of work. Others see them as the basis for establishing a rigorous foundation of techniques, to be used by fully professional practitioners, working with autonomy from management.

This chapter raises some specific issues concerning the nature of conflict over the introduction of formalised techniques by managers. Why are codified techniques being introduced and what

are their consequences for 'direct producers'? Are they being introduced to establish more direct management control over the work of individuals, or for some other motive? Regardless of their motive for introduction, what effects are they having over software producers' control of their work, and are they a site of conflict?

Chapters 4 and 5 are concerned with reviewing sociological literature in order to derive concepts capable of clarifying the developments described in Chapters 2 and 3.

Chapter 4 examines the adequacy of two main streams of theorising concerning relations between 'professional technical' workers, such as software producers, and managers of organisations employing them. It looks at work that has seen professional technical workers in conflict with managers because they have some sort of occupational allegiance which over-rides their commitment to organisational goals. It also looks at work in the neo-Marxist 'labour process' tradition, which has seen professional technical workers in fundamental conflict with managers over the control of work processes and levels of productivity.

Chapter 4 concludes that neither of these two schools provides concepts adequate for understanding the range of developments and conflicts suggested by Chapter 3. Developments do not seem to be characterised by either form of fundamental conflict. It is necessary to look beyond these two areas of literature to find a basis for analysing the nature of management concerns, worker orientations, developments in production technique, work organisation and management-worker conflict found in software production.

The chapter begins this task by drawing on a third stream of study on professional technical work, typified by Larson (1977) and Ritti (1971). Such authors see professional technical workers as basically bound to organisational goals, in exchange for autonomy in how they organise their work. Their work is taken as

providing a basis for investigating the orientations of software producers.

Chapter 5 is concerned with deriving concepts for investigating business strategies and the concerns of managers in software-producing organisations. It draws on the literature on business strategy, contingency theories of organisational structure, and theories of strategic choice and organisational adaptation in arriving at concepts for understanding how managers may shape systems of control and coordination to meet the demands of their business strategy and business environment.

In particular, this chapter emphasises that managers in general, and those within software-producing organisations in particular, are concerned with both control of levels of performance of workers, and coordination of work. The same mechanisms may not be adequate for both. In understanding the nature of conflict and the way that divisions of labour are shaped, management concerns of coordination and control are both important. The chapter concludes with a categorisation of mechanisms of control and co-ordination that may be adopted by managers under different business conditions, and discusses the way they may give rise to conflict with software producers. It outlines concepts for understanding the circumstances under which managers may wish to introduce formalised techniques to professional technical work, and the way they may be received by workers.

Chapter 6 sets out a conceptual framework which summarises the concepts discovered in Chapters 4 and 5 for analysing worker orientations and management concerns. The framework is used to draw up five detailed sets of research questions, based on the three issues set out earlier on pages two and three, to guide empirical research in case studies.

The five sets of research questions concern: the nature of business strategies and overall management concerns; the nature of organisational structures, divisions of labour and systems of control and coordination employed; the nature of software

producers' orientations to work; the nature of conflict and tensions between managers and software producers; and the nature of any initiatives for change occurring in this context, and their possible impact on the character of software occupations.

The remaining chapters are concerned with findings from the cases. Chapter 7 examines the nature of producers' orientations to work, looking at data across the four case studies undertaken. Chapters 8 to 11 present data from these four cases, under the headings corresponding to the five sets of research questions.

Chapter 12 then draws conclusions concerning the research questions and the more general issues outlined in this chapter. It discusses whether software-producing occupations are likely to continue as 'trusted' workers, enjoying autonomy in their work, or not. It examines how an understanding of the nature of conflict between managers and software-producers and the likely lines of development of software-producing occupations also illuminates the forces that are leading to the emergence of some form of occupational control and those that are pulling against. It returns to the more general question of the relationship between the existence of a 'trusted' occupational group and the emergence of a 'profession', through the establishment of occupational regulation of its area of work.

CHAPTER 2

THE EMERGENCE OF THE UK SOFTWARE INDUSTRY AND SOFTWARE OCCUPATIONS

INTRODUCTION

The aims of this chapter are twofold: first, to describe the development of the U.K. software industry and software-developing organisations up to 1983 and second, to review what data is available on the emergence of software occupations up to that date. Running through it is a conception that the shaping of the hardware and software technologies upon which this industry is based is a socially determined process. Technological solutions are sought by certain interests to solve various social and economic problems.

The chapter is structured on the basis that it is possible to divide the history of computing into three main periods. In each period we look at developments in hardware, in software technology, in the application of computer systems, in the character and number of software-producing organisations, and in the nature of software-producing occupations themselves.

2.1 UP TO 1961: THE FIRST AND SECOND GENERATION OF COMPUTER SYSTEMS.

First Generation Computer Hardware And Its Manufacturers

The computer emerged and developed gradually through a number of remarkably concurrent research and development programs in a number of countries. It was a response to social, economic and technical developments which gave rise to an enormous demand for information processing within business, scientific, military and state organisations by the end of the 1940s. For example, the rise of large bureaucratic corporations with their own internal systems of accounting and monitoring of business transactions meant that commercial information processing activities came to be as significant as direct manufacturing in their consumption of wage-labour (Westin and Baker 1972: 222).

By the 1930s and 1940s technologies were already available for the automation of these areas of information processing. In Europe and the USA, much commercial and state statistical work was done on hand-calculators and electro-magnetic tabulating and sorting machines. The American market for such technology was dominated by International Bureau Machines Ltd (IBM). The U.K. had two market leaders, the British Tabulating Machine Company (BTM) and Powers-Samas Accounting Machines. These firms were all amongst the market leaders in the manufacturing and selling of early commercial digital computers, and established a practice of being responsible for supplying the 'program' (the software) and maintaining it, along with the hardware. This remained the norm in the computer industry on both sides of the Atlantic for the first eighteen years of commercial computing, from 1951 to 1969.

Whilst the early commercial tabulating machine companies provided the institutional basis for the new electronic computer industry, the technical basis for the first generation of computers came from the pre-war and wartime calculating machines developed for military purposes.

The computational needs of the U.K. and U.S. military in the second world war for weapons design and code-breaking provided a great stimulus to the development of faster and more powerful electronic calculating machines. These, however, had no capacity to store instructions or to be 'reprogrammed' without having to undergo physical rewiring (Lavington 1980: 98).

In the USA the experience of building one such machine, ENIAC, was the basis for a subsequent university-based project to build a general purpose computer for the US army. This was to be able to store instructions and be instantly reprogrammable. It was called the EDVAC. It was for this project in 1945 that the mathematician John von Neuman wrote the first explicit proposal for a stored-program digital electronic calculating machine.

Von Neuman's concepts were freely circulated in the academic world, and were a vital part of the project that actually produced the first true Stored Program Computer (SPC), at Manchester University, in 1947. This was a very small model, built using internal university research funding, and war surplus valves.

Roughly contemporaneous with the early Manchester Project was a larger Cambridge University project, initially funded solely by the University. From 1948 there was major support from J. Lyons & Co. Ltd., the catering firm, whose directors had a conception of the commercial application of computers that can only be described as visionary. The project was also used as the basis for Lyons' own LEO (Lyons Electronic Office) computer, designed and built by a new computing division of the company. LEO was performing simple clerical tasks by 1951 and providing a full business data processing service (pay-roll, general ledger, and inventory control) by 1953 (Lavington 1980: 34).

This pattern, of an existing company developing a relationship with an academic research project once the initial viability of an SPC had been demonstrated and then developing a commercial product, was repeated with remarkable consistency to give rise to the six main manufacturers that made up the British computer industry by 1960. Soma (1976) identifies a similar pattern in the early US computer industry, although with a generally greater role for state funding.

In both Britain and America, the first generation of computers was characterised technologically by the use of valve logic circuits and a variety of store mechanisms until the almost universal adoption of magnetic core and tape. Already two broad types of application and computer can be identified. There were large-scale computers for military and scientific calculations, such as the MIT Whirlwind and ACE, and the smaller business data processing computers such as the BTM 1201, and the Ferranti PEGASUS.

The Second Generation of Hardware (1955-61)

Throughout the history of the computer industry, technological advance has been driven by competition between manufacturers to improve cost/performance ratios. Performance of a computer system derives from the size of its memory and the speed with which it can process instructions, and so there has been consistent attention in both academic and corporate laboratories of computer manufacturers and electronic component suppliers to memory and logic circuit technologies. The first great advance in the latter was the adoption of the semiconductor transistor as the basic building block to replace the thermionic diode. As with the SPC itself, the transistor "was not discovered in a scientific vacuum" (Soma 1976: 10), but resulted from an intensive research programme at Bell Laboratories, starting in 1945, to find a rapid-response semiconductor switching device. The second generation of computer systems is generally identified with the use of transistors.

Although the first transistor-based computer was again built as an experimental research machine at Manchester University in 1953, the initiative in software development and in turning the new technologies into products was firmly taken by US companies and state bodies. This was indicative of a higher level of state interest in and funding of computer development, and of the consequent developing superiority of American technology in this period. IBM unveiled the 1400 and 7000 series of transistorised computers in 1959, on the basis of which it consolidated its dominant share of the US market, built up over the previous five years.

Software Technology In The First and Second Generations

At first each computer manufacturer developed its own often highly idiosyncratic system of binary code for programming the computer, known as 'machine code'. Each transaction within the registers had to be specified separately in this binary code, an

extremely laborious process by modern standards. The mid 50s saw the first important development in software technology, the appearance of standardised high-level languages.

The late 40s and early 50s had seen some early high-level languages developed in the USA, the UK and Europe. It was realised that it was possible to write special programs, known as compilers, by which a computer could translate an algebraic representation of the algorithms involved in a program into binary machine code which could then be processed directly. This greatly speeded up programming, although it tended to slow down the actual processing, since the translated algebraic representations tended to end up as longer machine-code programs than if they had been turned into machine code by a programmer. Moreover, just about every type of computer used a different system of algebraic representation - or high level language - as well as a different binary machine code. Consequently there were about as many types of compiler as there were types of computer.

In 1954, IBM began developing a standard high-level language, FORTRAN, with a compiler, designed to facilitate programming without producing over-long and inefficient machine-code programs. The idea of producing programs that could be transferred from one IBM model to another was also a goal (Knuth and Pardo 1980: 241). FORTRAN was released with a compiler for the IBM 704 in 1956. Meanwhile the US Department of Defence (DoD) was becoming concerned with the multiplicity of high level languages and machine codes in use on its many different computers, and funded the development of a standardised high-level language which could be used in any of its machines with the aid of a suitable compiler. The Common Or Business Oriented Language (COBOL) was released in 1959, and to the present has remained the standard commercial language, although much modified. A language called JOVIAL was developed for 'real time' applications.

Applications For The First and Second Generations of Computers

By the end of the 50s, there were three clear classes of application: military, scientific, and commercial. Although military and scientific uses for calculation purposes overlapped considerably, particularly in the USA, the military had also sponsored a new and distinct class of computer application - that of 'real-time' control and guidance systems in air defence. The earliest example was the Semi-Automatic Ground Environment (SAGE) system for radar defence and anti-aircraft fire control, commissioned by the US DoD. This involved linking several IBM computers to respond, automatically, and instantaneously, to data from radar sensors. This was the first 'real-time' computer system, programmed in machine code.

'Commercial' applications of computers to areas such as stock control, payroll and administration were still limited to large organisations, which had to perform clerical data-processing on a large scale and which could justify the enormous investment a computer represented. Programs for these operations ran in 'batch' mode, where the program ran for a finite period - a few seconds or minutes - and was fed with all the data it required from a memory store of some form. Medium-sized and small industrial, commercial and public administration organizations seem to have continued to rely on electromechanical tabulators and calculators for data processing throughout the 1950s. By 1960, there were still only 220 commercially produced computers installed in the U.K. 90% of them designed and built by domestic firms (Lavington 1980: 103).

The Organisation of Software Production

Programming on the early computers commonly had the status of an afterthought, and was often performed by female assistants of the male engineers and mathematicians who pioneered the hardware. Kraft (1978) describes how women mathematics graduates were recruited to program the ENNIAC machine; and arguably the first

computer 'programmer' in history was Ada Lovelace, who produced mechanical disc programs for Charles Babbage's Universal Calculating Engines.

Software did not, in fact, emerge as a separate occupational sphere until the mid 50s. By then, staff identifiable as programmers were engaged in batch commercial and scientific dataprocessing, and in military real-time applications. They were generally recruited from the ranks of recent mathematics and science graduates and trained to programme the machine operated by their organisation, in machine code.

A general consequence of the diffusion of high level languages was that it became easier for a programmer to transfer from one machine to another. Programming became more standardised. This laid the way for the emergence during the 60s of labour markets for people skilled in software production. There appears however to be very little data available as to how specialised or general these labour markets were. It is unclear whether there were distinct labour markets associated with particular high-level langanges or application areas, or as to whether there were differences in this between the UK and the USA.

From an early stage, software production was organised rather differently in the commercial, military and scientific spheres.

In the commercial sphere, one small group of programmers would be responsible for the whole process of identifying the requirements for an application, programming it, installing it, testing it and getting it ready for use, as well as modifying it once it was in operation (Dolotta et al 1976: 88). They would sometimes be employees of the hardware manufacturers, loaned to the user. Most commonly though, they were specially recruited staff of the user organisation. In the latter case, 'programmers' might even be responsible for routine operation of the computer. There was then in this era no separate type of software-producing organisation in the dataprocessing field. Nor, initially, was there much of a distinction between the programmer and the user

of a computer system. During the course of the 50s, however, a separate job, computer operator, emerged within user organisations, since this task required much less expertise than the others within the sphere of responsibility of programmers.

In the military sphere, air defence systems were already so large and complex that they required large teams, sometimes of hundreds of programmers, to develop and maintain them. Typically, each programmer would be assigned one part of a vast computerised control system to design, program and maintain. The 'operators' were a distinct set of staff who scanned output on a print-out, or on a cathode ray tube display screen. As well as being associated with a distinction between users and programmers, military real-time systems were also typically produced in a distinct and new type of organisation. The earliest example is the Systems Development Corporation (SDC) in the USA, set up as part of Rand in 1956, to fulfil part of a huge DoD contract to develop the Semi Automatic Ground Environment (SAGE) system.

This highly sophisticated project set a pattern of the most technically advanced software being written for defence contracts in a different sort of organisation to those producing commercial software. It also meant that SDC made a major contribution to the training of the early US software producing industry (Baum: 1981). Organisations such as SDC became referred to as systems houses, because they took standard hardware and, by supplying software, delivered complete systems. The early US systems houses were thus a response to military needs. Systems divisions of existing British electronics firms came into existence during this era in a similar fashion.

In the case of scientific computer applications, there was rarely a distinction between 'users' and 'programmers'. The people who programmed the computers were often the mathematicians and engineers who had designed the machine, or else research scientists using the machine for performing calculations.

In terms of who the early programmers were, according to Greenbaum (1979: 15), expansion of computer usage in all three sectors combined with conditions of full employment and high demand for technically trained people to force U.S. software-producing organisations to 'poach' staff from other technical occupations. In particular, fresh graduates were recruited from mathematics, science and engineering courses. Employers offered higher salaries, and more flexible conditions of work than were on offer in traditional technical professions. Computer programming became associated with a mystique of technical advancement, high status and pay, and also with a culture of enthusiasm and irreverence for conventional 'organisational' behaviour and norms. An early survey of U.S. programmers characterised them as seeking 'independence and creativity' above all (Deutsch and Shea, Inc. 1963: 8-31). Pettigrew's (1973) characterisation of programmers in a British user firm of the period is consistent with this US picture.

The emergence of a division of labour in commercial data processing software production

Whilst there was an early division of labour in both military and commercial spheres between 'programmers' and 'users' or 'operators', for most of the 60's there was no formalised division of labour in the actual production of software. In both types of application area, the 'programmer' was responsible for all aspects of software development, from ascertaining the needs of users, through designing and coding the programs needed to meet these requirements, to installing and maintaining the completed code.

In contrast, the single occupational category of 'programmer' in commercial data processing software production appears to have lasted only a few years. Whilst programmers could be assimilated fairly easily into the culture of large science-based military research establishments, their introduction into commercial organisations was much more disturbing. This appears to have

been a major reason for the emergence of a second distinct software occupation, the systems analyst, who was responsible for mediating between the dedicated, enthusiastic, expertise-oriented, but irreverent culture of early programmers and the commercially aware, respectful, company oriented, disciplined culture of most traditional white collar and managerial jobs in commercial organisations.

Pettigrew (1973) describes the resentment produced amongst headquarters staff of a large UK furniture firm when a computer was introduced for stock control in the early 1960s, and a group of programmers recruited. They were recent maths graduates, with first class degrees, and some experience in scientific computing. The rarity of their skills enabled them to exercise a great deal of power in deciding their conditions of employment. They could command salaries that upset the company status system. They worked hours that suited them, dressed as they pleased, and flouted accepted company standards of dutiful and responsible behaviour.

Apart from a disruptive effect on morale generally, this behaviour also exacerbated resistance of user departments of the organisation becoming involved in computerisation. Users disliked the programmers, and were also afraid that the computer would strip them of expertise and status.

According to Pettigrew, a major factor under-pinning this situation was the way that programmers established control over all the various stages of software development, from requirements analysis onwards. This itself depended on a number of factors. First, the programmers were working in machine code on a very restricted early machine. The systems that could be run were heavily constrained by the technical capabilities of the machine, and the programmers saw technical criteria as of overwhelming importance in questions of designing the system's functions. They considered that users should accept the functions that they, the programmers, defined. Second, their technical expertise enabled the programmers to actually establish and maintain this

desired control over the entire process of system development, installation and modification, without involving user departments to any significant degree.

This state of affairs prompted senior management to locate some new staff with computer responsiblities in a department separate from the computer department that housed the programmers. These new staff had commercial rather than computing experience. They were responsible for drawing up system requirements, and then handing them over to the programmers, who designed, and coded the software. All liaison with user departments was thus taken away from programmers. This development, in the early 1960s, coincided with the acquisi tion of a new, more powerful and less constraining computer, and the introduction of high level languages. These technical improvements allowed the new 'systems analysts' much greater latitude in specifying system functions. They were able to design systems more nearly in line with conceptions of the user department as to what was needed, as opposed to starting from what was technically possible, as the programmers generally had.

Once the programmers had turned the functional specifications delivered to them into code in a high level language, systems analysts and programmers worked together on testing of the software. Installation of the program and introducing it to operators and users were handled by the analysts. Any subsequent modification and maintenance again involved both groups of staff.

The markedly greater utility of the systems that resulted was not due merely to the greater power and flexibility of the new computer. Even before its installation, the systems analysts had a considerable impact (Pettigrew 1973: 92). This was attributable to their greater commercial understanding, as well as higher levels of trust and cooperation between users and those specifying the functions to be performed by the system.

By the mid-60s, the separate job descriptions of 'programmer' and 'systems analyst' were common in commercial data processing in both the USA and the UK. Pettigrew however describes a continued battle between programmers and analysts in the company he studied as to what precisely their fields of responsibility were, and as to what types of expertise were appropriate for developing applications software. This is an indication of a generalised lack of a clear boundary between the two occupations, which has in fact persisted.

According to Pettigrew, indeterminacy and conflict arose from questions of the extent to which these phases of the development process could realistically be isolated from one another, on the basis that some phases only required contact with the user and others only with the computer. In fact most phases required a degree of contact with both, and there was then conflict as to who was best equipped to perform or direct proceedings.

Within the more technical sphere of military real-time system development, no corresponding split of responsibilities emerged. This was presumably because of the comparative mathematical and scientific sophistication of computer users involved. Military and scientific staff were already used to specifying their requirements in a precise manner, and were already committed to the use of computers. Problems of communication and trust between programmers and users did not arise in the same way. The single occupational title of 'programmer' in this sphere of software development has remained intact to the present, although technology, team organisation, and paths of career progression through different areas of responsibility covered by the generic title have varied considerably.

2.2 1961-69: THE THIRD GENERATION OF SYSTEMS

The term 'third generation' is a convenient rather than a rigorous category, because it is a fairly arbitrary division of a continuum of computer development, driven by persistent competition between manufacturers to improve cost/performance ratios, and by the goals of academic laboratories of building faster and more powerful computers. There were, in fact, several significant discontinuities in computer technology, the uptake of computers, the development of the UK and US computer industries, and the organisation of software production between 1961 and 1969.

Hardware Developments

The third generation of computer systems is commonly identified with the application of integrated circuits.

During the 1950s, the US Federal Government invested huge sums of money within the US electronics industry and universities for the development of integrated circuit (IC) technology. This promised cheaper, smaller circuits, lower power consumption and greater reliability, due to fewer external connections, and faster individual circuit response times, due to smaller distances between components. The IC technology available by the early 60s promised greatly improved cost/processing speed ratios in computer hardware, and was rapidly taken up by computer manufacturers - far more rapidly than transistors themselves had been (Soma 1976: 24).

The large-scale use of ICs in the US military and space programs was a significant factor in this. It meant that economies of scale could be realised in volume production of ICs, lowering their price. A second significant influence on the take-up of ICs was the early decision by IBM to use them when they first became commercially available, albeit in a primitive form. This technology was the basis for the enormously successful business-

oriented 360 series, developed at great cost ($5 billion) with profits from the 1400 and 700 series and released in 1964.

During the 60s, there was consistent progress in producing smaller, cheaper and faster integrated circuits, which made annual reductions of 70-80% possible in the cost/performance ratios of computer hardware (Soma 1976: 27). Some smaller-scale scientific and commercial applications, such as instrumentation, and real-time control and monitoring of industrial processes, did not require the full processing power and large memories of the standard commercial computers of this period. Lower hardware costs allowed some manufacturers to perceive a market for a significantly cheaper and smaller type of computer, initially with scientific research laboratories in mind.

The first such manufacturer was the Digital Equipment Corporation (DEC), in the USA, formed in the early 60s by a group of engineers from MIT to develop a small computer for the US Army - the PDP I. They produced a series of small computers, mostly sold to research organisations, which became DEC's business-oriented 'minicomputer' series of the 70s. The first of the line to achieve large-scale commercial sales was the PDP 8, released in 1968, and sold for less than $10,000.

Software Technology

The development of operating system (OS) software was the major third generation milestone in software technology. Operating systems allowed computers to be loaded with several programs at once, to be stored until the central processor was free to run them. The 'classic' operating system was that of the IBM 360 series, which included job control and multi-programming facilities, compilers for high level languages, file storage and file management software. The 360 series came to be regarded as the epitome of the 'third generation'. With it, the practice of marketing computers with an extended operating system or 'systems software' became established.

The state-of-the-art technology for writing software at the end of the fifties was the high level language, used with ingenuity to provide a working applications program on a machine of limited processing speed and storage capacity. This remained the case for most of the 60s. Apart from the increasing sophistication of systems software in making computers easy to use, fundamental developments in hardware technology far outstripped those in software technology. The basic techniques used to build software systems remained unchanged, although, as we shall see, applications became much more varied.

Third Generation Computer Applications

In both Europe and America, the decreased cost/performance ratios of the third generation hardware meant that computers powerful enough to run batch data processing systems for general accountancy, payroll and inventory control, became economically viable for a huge number of medium-sized enterprises. The mid-60s brought an explosion in the computerisation of UK industry.

The range of applications also began to widen enormously, as cheaper computers made more applications possible, and as computer manufacturers began to produce systems software that allowed real-time systems to be written easily in high-level languages, opening up a whole range of commercial and industrial real-time applications. These included the monitoring and control of continuous manufacturing processes, transport and distribution control, defence systems, air-traffic control systems, flight simulation, computer aided design and graphics. At the same time 'traditional' batch commercial data processing applications was also growing. Further, there was a steady diffusion of real-time computing into the business world from the more technologically advanced military sector. A symposium was held by the British Computer Society on 'Real-Time Systems' for commercial data-processing managers at Imperial College in January 1966.

By 1970, there were 4,500 commercially-built computers in the UK, but only half of them British designed and manufactured (Lavington 1980). The success of IBM and other US manufacturers in their UK operations is generally attributed to their ability to keep at the forefront of technical developments through the size of their reinvestments in product development; their more rapid growth sustained by the size of the US market, which allowed greater economies of scale in hardware manufacture; and also their high standard of after-sales service and support. The competition from giant US firms accelerated the concentration of the UK computer hardware industry.

The Emergence Of A Software Industry

The 1960s saw the rapid emergence of a whole new set of organisations, concerned with producing software and providing other 'computer services'. This growth involved the setting up of several different kinds of organisation.

The earliest type of organisation was the computer bureau, which hired out batch processing-time on a large computer as well as programming and card punching services, to organisations that only needed computing occasionally or which could not afford the overheads of their own computer. LEO computers hired out time from 1951, but the first proper bureau was set up in 1956 in London by Ferranti, using a PEGASUS machine. By 1963 there were 23 in Britain (Computer Survey, June 1963: 44), and also at least one organisation, Computer Programming Services Ltd., set up in 1955, which hired out programmers to user organisations for data processing jobs, and then ran them on a bureau's machines. About half the bureaux were set up by UK or US computer manufacturers to hire out processing time on their own machines.

By 1966, the UK Computer Survey estimated 90 organisations in the UK offering processing services, as well as programming and data-preparation services, and a handful of new firms, such as Computer Analysts and Programmers Ltd., founded in 1962, which

offered only systems analyses and programming services in various applications areas, such as accounting and process control. Such organisations, which contracted to write software for new commercial users, became known as 'software houses'.

Software houses also typically offered training courses in programming and systems analysis, in response to a shortage of experienced computer programmers and analysts at a time when the economic desirability of computerisation was becoming clear to many industrial concerns. By 1966, the Computer Survey reported 45-50% annual growth for the scientific bureau business and 20% on the commercial side, with a total volume of £40 million per year for all sales, including software and consultancy, of which £28 million was processing services. The Survey listed 82 bureaux offering processing services and also, generally, systems analysis and programming, and ten software houses which did not offer processing services.

The 60s thus saw the emergence of independent organisations selling data processing software, both 'systems' and 'applications'. Bureaux and software houses sold mainly on a contract basis, developing 'bespoke' systems for particular clients, although a few sold standard applications packages, such as payroll, where system requirements were common to most organisations. However, the bulk of systems software was still written by software teams in hardware manufacturing firms, and most applications software was still written in-house by user firms. In the USA, it has been estimated that the gross software product in 1970 was $5.32 million, broken down as follows:

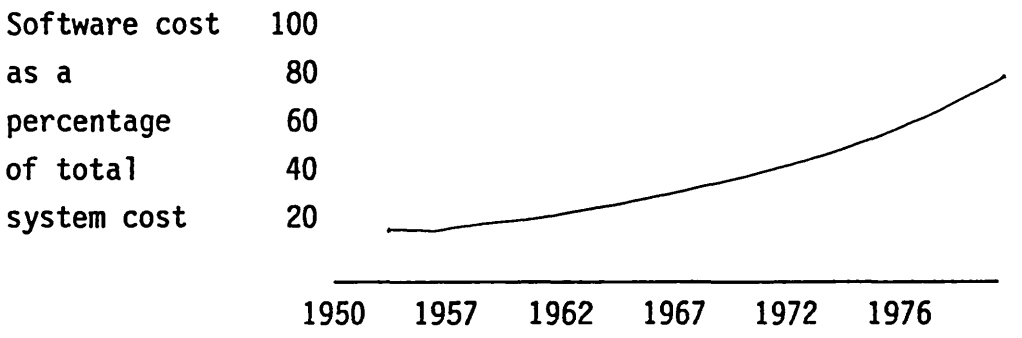
$0.05m products purchased from independents
$0.02m products purchased from hardware manufacturers
$0.3m provided free by manufacturers
$4.3m supplied in-house by users

(Dolotta et al 1976)

The explanation for the emergence of independent firms selling software is twofold. Firstly, the constantly increasing number of new users meant there was a market for customised software, and low barriers to entry, since only skills and experience were needed to write programs. Anyone who had had commercial experience of computing could consider setting up a software consultancy. Software was also written in small teams, and so there were no economies of scale associated with large organisations at this time.

Second, there was the changing proportion of total system costs accounted for by hardware and software in the average system. As Figure 2 shows by the mid 60s the fall in hardware costs and lack of similar progress in the technology of writing software meant that the software had on average taken up half of the user's costs in acquiring a system.

Figure 2.1: Software As A Proportion of Total System Costs



Source: (Soma 1976: 29)

This meant that software was no longer an economically insignificant add-on that users or manufacturers could afford to develop as an after-thought, but rather a commodity as profitable as the hardware itself. Market opportunities stemmed from the willingness of new users to turn to software houses of some experience, rather than set up their own programming

workforce, to save costs. Hence the increasing amount of resources, in the new software houses and in hardware manufacturers, devoted to software production.

The approximately equal sizes of total turnover in the hardware and software markets by the end of the 60s began to put hardware manufacturers in a dilemma. At first, IBM, the market leader in both US and Europe, continued the practice of the early 60s; hardware, systems software and standard applications software and after-sale support were all sold in an indivisibly priced 'bundle'. This seems to have been an attempt to realise both hardware and software profits, by not allowing independent software suppliers to compete for the software side of the system. However, it also meant that IBM was effectively charging for applications software that would not be used if the user had an unusual system to be computerised or an applications area not covered by standard IBM packages. It was arguably restricting potential hardware sales by this strategy.

In addition, in the USA at least, it was exposing itself to an incredibly complex series of anti-trust law suits from other hardware manufacturers, which claimed IBM was stifling competition in the computer industry. Finally, in 1969 the Justice Department itself filed a suit, and in June IBM announced that it was charging separately for applications and systems software (except for the most basic OS components), and support services. The effect of this decision was finally to open up a straightforward competitive market for virtually all types of applications and systems software, as other UK and US hardware manufacturers followed suit.

Software Occupations and Work Organisation in the 1960s

We have seen how the third generation of computer systems saw an enormous expansion in the application of computers and the number of computers in use, and how there was a commensurate increase in the volume of software production, and numbers of organisations

engaged in it, in both the USA and UK. There appears however, to have been very little change in terms of techniques of software production, the occupations involved, and the forms of work organisation adopted.

In terms of occupations, there was increasing, but by no means hard and fast, differentiation between those working in commercial data-processing, and those producing more technically intricate military or scientific applications software. In addition, a separate class of 'technical' programmer engaged in developing operating systems, high-level languages, compilers and other 'software tools' for running computers or assisting applications programmers emerged. These were known as 'systems programmers', and were mostly employed by computer hardware manufacturers.

In the commercial data processing sphere the 60s saw the establishment of the separate occupations of programmer and systems analyst, although it was common for 'programmers' to become 'analysts' as they gained experience of the 'user activities' they were involved in computerising. Although no systematic data is available, programmers and analysts appear to have been recruited from a great variety of backgrounds. In the US, according to Greenbaum (1979: 15), the majority of graduates entering the industry were recruited and trained by hardware manufacturers. The period was characterised by job-hopping and inflated salaries as many were enticed away to independent producers and user firms. Many software producers, in both the UK and US, were however not graduates. They were often simply technical or white collar employees whom managers put to work on the computer when it arrived, after a little training from the manufacturer's representative. During the 60s, a fairly integrated labour market for data processing applications programmers and analysts appears to have become established in both US and USA, based on the acceptance of COBOL as a standard language for business applications. There were however many specialist niches within this labour market, based on knowledge of the operating systems and other systems software of particular

hardware manufacturers, or expertise in particular applications areas, such as stock control.

In the more 'technical' military, scientific and 'systems' spheres, several overlapping labour markets became established, based on expertise with particular high-level languages, such as FORTRAN and JOVIAL, or low-level languages, such as assembly language. In such areas, it was common for a programmer to be expected to learn a new language on going to work on a new application. As in data processing, whilst many programmers in this sphere were maths graduates, the rapid expansion of the field at a time of near full employment, in both the UK and USA, meant that many people from a wide variety of backgrounds, usually young, and not always graduates, were recruited. This period saw the introduction of tests of logical aptitude as a tool for recruitment. Again, the pattern was that large organisations did the initial training, with smaller organisations relying on 'poaching' trained staff by offering higher salaries.

In terms of work organisation, the 60s saw the consolidation of the team as the basic organisational unit for producing software in most types of development. For smaller developments often found in data processing, one team of up to 10 analysts and programmers, led by a team leader, would be responsible for developing a complete applications system. For the larger developments typical of military real-time systems, team-based forms of organisation derived from established practices in engineering design and development projects appear to have been adopted. Several small teams totalling perhaps several hundred software producers, would be coordinated by a middle-management structure.

2.3 THE FOURTH GENERATION OF COMPUTER SYSTEMS

Using the term 'fourth generation' to describe computers and computer systems developed between 1969 and 1983 is once more a fairly arbitrary division in the development of computer technology and the computer industry. The period is characterised by some quite dramatic developments in hardware technology and applications, and in the organisation of UK hardware and software industries, many of which originated before 1969.

Hardware Developments

The period saw dramatic advances in the amount of electronic circuitry that could be manufactured in integrated circuit form on a piece of silicon substrate. By 1970, the technology was known as large-scale integration (LSI) and in 1971 it became very large-scale integration (VLSI) when the, then small, US firm of Intel designed a process for putting a complete computer central processor on one chip of silicon, as part of a contract with a Japanese calculator manufacturer. There was continuous progress in the reduction of cost/performance ratios for computers, as such techniques have made it possible to produce more powerful circuitry more cheaply.

In late 1970, IBM launched the 370 series of mainframe computer, in a range of sizes, to replace the 360 series. It was able to use much of the 360 series systems software. The other US manufacturers and ICL (the result of a series of mergers and established as the largest UK manufacturer), soon followed with improvements on their third generation mainframes, but IBM was able to maintain its huge market share. The mainframe market in both the US and the UK grew less fast than it had in the late 60s however (Sobell 1981: 283). Nonetheless, significant expansion of computer use continued, driven more and more through sales of types of computers other than the mainframe. The

mainstay product of the hardware industry was now complemented by new varieties.

The application of LSI to minicomputer design allowed the processing power, range of applications and market for minicomputers to grow spectacularly during the early 70s. DEC preserved the largest share (41% of minicomputer sales worldwide by 1981), but was rapidly joined by eight other major US mini-manufacturers, including IBM. Models with 16-bit processors were designed, many times more powerful than the largest mainframes of a decade earlier, and able to satisfy all the data-processing needs of some small and medium-sized companies. New business computing markets were thus opened up. No UK mini-manufacturers emerged, and so the US manufacturers were able to achieve a sales growth of greater than 40% per year in the UK for most of the early and mid 70s (F.T. Survey on Computers, 18.1.82, P. VII).

The late 70s and early 80s saw a slight fall off in the rate of expansion of minicomputer markets, to around 20% per year worldwide. This has largely been attributed to the great success of the microcomputer for small-scale business or 'personal computing' made possible by Intel's development of the microprocessor. This coincided with the development of extremely small and cheap memory chips, which made it possible to manufacture small but self-sufficient computers, complete with memory and rudimentary systems software, for less than $1,000. A new batch of US microcomputer companies formed in the mid 70s to develop the market for 'personal' business computing. They were rapidly joined in the microcomputer market by new UK companies, such as Apricot, and then by mini- and mainframe manufacturers, such as Olivetti and DEC, and, most significantly, IBM in 1982. In 1982, the UK microcomputer market grew by 100% to reach a value of £375 million (F.T. 11.4.83, Computers in Business Special Survey, P. III).

Fourth Generation Software Technology

The first software advance of this period was 'time sharing'. This referred to a type of system where a number of terminals were connected to a computer's central processor, and where each terminal could be operated simultaneously in 'real-time' mode. Such a system allowed one central large computer to be used by tens or hundreds of different people, running different programs simultaneously. Time-sharing was originally developed in the USA for university systems, so that a greater number of people could have more-or-less instantaneous access to the computer. By the end of the 60s, IBM had developed its Time Sharing System (TSS) and Honeywell its MULTICS system, and real-time, time-shared installations were becoming increasingly widespread also in industry.

Time-sharing allowed one central computer to be used for batch processing of applications that did not require real-time processing, such as weekly or monthly accounts, and simultaneously for applications that greatly benefitted from real-time operation, such as inventory and production control. It also transformed and greatly expanded the service bureau industry in the US and the UK in the late 60s and early 70s. Most bureaux began to offer real-time processing by installing terminals within user premises and running a time-sharing 'host-computer' on their own site.

The linking of large numbers of terminals to one mainframe computer was commonplace by the late 70s. The development of mini and microcomputers raised the possibility of linking several smaller computers to one large mainframe. Simple processing could be done by the 'peripheral' computers, whilst large-scale batch processing, and the up-keep of central databases was performed by the mainframe. The mid 70s saw the development of 'networking' software technology in US and European universities, some software houses, and the software departments of the major mainframe manufacturers. This involved the development first of 'data communications' software, to manage the transfer of

information between the computers in a network, and second of database management systems (dbms), which allowed several different applications systems to run off a central database.

The Application of Fourth Generation Technology

The industrial application of such technology took the form of systems where: (i) a mainframe maintained and managed a central company database and performed central batch accounting processing, as well as 'strategic' computing, to supply top management with summary statistics and projections on performance, (ii) 'distributed' mini and later micro-computers which performed real-time 'command and control' processing, to aid operational management in inventory control or production control for example.

A second strand of computer application was a greater variety of more straightforward commercial information processing activities that it had become viable to computerise using relatively cheap, 'stand-alone' mini- and microcomputers.

Overall, the principle new areas where computers were in common use by 1983 were:

(i) control of physical processes e.g. robots, CNC machine tools, instrumentation, navigation, materials handling;

(ii) computer aided design and graphics;

(iii) word and text processing;

(iv) point of sales automation;

(v) printing and photocomposition;

(vi) telecommunications switching control, for networks carrying digital messages;

(vii) control systems in consumer goods e.g. televisions, washing machines.

The Development of the UK Software Industry in the 1970s

In late 1969 and early 1970 the effects of IBM's 'unbundling' decision became clearly visible. By mid 1970, the Computer Survey had entries for 167 computer bureaux and software houses, an increase of 75 over the previous year. The open market for systems and applications software seems to have encouraged a flood of new entries to the market, although the life-expectancy of many of these organisations was short. The independent software industry was none-the-less characterised by continued and spectacular growth for most of the 70s and early 80s.

Table 2.1: UK Computer Services Sector: Billings to Clients
(£ million, 1978 prices)

	1971	1972	1973	1974	1975	1976	1977	1978
Computer Processing	40.2	46.4	56.9	73.5	102.9	118.6	133.8	163.6
Software Sales	14.0	16.4	22.7	30.0	29.8	51.1	65.1	79.8
Consultancy	4.8	5.8	7.4	9.8	12.1	24.4	30.7	43.7

Source: F.T. Survey on Computer Services: 17.9.79

The 1982 edition of the UK Computer Users Handbook has just over 1,000 entries for organisations and individuals offering 'programming services' and general consultancy related to the design and installation of software systems. This number includes software and services departments attached to mainframe

and mini hardware manufacturers. The huge expansion of the sector, in terms of both numbers of firms and total revenue is explained at a most basic level by a continuation of processes described as operating during the 60s. The fall in hardware costs achieved during the period led to an overall fall in total system costs that allowed great expansion in computer usage, which generated a large demand for software.

The oil crisis of the early 70s and the ensuing recession brought a slowing of the increase in demand for computer services, which caused a spate of bankruptcies and redundancies in software producing organisations. Table I shows, however, that the dominant tendency was one of growth in computer services.

It is also important, however, to identify other processes which shaped the evolution of the industry, in particular the balance between software production in independent suppliers, mainframe manufacturing firms and user organisations, and the character of software production occurring in each.

Types of Software-Producing Firm

The 70s saw a straightforward expansion in the number of firms making up the independent computer services sector. The low barriers to entry and small-scale of operation required for many types of software development meant that it was easy for new firms to enter the market. A chronic shortage of skilled software staff has also been seen as a reason why independent software firms were able to establish a market for themselves (NEDC: 1980).

Because of the proportion of overall system costs deriving from software, and tighter profit margins in hardware markets, hardware manufacturers were placed in the dilemma of choosing whether to compete with independent software houses, to realise software profits, or whether to cooperate on joint sales, to maximise hardware sales. In practice, manufacturers managed to

do both. They set up software departments to supply applications software for standard applications, such as payroll, or general ledger, in competition with the independent suppliers. They also set up schemes for formal cooperation with specialised software suppliers, to cover non-standard applications. The great advantage hardware manufacturers had over small software houses was their extensive networks of sales representatives, originally set up to sell hardware, and used increasingly during the 70s to sell software as users came to be increasingly discerning about the software side of systems. As a result, IBM, for instance, managed to establish itself as the world's largest supplier of software as well as of hardware. In 1981, 60% of systems and application software supplied world-wide came from IBM (F.T. 5.11.82. Survey on Computing Services and Software, P.1).

The independent software sector was also boosted greatly by the early character of minicomputer manufacturers. DEC and its immitators - Hewlett-Packard, Data General and Prime - were all formed by small groups of engineers who perceived a market for small, cheap, and low-powered computers in scientific and military research establishments. Such users were capable of performing all their programming and servicing themselves. Initial success for the mini-manufacturers depended on the quality and cost of their hardware, and little else. They thus made no attempt to develop marketing, sales, software and support organisations to speak of, in the greatest contrast to the mainframe manufacturing giants.

Both mainframe and mini-manufacturers seem to have been slow to realise the potential of mini-computers in the business market. This was left to a variety of independent firms - mostly existing bureaux or software houses in the US - who bought mini computers at a discount in bulk, added systems and applications software, and peripherals such as terminals and printers, and sold them to commercial firms and government organisations for data processing purposes. Such firms became known as 'systems houses', since they made up components into complete systems, or, more confusingly, as 'Original Equipment Manufacturers' (OEMs).

Several were also set up in Britain in the early 70s, such as Logica, in 1973. Some existing bureaux and software houses also set up 'OEM' sections.

The particular character of minicomputer manufacturers thus led to a much greater proportion of minicomputer software being written by the independent sector as opposed to hardware manufacturers, than was the case for mainframe systems. Since the early 70s, however, minicomputer systems have been affected by the decreasing proportion of total system costs attributable to hardware, just as for mainframes. Minicomputer manufacturers thus became concerned about loosing out on the profits from software, particularly with the great popularity of microcomputers in the early 80s slowing the growth of the minicomputer hardware market. The larger mini manufacturers, such as DEC and Hewlett-Packard, moved into developing their own software, and supplying complete systems - particularly for office automation - direct to users. This made the OEM market much more competitive by the early 80s, forcing many OEMs into specialisation in particular application areas, or sizes of system, not catered for by the mini-manufacturers.

At the beginning of the 80s, the software industry has also seen the entry of firms concerned with selling software for microcomputers. As microcomputers became more powerful, they became suitable for 'desk-top' executive personal computing, and even 'traditional' data processing applications in small firms. Microcomputer manufacturers firms began to market software for such applications, as did a number of small specialised microcomputer software companies. The consequent threat experienced by both mini-manufacturers and OEMs to their sales of small systems has led both to diversify into supplying microcomputer systems. Meanwhile, mainframe manufacturers, such as IBM, began to market minis and micros to complement their mainframe ranges. By the mid-80s, the distinctions between mainframe, mini and microcomputer manufacturers as well as hardware and software suppliers had become rather blurred.

Particularly significant types of specialised OEMs or systems houses were those involved with the more technically complex real-time computer systems required for military 'command-and-control' systems, and telecommunications switching systems. In 1980 there were only seven software houses with the technical and security clearance to tender for Ministry of Defence (MoD) work. Military software was generally written on a sub-contracting basis for one of the electronics firms, such as Marconi, which contracted directly with the MoD. Telecommunications systems were developed by a few independent OEMs, such as Logica, and also by systems divisions of large electronics conglomerates, such as GEC, Plessey and STC. The more 'technical' end of the computer services industry was thus populated by a mixture of independent firms and subsidiaries or divisions of established electronics firms which had seen the need to diversify into software.

The Emergence of Packaged Applications Software

There were also sweeping changes in the nature of commercial data processing software products during the 70s, which affected the behaviour of firms and the structure of the industry. In 1970, most data processing software, systems and applications, was written for a mainframe, either by the manufacturer, the user organisation, or by an independent software house. Except for a few very standard applications, such as payroll, applications software was written as a 'bespoke' system. A new program was written for each computer application, tailored to the needs of the user organisation. Systems software existed as standard packages, such as operating systems, compilers or file-management systems, and was available principally from the hardware manufacturers, but also from software houses and bureaux.

During the 70s there was a move away from bespoke development contacts in applications software, too, and towards the marketing of standard packages for data processing applications. If the user organisation did not think it possible to adapt to this

standard, it had to either pay for a degree of customisation of the package, or have a bespoke system developed. The advantage of buying a standard package was that it was cheaper. Although developed and tested to higher standards than a bespoke system, the costs could be recouped by the supplier over several sales. The price also generally included maintenance by the supplier, and regular updates.

From the point of view of the supplier, developing a package product demanded considerable up-front investment, but the possibility of a continuous stream of income once the product was proven. It was also a way of harvesting the fruits of labour of a few highly skilled programmers, put to work to develop a package at a time when there was a shortage of skilled applications programmers.

By the late 70s, the proven quality and low cost of applications packages began to be appreciated, particularly as a great number of US-developed products became available through UK software houses. By 1980, the total market for package products in the UK was well established at £70 million per year, of which £43 million was systems products, and £27 million applications products - with the latter growing at twice the rate of the former. It is important to note that outstanding growth in this area in the UK did not mean that software houses necessarily reaped outstanding profits. This is explicable in terms of the large number of competitors, and also the fact that most packages for sale in the UK were on licence from the US developers.

Overview: The UK Software Industry in the Early 1980s

The best way of summarising the development of the software sector in the 1970s is first to emphasise the division between organisations developing 'technical' software - for example for real-time military systems, industrial process control, or telecommunications - and those supplying data processing software. The former consisted largely of independent firms and

new software divisions of electronics firms, which had previously provided 'hard-wired' systems for the same applications. In data processing software production, there was a much greater variety of organisations involved, and moreover, no simple way of categorising the organisations involved on the basis of their product mix, such was the general diversification taking place. Many firms that came into existence as software consultancies or

as bureaux in the 60s moved into supplying complete mini systems, or package software products for minis, micros and mainframes. Some software consultancies set up bureau divisions, and all mainframe manufacturers and most mini-manufacturers developed large software divisions, usually supplying a mix of package products and general software consultancy.

This range of product mixes also appears to have given rise to a similar variety in competitive strategies. The following examples are not meant to be comprehensive, merely to give an indication of the range that has to be taken into account. First, there were the competitive strategies of software divisions of large hardware suppliers, such as IBM or ICL, who sought competitive advance through the **range** of hardware, software, and consultancy services they could provide a business organisation in an integrated manner. Second, there were organisations specialising in providing software and design/customisation/ installation consultancy in a particular 'niche' application area e.g. manufacturing process control or telecommunications. A report by the Manpower Sub-Committee of the Electronics Sector Working Party of the National Economic Development Council in 1980 observed a trend for data processing systems and software houses to become more specialised as computer manufacturers took over the market for standard or general software. Thirdly, there were organisations which supplied software and hardware for common and standard applications e.g. word processing, general accounting, competing on the basis of price, and usually addressed to the small business or individual executive market.

In terms of the overall volume of software production from the

computer services sector, including data processing and technical systems and the size of other elements of the computer services industry, bureaux services were the largest component of the whole computer services industry in 1980, with a turnover of around £250 million (F.T. Survey on Computers, 2.3.81), compared with £70 million for packaged software and £80 million on 'professional services' and bespoke software.

'Professional services', including bespoke software development, and package sales were, however, both growing at around 40% p.a. whilst bureau turnover was only expanding at 20% p.a. The late 70s saw many bureaux moving into supplying data processing software as it came within the reach of more and more users to own their own mini or micro computers (NEDO: 1980).

The lack of concentration of the UK computer services industry was noteworthy. In 1980, the output of the twenty largest companies comprised only half the total turnover (F.T. Survey on Computing Services: 17.9.80). In 1980, only four companies employed over 1000. This contrasted with the French computer services industry, which was notably more concentrated. It had around twice the turnover of the UK industry, and consisted mainly of a handful of large bureaux/software house conglomerates, each roughly twice the size of the largest UK firms. By international standards, the UK industry consisted of a very large number of small firms, which may explain why very few software or systems houses have had the resources to develop their own major package application software products (Acard 1986).

Software Producers In The Early Eighties

In user firms, hardware manufacturers and independent software firms, there was by the end of the 70s a mixture of older software producers, recruited from a variety of backgrounds during the 60s, trained almost entirely through work experience, and newer software staff recruited in a more systematic way.

Most training of new recruits to the industry was still done within user firms and hardware manufacturers (NEDC 1980). Data processing user firms recruited staff from a wide variety of backgrounds, often at graduate level but by no means exclusively. It became widely accepted that data processing software skills could be acquired with a few week's introductory training, followed by one or two years of experience on increasingly complex work. A background in computing or formal mathematics was not considered necessary.

Independent software producers tended to rely on 'poaching' trained applications programmers from users, but also recruited from a similar range of disciplines for its applications programmers, tending to prefer graduates however. This was at least as much to do with the way graduates were better at relating to users as with their technical capabilities. 'Independents' also recruited at least as many more specialised graduates, with degrees in maths, sciences or computer science, to train as systems programmers or for more 'technical' applications programming, in areas such as real-time control or telecommunications.

Manufacturers tended to concentrate their recruitment on the more specialised graduates. Degree courses in computer science had begun to be offered by British universities from the mid-60s to meet the rapidly increasing demand for the more 'technical' side of software production. Although by 1980 most higher education institutions offered some form of computer science degree, according to the NEDC report, supply was not meeting demand, with shortages of up to 20% in particular skills (NEDC: 1980, P.xiii).

In terms of numbers of software producers, the 70s saw for the most part sustained growth. This was on the one hand a consequence of growth in numbers of software-producing organisations and overall turnover. It also stemmed from a lack of increase in labour productivity. In 1977 it was estimated that whilst cost/performance ratios for the hardware component of

systems had on average fallen by a factor of 200 since 1955, software productivity had risen by only 3 per cent each year (Nyborg et al 1977). According to Dolotta et al (1976: 93) most of this net two-fold increase in productivity was matched by increase in salary costs and so the cost of a line of software had hardly changed at all.

A survey carried out by the Institute of Manpower Studies for the NEDC in late 1978 estimated 17,000 'computer skilled professionals' specialising in software in the independent computer services sector. The NEDC also estimated there had been 20% financial growth and 40% manpower growth between 1971 and 1977, in a sector where software skills accounted for over 75% of the workforce (NEDC 1980: 103-110). The 1978 survey reported growth as particularly strong in large established firms and those offering services other than bureau processing facilities. It seems most of the growth was in software producers. The same report also concluded that the main barriers to further growth were shortages of skilled manpower, together with a lack of depth in management experience in the sector, needed to steer rapidly expanding businesses.

> "It is something of a paradox that the phenomenon which partially explains the rationale for a services sector - CSM shortages - also limits its ability to reap the glittering prizes which are there for the picking."
>
> (NEDC 1980: 115)

The same NEDC report also allows a comparison to be made with the numbers of software producers working for user companies and hardware manufacturers, to put the growth of the independent sector in perspective. It estimates 99,000 'professional computer-skilled manpower' i.e. software producers employed by user firms, and around 9,000 in hardware manufacturing firms in early 1979. By far the majority of a total of 126,000 were thus employed in the user sector. Employment in this sector was seen as growing at around 12% per year. This part of the software

workforce consisted mostly of data processing applications programmers and analysts, but with some more technical systems programmers. The 'independent' sector consisted of both data processing practitioners and a variety of more technical specialisms - such as systems software, telecommunications and industrial process control. The 'manufacturer' component appeared to consist largely of systems software specialists. Unfortunately, the NEDC report contains no estimates as to the numbers in each broad specialisation within each of the three broad types of employing organisation.

According to one review of the US software industry, the 70s saw a steady increase in the degree of specialisation amongst software producing occupations. (Dolotta et al, 1978: 92). This was attributed to the increasing number of application areas, and their increasing complexity, as well as the increasing sophistication of systems software, and of software tools such as 'librarian' systems, being produced to provide support for applications programmers. As each new area became established, labour markets developed for people skilled in it. Greenbaum (1979: 16) sees such 'horizontal' specialisation amongst applications programmers and analysts as having been encouraged by management in user organisations, as a way of limiting job mobility.

CONCLUSION

In this chapter, we have seen how the nature of software-producing organisations and occupations was shaped at a macro level by the development of computer technologies and a constant broadening of the scope and volume of their applications. A rapid increase in demand for computer software without any great advances in productivity in its production led to an expansion of the numbers of organisations producing it and the numbers of people working in them, as well as shortages of software producers. Together with the initial lack of interest in the new

mini-computer manufacturers in supplying software, this appears to have been a major factor in the rapid growth of an independent computer services sector in the UK.

The general picture by the early 80s is on the one hand of a great variety of organisations producing different kinds of software, as well as other computer-related goods and services, with similar variety in competitive strategies. On the other hand, there were two broad classes of producer, those involved in data processing and those in more technical areas of software. Particularly amongst the former, there appears to be a great variety in terms of educational level, and a strong emphasis on on-the-job training.

We now turn to the data available as to how software occupations had developed in more qualitative terms by the early 80s, examining the nature of management concerns at the level of the enterprise and the sorts of initiative in work organisation and production technique that occurred.

CHAPTER 3

MANAGEMENT CONCERNS AND INITIATIVES IN SOFTWARE PRODUCTION

INTRODUCTION

The aim of this chapter is survey the sorts of issues confronted by managers in software-producing organisations and the sorts of initiatives they were taking in response during the early 1980s. This is to provide an overview of the sorts of phenomena that are involved in researching the nature of conflict between managers and producers and the forces shaping the development of software occupations.

The first section outlines how the 70s was characterised by a persistence of the chronic management problems of labour turnover, project management and system maintenance, which had first emerged in the 60s. The second section outlines efforts in this period to develop codified techniques for producing software, as a way of tackling these problems, and data available as to their impacts on software producers. The third section then describes the types of initiative that were advocated in terms of changes in the division of labour and work organisation.

The final section summarises data available as to the overall development of software-producing occupations by the early 80s and the forces pushing for and against standardisation of training and regulation of skills. It also draws together the issues emerging from this account of the development of software occupations up to the early 1980s which must be considered in investigating the nature of conflict between managers and producers in software production, the lines of development of software occupations, and their prospects for professionalisation.

3.1 PROBLEMS IN THE MANAGEMENT OF SOFTWARE PRODUCTION

Whilst the 60s appear to have seen the consolidation of software producing technologies, occupations, and forms of work organisation that had first emerged in the 50s, they also saw the beginning of a focus on the problems of software production, which led to initiatives in technique and work organisation over the next two decades.

Expansion

A first basic source of problems in the development of the software industry was its rapid expansion and the failure of the UK vocational education and training system to respond to its needs. There was a practically continuous shortage of skilled and experienced software producers. This led to problems of poaching of skilled staff, high turnover and inflated salaries, as well as reluctance of employers to offer training for fear of losing their investment. This in turn exacerbated the tendency of staff to 'job-hop' in order to acquire new skills.

Project Control and Performance

Second, it became generally accepted that by the standard of most industrial processes, software production involved a high degree of complexity, both in assessing user requirements and in developing software to meet them. It was, however, difficult to anticipate (in advance) the complexity in developing a particular system, which introduced high levels of uncertainty for project managers, particularly in large system developments. Complexities in the tasks of individuals were found to have implications for the work of others, and were multiplied because intricate patterns of communication and mutual adjustment were called for. The resulting levels of complexity and uncertainty became difficult to manage successfully.

In the mid-70s, F.P. Brooks, the 'Father of the IBM system 360 Operating System' published 'The mythical man month' (1975), which vividly described the nature of project control problems on large development projects in the USA and was very influential on both sides of the Atlantic.

According to Brooks, a typical scenario was this. Clients or user divisions put pressure on development managers to deliver quickly. Managers then used a large number of development staff to do so. Managers then made estimates that were suspect in two senses. There was first a tendency to try and please the client and give optimistic estimates. Further, managers tended to estimate in 'man-months', 'as though men and month were interchangeable'. When slippage occurred, more staff would be added to the already large group of teams, to make up for lost time. In Brooks' opinion this often had minimal impact on delivery dates, and could even slow things down. The reasons were twofold. First there was the amount of time it took to train new project members in the overall system plan and development strategy. Second, the more people worked on the project, the greater the complexity of communication patterns needed to deal with adjustments that had to be made to the design of each part of the system as it was developed, since changes in one area generally affected others:

> "As work proceeds, the several teams slowly change the functions, sizes, and speeds of their owm programs, they explicitly or implicitly change their assumptions about the inputs available and the uses to be made of the outputs." (1975: 74)

The result was either a multiplication of errors, or an amount of producer time spent on inter-team communication that could exceed the extra resources added to the project by adding more people. This sort of scenario was undoubtedly a major reason why the industry became notorious for its poor performance in project management. Software developing organisations acquired a

reputation for delivering systems which not only did not meet user requirements, but which were also late and over budget.

Whilst this proved a more serious problem for organisations developing systems involving large numbers of producers working for several months or years, issues of poor quality and/or failure to meet time and cost targets were also endemic for small system developments and smaller producing organisations. As a result, in spite of a steady growth in numbers of firms, overall output and employment, the UK computer software industry was characterised by a high bankruptcy rate, periodic crises of consumer confidence and 'shake-outs'. At the same time, now that data processing computing was past its experimental phase, industrial user companies began to have higher expectations of what their internal data processing department was doing for them (Greenbaum 1979). They began to set more rigorous targets for project costs and benefits, and to want to control them more explicitly.

Maintenance and Modification of Software Systems

A third basic issue concerned difficulties that were being encountered in modifying software, once it had been developed, to take account of changes in user requirements, after a few months or years of use. In the 60s, software was written by programmers conscious of the need to save the restricted memory space on early hardware. Programs often contained code in which logical functions were compressed together and inter-twined. Whilst this required great ingenuity and extracted, in a certain sense, the best possible performance from the hardware, it made programs very difficult to read and understand for anyone other than the author. Consequently, software was often extremely difficult to correct when faults were found, or to modify when requirements changed. Moreover, because the code used often represented logical functions in forms that were compressed and unclear, what appeared to be a quite straightforward modification might have drastic unexpected effects in the functioning of another part of

the program. Such effects might take even the original author a great deal of time to unravel.

3.2 DEVELOPMENTS IN PRODUCTION TECHNOLOGY

Together, these issues of labour turnover, project control and maintenance first of all led to the creation of middle management structures to control the activities of software-producing teams more closely (Greenbaum 1979). Beyond this, attention within the industry became focussed on ideas of moving away from simple, uncodified ingenuity, gained through experience, as the best basis for developing software. Interest developed in notions of laying down more rigorous principles and techniques for moving from user requirements to the final code. These were seen as a way of assuring quality of software developed, as a basis for making it more standardised and easy to modify, and as a basis for training recruits to fill the manpower shortage. Interest in software development technique can be seen as the result of the increasing importance of the software-producing sector combined with problems in its performance.

The first codified techniques did not emerge until the end of the 60s, and so had little actual impact during this period. However, the idea of 'software engineering' according to rigorous standards of reliability, similar to those in traditional branches of engineering design and development, was born.

The term 'software engineering' was first used at a NATO conference on software development practices in 1968. Before describing the new techniques, we first of all describe the situation, in terms of software production technique, into which they were introduced.

Software Production Technology in 1970

In the last chapter we noted that at the beginning of the 70s software production was characterised by the use of high-level languages according to a fairly mysterious and intuitive 'black art', for designing programs that performed the required functions within the capacity of the hardware available. This 'black art' was however by then characterised by a broad notion that programming should proceed through 'modular design'. By the end of the 60s, six basic stages of software development were commonly recognised: requirements analysis; system design; program level design; coding; testing; and maintenance. Modular design covered the second of these.

In its most basic form, it consisted of splitting a large software system into a number of modules at an early stage in the design process. This process of modularisation is in fact a feature of just about any software design technique, but the terms 'modular design' or 'modular programming' are usually used to refer to a design methodology where the only formalised technique employed was this initial demarcation of distinct modules. This technique has little to say about how programming is conducted once modules have been designated.

The basic principle of modularisation is that modules should be logically self-contained and should perform one basic function. This means that it should be easy to substitute a module with an improved version for performing its particular function, without affecting the rest of the system. Modular programming therefore in theory gives rise to programs that are easier to modify and maintain.

Generally, during the 60s, individual modules were allocated to individual programmers. Part of the reason for the emergence of the technique was undoubtedly to find a way of distributing large developments over a team of programmers. There were further organisational benefits. If a program was divided into modules of different degrees of complexity or difficulty, they could be

assigned to programmers of corresponding levels of experience. Modular design enhanced the optimum use of skills. It also provided a cheap form of on-the-job training. The difficulty of modules allocated to a particular programmer could simply be gradually increased. Finally, the technique provided some sort of basis for monitoring the output of programmers.

The main problem with modular programming as a technique in itself was that it was really just a set of principles as to how a software system should be constructed, without a set of practical rules or instruments for putting those principles into action. The basic notion of modularisation needed fleshing out with techniques for dividing programs into modules, that could reliably be assumed to be logically independent, and then proceeding with development to more detailed levels. This need lay behind the development of a number of more detailed design and programming techniques.

The Arrival of Structured Techniques

The late 60s saw the invention of a number of more 'rigorous' software programming methodologies, which collectively became known as 'structured techniques'. They were all attempts to deal with the technical shortcomings of modular design, which were seen as major causes of problems of project control and maintainability of systems. The new techniques were also seen as relevant to issues of labour turnover.

(a) Structured Programming

The first of these techniques was structured programming. This consisted basically of a set of rules for carrying out the later stages of moving from a statement of user requirements to code. The basis of the technique was first published by Bohm and Jacopini (1966). It was that any program could be made up of combinations of just three logical elements. The vital common

feature of these elements was that each had a single point of entry and a single point of exit, which made the sequencing of a program made up of them very clear. They provided the basis for well designed programs which were easy for someone other than the author to understand and modify. This technique was seen as making maintenance a lot easier to carry out, as a way of assuring the quality of systems developed. It was also linked to issues of labour turnover and skill shortages. It was seen as a way of standardising the 'style' or approach to programming, so that it was easy for one programmer to take over development work started by another, or maintain a system developed by someone else. It was also a way of rapidly inculcating good programming practice, speeding the training of new recruits, and making them more productive more quickly (Yourden 1978).

To be strict, the definition above was in fact of how a structured program should be organised. Structured programming itself was a technique for producing structured programs. Although there were different opinions as to how structured programming should be conducted, there was a broad agreement that it consisted in starting with a high level representation of the basic functions to be performed by the program, in which only the permitted logical elements were used. The programmer should then proceed to more detailed representations of each function, again sticking to the permitted elements, in a series of stepwise refinements. Finally a structured diagram of the finest level of logical design was arrived at, which simply required coding into a machine-readable language. This required a language that had commands and procedures analogous to the permitted logical elements.

(b) Top-Down Design

As structured programming gained in popularity, attention turned to codifying analogous sets of rules for earlier stages of software production. Top-down design was in many respects an extension of the principle of modular design to later stages of

the development process. A program was first decomposed into a small number of basic functions (perhaps 4 or 5). Each of these was further decomposed into sub-functions, and so on, until the decomposition reached the level of modules considered the right size to be allocated to individual programmers for coding.

Most of the actual transformations of data to be performed by the program occurred within the lowest level modules, and so were specified last of all. In effect, the designer was allowed to forget about the details of transformations until the 'flow of control' between the various levels of functions and between the sub-functions at each level have been specified. Because top-down design separated out the part of a program that transformed data (the lower level modules) from the part that determined the flow of control, it was a very simple matter to change the former. If user requirements changed or were wrongly assessed, the transformations involved in a low level module could be altered without affecting higher level modules. Finally, because top-down designs were broken up into small modules, they provided a technical basis for monitoring and controlling progress of teams and of individuals engaged in coding them. They made progress more visible to project managers.

(c) Structured Design

The 1970s also saw the emergence of a number of more specific sets of rules as to how to proceed with a top-down modular design. The basic concern of these techniques was to assure the quality of the final code produced by first producing a clear design, according to definite standards in a distinct design phase. This made it easier to 'inspect out' design errors at an early stage in the development process, before anything was coded, which saved time compared to a method which did not have any criteria against which an un-coded design can be evaluated. In the latter case, errors could only be spotted in testing of the coded program. Error correction might then involve re-doing coding as well as design work. The clarity of design was also

intended to lead to a more reliable and predictable program, easier to modify and maintain.

Like structured programming, structured design was also seen as producing a desirable uniformity of style. If the same structured design methodology was used throughout an organisation, this should facilitate checking of designs by people other than the author, as well as handing over of work when staff left or were sick. Structured design was further seen as a codification of good practice, which allowed new and inexperienced software developers to learn how to produce good designs quickly.

Most of these techniques were intended for applications in data processing, where the structure of the data was the principal source of complexity and difficulty. Such applications are basically concerned with maintaining files of data, involving large numbers of cases, where each case has a large number of pieces of data or variables that describe it. The main difficulty in producing the system is taking into account all the different inter-related variables, and conceptualising their interaction e.g. how orders received and stocks moved both affect current stocks. The individual interactions are very simple, but the overall pattern is complex. Once a structure for the files and for the individual records within them has been arrived at, which takes account of interdependency between different variables, programming the effects of one variable on another, or the preparation of a summary report, is straightforward.

The two main techniques designed in the 70s for this sort of application were those of Yourden and Jackson. Each produced a textbook setting out their structured design methodology and also set up a company to market training for it. Their approaches have become established in both the USA and Europe, although Yourden predominates in the USA and Jackson in the UK.

The State of Development of Technique in the Early 1980s

It is apparent from the above that the 1970s saw the development of techniques consisting of rules for various stages of the software development process. The general movement was from structured techniques for the later stages - low level design and coding - to techniques for 'structuring' the earlier stages - first system design and then requirements analysis. The basic goal appears to have been to make approaches to development more reliable, easier to learn, and more standardised, and to produce clearer designs and code that were easy to maintain and modify. The move to 'structuring' at progressively earlier stages appears to have stemmed from a realisation that errors from earlier stages affected later ones, and so structuring of low level design, for example, was of limited benefit unless higher level design was also structured.

By the early 1980s, structured techniques were in widespread but by no means universal use in data processing software production in the UK. A survey of organisations producing data processing software in 1981 indicated that 38% were claiming to use 'modular programming', and 65% some form of structured programming or design. A much smaller proportion claimed to be using structured analysis, around 6%, on average introduced after 1979 (Friedmann and Cornford 1982: 8). The picture was, moreover, one of considerable variation as to the type of structured technique being used, with no obvious patterns across different types of data processing software-producing organisations. There does not appear to be a comparable survey of use in the more technical spheres of military, scientific, process control or telecommunications software production. Here usage may be assumed to have been rather lower, however, because of the need to economise on memory space for real-time applications.

This uptake of a great variety of structured techniques does not, however, appear to have resolved the problems experienced in the software industry. A survey of the state of the European software industry characterised its problems as follows, in 1983:

"(a) There is a wide range of approaches and little sign of agreement on why there are differences, or whether these differences are important.

(b) Communications between the different groups and individuals involved [in system development] are often inadequate.

(c) Cost and time over-runs are frequent.

(d) Maintenance costs absorb a high proportion of the total resources available to an organisation.

(e) Documentation from various phases is either hard to understand or absent, or inadequate.

(f) Users (customers) of the process don't get the system they thought they were getting."

(Perring 1983: 15)

Whilst these deficiencies were often attributed to a lack of adherence to structured standards by organisations purporting to use them, the late 70s and 80s saw further exploration of software production technology designed to deal with the industry problems. On the one hand, IBM pursued the 'structured' path with an extremely mathematically rigorous system development methodology used for its largest US projects. Here design and coding were not only carried out according to mathematically expressed standards, but the correctness of each piece of design and code had to be proved by formal mathematical logic. (Perring 1983: 18)

On the other hand, a handful of newer approaches sought to tackle the issues of getting better user specifications in the first place, or of being able to alter a system under development more easily, once it became apparent that the original specifications

were inaccurate or ambiguous. New techniques emerging in this period included:

- Languages for specifying user requirements (RSLs)

- Prototyping, in which a hastily assembled 'working model' of the basic features of the system is rapidly produced, and then experimented with by the user, to produce a specification for a more rigorously developed system.

- Applications Development Tools (ADTs) and '4th generation languages' which, for a limited range of applications, allow a user to specify their requirements directly to a computer system, which performs the remaining stages of development automatically.

- 'Software Tools' which automate various of the stages of development once requirements have been analysed, and which allow rapid iteration between phases.

- 'Programming Support Environments' (PSEs) an integrated set of software tools designed to support system development seen as a repeated and iterative process, since user requirements are generally ambiguous and changing (Lehman 1981).

The last two above have in particular been associated with the notion of 'software engineering', of establishing a comprehensive set of tools and standards to guarantee the quality of software development (Böhm, 1977).

By the early 80s, the use of most of these newer techniques was, however, much less common than of structured techniques. PSEs and RSLs were in their infancy and Program Generators - which produced code from design - were used in only 14% of data processing installations (Friedman and Cornford: 1982, P.15). The simpler forms of ADTs, Report Generators, with which users could request reports from a database, were however in use in 67% of organisations taking part in the above survey.

On the basis that structured techniques were the predominant development in software production technology affecting software producers in the early 80s, we now turn to a consideration of their effect on the nature of work.

The Impact of Structured Techniques

In terms of the effect of structured techniques on the work process of software developers, the evidence available from the literature is somewhat ambiguous. Some more sociological commentators, notably Kraft (1976) and Greenbaum (1979) see the techniques as a routinisation and de-skilling of software development work, designed to decrease labour-time and the cost of labour necessary for producing software. The data they themselves use to establish this, and the way the techniques were represented in the software engineering literature, suggest that this was, however, at best a simplification and at worst, inaccurate.

Greenbaum is of the opinion that in the USA structured techniques were introduced by managers primarily to increase their direct control over the work of individual software developers, to make progress in their work more visible by spelling out the steps involved. She also sees structured programming as a means for permitting the replacement of fully-skilled programmers with less trained and lower paid 'coders' whose job was simply to code the low-level designs produced according to the rules of structured programming. She concludes that:

> "Structured programming does seem to remove decision-making and therefore skill and control from programmers." (1979: 81)

There is some evidence from the software management literature that structured techniques were indeed seen as an aid to project control through making progress more visible. Structured design was seen as making the progress of individuals and teams more

visible to management, because it specified stages of development prior to the production of actual code. These design stages had definite outputs - bubble charts or data structure diagrams, which could be assessed and monitored. There is furthermore acknowledgement of a difficulty with structured techniques, in that software developers were likely to resist using them, on the grounds that they restricted flair and creativity through setting out definite rules of program design.

However, this is not the same as an indication that structured techniques served primarily to take control over decision-making in the realm of the approach to work away from direct producers of software, so as to increase direct management control over the quality and pace of work. The data gathered from data processing software producers by Greenbaum herself suggests a more complex picture, where the restrictions of the new techniques were on the one hand resented, whilst on the other hand they were seen as giving producers more control in at least two senses:

> "Certainly programmers complain about the restrictions of the structure but they also enjoy being sent to training class to learn a new technique - particularly one demanded in the labour market:
>
> 'At first I thought it was tyrannical, but I have gotten to really like it. You see, as a programmer, you often have to mop up after other programmers' mistakes; debugging others' programs and modifying programs of people who have left. Programs written in structured code are much easier to modify. It's also easier to help each other with our programs because you can follow the logic in someone else's program'."
>
> (Greenbaum 1979: 82)

Clearly, programmers could expect to increase their standing in the labour market, and also to have an improved understanding of maintenance tasks. The notion that structured techniques were a

way of introducing cheaper labour to do the more routine tasks is also not substantiated by Greenbaum's data. Her account of the technique is quite consistent with 'programmers' still performing both design and coding, but in a more rule-bound way.

It is also significant that, in their 1981 survey, Friedman and Cornford reported that many data processing software managers found that the Jackson method did not "reduce the time required to write programs initially by very much. In fact the design time involved in programming is increased ... . The most notable gains however ... are the improvements in productivity in maintenance of programs." (Friedman and Cornford 1982: 13)

It would appear that the effect of structured techniques on the nature of software production was a complex matter, and depended on work organisation. From the point of view of the producers, there is some indication that control was lost along some dimensions and gained along others. This is clearly an area where conceptual clarification and empirical investigation are needed. For the moment, it is important merely to note the lack of evidence to support the universal application of Kraft's assertion that:

> "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 standardised product made in a standardised way by people who do the same limited tasks over and over without knowing how they fit into a larger undertaking."
>
> (Kraft 1977: 59)

We now turn to the data available concerning initiatives in work organisation that have been documented from 1970 onwards, to fill out our understanding of the situations in which structured techniques were introduced.

3.3 DEVELOPMENTS IN ORGANISATION IN THE 1970s

The 60s and 70s saw the consolidation of the basic team form of work organisation in all types of software development. In the conventional team, the day-to-day functioning and administration of the team's work revolved around the team leader. Within this framework there were however a variety of developments in work organisation, described in the software management literature of the period.

The first area of variation suggested by these accounts is the way in which tasks were divided up between different job titles. Particularly in large organisations there often occurred a separation out of higher levels of program design, the stage immediately following requirements analysis, from the sphere of the programmer. These would be performed by a specialist system designer. By the end of the 70s, then, business data processing workforces sometimes consisted of three specialisms - systems analysts, system designers, and programmers. System design was however just as often simply added to the responsibilities of systems analysts, without the creation of a third specialism.

According to Friedman and Cornford's 1981 survey, moreover, 28% of data processing organisations used a combined programmer/analyst job description (1982: 13). This resurgence of one job responsible for all stages of development is commonly attributed to emergence of many small development projects for microcomputers.

In non data processing application areas, two broad specialisms, system designer and programmer, were fairly standard. In general, though, occupational boundaries continued to be blurred: "in a sector in which highly structured firms are the exception, and very busy people are the rule." (NEDC: 1980). In particular, there is little data available as to whether analysts, system designers and programmers were staff at different points in their career, or separate occupations and

labour markets.

In terms of how different specialisms were combined within a team, there was again considerable variety. There were, however, some basic options for making up a team. A description of each is followed by their frequency of occurrence in Friedman and Cornford's 1981 survey (1982: 13):

functional: e.g. a team of data processing systems analysts working on a number of inter-related software systems to be used in a company, or a team of programmers working on a number of different communications systems (25%);

product-based: e.g. a team consisting of systems analysts, systems designers and programmers, or analyst programmers all working together on various stages in the development or modification of a particular commercial data processing system (75%).

This lack of standardisation in the division of labour was (and continues to be) reflected in an extremely confusing lack of standardisation in job titles themselves. Even across organisations producing software for a particular type of application area, there emerged a great variety of job titles, with no consistent meaning or set of responsibilities for any particular job title. People who were principally concerned with system design might be called system designers in one organisation and computer analysts in another. In commercial data processing, a systems analyst might be responsible only for initial requirements analysis, or on the other hand might also perform the initial software design and even do some low-level design and coding.

According to Weinberger (1979), organising software development in conventional teams worked well provided that team leaders were technically competent, and rewarded good work appropriately. However continued concern with project management problems of poor quality and cost overruns led to the development of a number

of more formal monitoring techniques for team leaders to use.

(a) **the use of worksheets**, to record allocations of work to individuals in the team. Such worksheets generally record the amount of time spent on tasks against estimates. Friedman and Cornford (1982: 19) found use of worksheets in 76% of the data processing organisations they surveyed.

(b) **Keeping records of failed test runs** of modules or portions of modules produced by individuals. This is a mechanism for assessing how accurate the work of an individual is, to be called upon in performance reviews. Friedman and Cornford found this practice in 49% of data processing organisations (1982: 19).

(c) **Producing documentation at the end of phases** of the development process, so that intermediate outputs are readily comprehensible and more easily inspected by the team leader for errors. Friedman and Cornford found formal documentations in 85% of data processing organisations they surveyed in 1981 (1982: 17).

There are also a group of more sophisticated and detailed procedures that were advocated as ways of collectivising reviews of work, and making quality assurance of the outputs of intermediate phases more effective. There were found by Friedman and Cornford in 78% of the data processing organisations they surveyed, and were:

(a) **Structured Walkthroughs**

The concept of the 'structured walkthrough is generally attributed to IBM in the USA, and emerged in the early 70s. It describes an arrangement by which a team member responsible for a piece of design or coding is supposed to convene a 'walkthrough' group to assure the quality of the product. This becomes a group responsibility, rather than that of the team leader alone. Those in attendance are

generally other members of the developer's team, possibly with some additional peers from other teams. An important feature is that there is no management presence, and no employee evaluation during the review.

(b) **Formal Technical Reviews of Development Phases**

This is again a group review procedure applied to the products of development phases, advocated by Freedman and Weinberg (1977). It is similar to a structured walkthrough in that only 'peer' software producers are present, and the aim is for there to be a collective responsibility for quality. However, the remit of the review is more comprehensive, including suggestions for improvements as well as criticisms. The review is led by a facilitator who is supposed to ensure active participation by all, and the result is a written report on the phase, authored by the whole group.

Project management concerns also led to two more far-reaching initiatives in work organisation, which involved moving away from the conventional team system altogether.

(a) <u>The Egoless Team</u>

The term 'egoless team' was coined by Weinberger in 1971 to describe a form of team organisation intended to eliminate the conflict inherent in conventional teams between cooperative working and individual evaluation. Weinberger's description (1971) has much in common with developments documented in IBM in the USA at about the same time. The difference from a conventional team is in the mode of assignment of work, its control and evaluation. These three functions are all performed collectively by the whole team, which operates without a formal leader.

The 'egoless team' as described by Weinberger is in fact very similar to the 'semi-autonomous team' advocated in management literature under the headings of 'employee involvement' in the 70s and 'quality management' in the 80s. The team collectively decides how to allocate work and meets regularly to monitor progress. Each individual is responsible for quality of and rate of output of the others, and so is encouraged to be constructively critical at all times, and cooperative when lateral interaction is required. The basic rationale for this form of work organisation was that it first encouraged cooperation between members, seen as important because it was not possible to separate out software development into discrete tasks, and second that it encouraged collective responsibility for quality assurance.

According to Schneiderhan (1979), this arrangement worked well in the short run with groups of individuals of roughly equal seniority and technical competence. If everyone felt they could contribute equally and that their performance was assessed by the output of the team, they were quite likely to work effectively and cooperatively. There were, however, two major sorts of problems.

First, over time, more skilled and experienced team members tended to resent the fact that their contribution received no more recognition than that of less experienced colleagues who they were in effect training. This resentment was particularly likely to arise if promotion within the organisation was based on individual assessment. Although junior and senior team members alike could find the egoless team stimulating in the short term, senior members apparently soon began to feel they were selling their career prospects short. The second problem is related. Even in the short term, individuals might perceive that a withdrawal of cooperation and effort had little effect on the output of the team as a whole, and take to 'free-riding'.

(b) The Chief Programmer Team

The idea of the chief programmer team was set out by Harlan Mills at IBM (Baker: 1972), as a response to two problems inherent in conventional teams. First, the observed 10:1 range of productivity between programmers of different levels of experience and ability, and, second, the problems of communication between many teams involved in a large project. According to Brooks, the most important failure in large software projects stemmed from failures to communicate over changes in the overall design in one area which made designs or code produced in other areas faulty. He saw 'conceptual integrity [as] the most important consideration in system design' (Brooks 1975: 42).

The basis of the chief programmer team is to concentrate all the direct software production in one team in one mind, along the lines of a surgical team. The 'surgeon' defines requirements, designs the programs, codes, tests and documents them. The surgeon is directly assisted by a co-pilot, who discusses the surgeon's work with him or her, researches alternative design strategies, and represents the team in its external contacts. In addition, Brooks (1975, P.32) suggests an administrator, a documentation editor, a programming clerk, who keeps records of test runs, and a toolsmith, who provides software tools specified by the surgeon. According to Brooks, this ensures that there is maximal conceptual integrity between the phases of software development, and simplifies communication between teams, because one or two minds within each team know everything that is going on.

Brooks extended this principle to an organisational structure for large software projects. He recommended a small team of 'system architects' as the best way of producing a highly integrated high-level design. Once the overall design was complete, its modules were to be passed out to a set of 'chief programmer' teams for implementation.

To take account of ambiguities in the overall design and the need to make modifications to it during implementation, he advocated regular - say weekly - conferences between the system architects and representatives of each implementation team. This group would then publish details of modifications for the project staff as a whole.

The Impact of Initiatives in Work Organisation

There does not appear to be any data available to assess the impact of the review mechanisms introduced to enhance the working of the conventional team. Like the various structured techniques, they represented a formalisation or codification of how certain elements of the software production process should be conducted. It is, however, not clear whether their widespread introduction of itself threatened or supported a 'high-trust' basis of work organisation, whether it produced conflict, or had consequences for the respects in which various types of software producer had control over their work.

The egoless team does not appear to have attained much actual usage in software-producing organisations. It has had in fact little impact on the development of software production work and the occupations involved. Its advocacy by Weinberg (1971) and Schneiderhan (1979) does, however, indicate the existence of problems associated with the conventional team, of encouraging cooperation, particularly in terms of taking responsibility for quality assurance.

The Chief Programmer team also seems to have attained little actual usage. Greenbaum (1979: 123) attributes this to the fact that programmers were unwilling to have patterns of communication imposed by management. They valued setting up their own informal channels for each project. It is also far from clear what its impact on the nature of software occupations was in cases where it was used. Brooks (1975) implies that 'surgeons' should always be the most experienced producers, preferably with at least ten

years experience. This implies that the other, supporting, roles within the team were seen as less responsible positions through which more junior staff could acquire experience. It is, however, not apparent from Brooks' account whether he saw all junior producers as eventually making the grade to 'surgeon' level, or whether he saw some remaining as 'technical assistants'. In principle, the strong vertical division of labour in the 'chief programmer' team, is consistent with both greater integration and greater segmentation of software occupations. The more integrated version would be where all producers expect to gain experience in a number of different support roles, and eventually become 'surgeons', responsible for all stages of development work. The more segmented approach would be where selected junior staff were given training and experience necessary to become 'surgeons', whilst the remainder were kept specialised in particular less responsible support roles.

3.4 THE STATE OF SOFTWARE OCCUPATIONS IN THE EARLY 1980s

The overall picture of the software producing occupations at the beginning of the 1980s is first of all one of a numerically expanding but increasingly fragmented set of specialisms, where considerable business pressures have led to increasing formalisation of work techniques, forms of reporting and quality control.

Fragmentation has occurred along a number of dimensions. First there was an increase in 'horizontal' specialisation as the increasing scope of application of computer systems and increasing complexity of systems software opened up the need for new software specialisms. Second, even within one area of software production, there was increasing variety of techniques used from organisation to organisation, as different types of formalised or 'structured' techniques were introduced in order to assure the quality of systems, and improve project control. Third, there was a similar variety in terms of forms of the

division of labour and organisational mechanisms used. In some cases, structured techniques and new software tools were associated with a move towards greater integration of all stages of software production within one head and occupational specialism, that of the 'chief programmer' or 'software engineer'. In other cases there was emphasis on greater vertical division of labour, with the earlier stages of design performed by more senior staff. Within the data-processing sphere there was also variety as to the roles of systems analysts and programmers, and whether these broad specialisms worked together on a project in one team or in separate teams.

There were at the same time forces producing some degree of standardisation within a particular area of software production, but not particularly strong ones. The strongest was the development of labour markets in particular application areas, which led to some standardisation of skills. The standard was usually set by the practices of large organisations - typically computer hardware manufacturers or user corporations - which undertook recruitment and training of inexperienced staff. The pattern persisted of large firms on the whole undertaking training whilst smaller firms tended to 'poach' skilled staff and not offer much training themselves. The practice of offering high salaries but no training encouraged 'job-hopping' in pursuit of better pay and wider experience, which in turn exacerbated the preference of many firms not to train.

The 70s saw the emergence of a trade association for the U.K. independent computer services sector, the Computer Services Association. A strong lobby developed within this to encourage firms to train, in order to avoid skill shortages which were seen as leading to inflated salaries and restricting the growth of the industry, which led to the establishment of the Computer Services Industries Training organisation (COSIT). Even in the late 80's this has however not been successful in establishing any voluntary regulation of training volumes or training standards.

The lack of any regulation of training standards through an

industrial body has not been compensated by the emergence of an effective occupational body or set of them. In the data-processing area, the 70s saw an attempt at occupational regulation of training standards by the British Computer Society (BCS), which began to offer certification in programming and systems analysis skills and experience with its Part 1 and Part 2 membership exams. These, however, have not been seen as essential forms of certification or as useful indications of competence by either employers or software producers themselves.

In the more 'technical' areas of software production, standards as to what a degree in computer science should entail had emerged by the end of the 70s, ensuring that many graduate entrants to this sphere had a common basis. However, there was no occupational body controlling entry, and many continued to work in this area without recognised qualifications - either new recruits coming from university degrees in maths or sciences, or older employees who had acquired skills through experience.

In terms of the increasing use of formalised production techniques and monitoring mechanisms, and their consequences for the control of work and developments in the division of labour, Kraft (1977) is of the opinion that there was a general tendency operating beneath all this apparent diversity. He sees software production work as rapidly being moved away from 'high trust' principles, where workers were expected to organise their own work to a considerable degree, to predominantly direct control of work processes by management, by way of formalised techniques and an increasing vertical division of labour.

The software practitioner literature does not appear to indicate that formalised techniques and procedures were introduced as part of such a wholesale move to 'low trust' forms of work control, or a necessarily greater vertical division of labour. They indicate a range of management concerns behind their introduction, including improving quality assurance, project control, stemming or compensating for labour turnover, and making systems more easily modifiable and maintainable. However, their introduction

does raise the issue of whether they conflicted with preferences of workers to control their work, based on established practices of informal relations between team members and team leaders, and of what the outcomes in terms of the control of work and division of labour were. It is not clear from the software management literature of the period whether they were associated with 'chief programmers' and 'software engineers', or with the creation of a less skilled class of 'software technicians' doing only detailed coding work, or with both.

CONCLUSION

We have established that by the early 80s, software producers were working in a great variety of organisations - for example, firms primarily concerned with manufacturing computer hardware, independent software houses and manufacturing and service-providing organisations that were major user of computer systems. Moreover, these organisations had a great variety of business environments, product mixes, and competitive strategies, from competing through providing a comprehensive range of hardware and software products and services to competing on the strength of the quality of customised systems supplied in a particular specialised application area, or competing in terms of cost in supplying standard packaged systems.

It appears then that there were potentially a great variety of business goals underlying the various management control concerns giving rise to initiatives in software production technique and work organisation. As well as the issue of how such initiatives affected the nature of conflict over the control of work and the evolution of the division of labour, there is also the question of how such initiatives were related to different business goals and concerns.

In the light of this, the next two chapters are concerned with

developing a conceptual framework to guide empirical research concerned with the nature of conflict over the control of work and its relation to business performance requirements at the level of the enterprise. This chapter has served to show the range of business goals, management concerns and initiatives in technique and work organisation which must be encompassed by such a framework.

CHAPTER 4

PROFESSIONAL TECHNICAL WORKERS: CONFLICT AND THE EVOLUTION OF OCCUPATIONS

INTRODUCTION

The previous two chapters have established that software producing occupations emerged through the 60's and 70's as relatively high status white collar technical employees, employed by business and public administration organisations, of a great variety of types. They are identifiable as 'professional technical', a term commonly used in the literature of the sociology of work and occupations to refer to workers who are responsible for complex technical tasks within business or public administrative organisations. They enjoy relatively high status and levels of reward, and are also identifiable as members of an occupation because of basic common areas of expertise or technique used across different employment situations.

Over the last thirty years such occupations have been the subject of a great deal of theorising and a regrettably much smaller volume of empirical research. Study has focussed on the one hand on their similarities and differences with the 'established' or 'pure' professions, as typified by medicine and law, and on the other on the meaning of the concepts of 'profession' and 'professionalisation' in describing the development of these newer 'organisational professions'. There have also been a series of attempts to characterise the relationships of interdependence and conflict between professional technical workers and the managers of the organisations that employ them.

This chapter is concerned with an examination of various conceptualisations of conflicts between 'professional technical' workers and managers at the level of the firm, particularly in terms of the tensions between managerial, occupational and individual control of work. It also looks at how these different conceptions of conflict illuminate the evolution of the division of labour, forms of work organisation, modes of supervision and other aspects of the nature of such occupations. The aim is to see how adequate existing frameworks are for addressing these issues given the variety of employment situations and management concerns already identified as affecting software producers.

The first two sections concern the two main approaches to understanding professional technical worker/manager conflict. Within the predominantly neo-Weberian 'sociology of the professions', there are first of all theorists who have focussed on issues of conflict and compatibility between two different sources of authority within work organisations. On the one hand they identify 'professional' control of work and on the other 'administrative' or 'managerial' authority and control. The second section summarises how neo-Marxist theorists have focussed on a class notion of 'professional'/managerial relations. For the most part they have seen a fundamental conflict stemming from management's need to establish control over the productivity of professional technical workers. An examination of empirical work carried out in both traditions indicates that researchers have had to revise their theoretical assumptions and recognise strong elements of stable interdependence between professional technical workers and their employing organisations. A third main section of the chapter outlines a framework that conceptualises the nature of this complementarity more explicitly. On the basis of existing empirical research, it also explores how conflict can be understood as occurring within a context of shared goals.

4.1 THE SOCIOLOGY OF THE PROFESSIONS: CONFLICT BETWEEN 'PROFESSIONAL' AND 'ADMINISTRATIVE' PRINCIPLES

During the 60's and 70's much work within the sociology of the professions focussed on the meaning of the concept of 'profession', the nature of the process by which an occupation became 'professionalised' and the nature of conflict when professionals were employed within bureaucratic business or administrative organisations. The development of notions of 'profession' and 'professionalism' in this literature have been well analysed elsewhere (Johnson 1972, Saks 1983) and needs only brief treatment here, in order to bring out conceptions of relationships with predominantly bureaucratic work organisations.

'Taxonomic' and 'Symbolic' approaches

'Taxonomic' accounts of professionalism, appearing through the 50's and 60's, attempted simply to mark off 'professions' from other occupations by attributing to them a number of characteristics. One of the problems with this approach was that no single list of 'traits' seemed to fit all occupations commonly accepted as professions, and different theorists tended to come up with rather different lists (Millerson 1964). However, there was broad agreement on a core set, summarised by Hall (1968: 92-3) in terms of two lists of attributes, one 'structural' and the other 'attitudinal'.

Structural Attributes

1. Creation of a full-time occupation.
2. Establishment of a training school, generally affiliated to a university, in which a complex and theoretical body of knowledge is imparted to entrants to the profession.

3. Formation of a professional association with the power to accredit and expel practitioners.
4. Formation of a code of ethics, concerning behaviour towards both colleagues and clients.

Attitudinal Attributes

1. The use of the professional association as a major reference.
2. A belief in service to the public.
3. A belief that the occupation should be self-regulating.
4. A sense of calling to the field.
5. A commitment to autonomy, a 'feeling that the practitioner ought to make his own decisions

> without external pressures from clients, those who are not members of his profession, or from his employing organisation.' (Hall 1968: 93)

The last of these attributes, the commitment to autonomy, has clear implications for tensions with employing organisations and has remained an important feature in most subsequent theorising concerning professionals. A rather reduced number of traits was advocated by structuralist functionalists such as Barber (1963) and Parsons (1968). They placed heaviest emphasis on the knowledge-base of professional practice, and on the importance of its disinterested, rational and scientific application to the welfare of society. According to Barber, this functional importance was linked to a primary orientation to the community interest, a high degree of self-control through codes of ethics, and high levels of reward, in both monetary and status terms.

'Trait' and structuralist-functionalist approaches alike saw the development of 'occupation' into 'profession' in terms of a linear process by which 'occupation' acquired the attributes of a true profession. Moreover, according to the functionalist model, the attributes were acquired by virtue of their contribution to the functioning of society as a whole, and so were inherently stable. 'Professionalism' was seen as a state towards which occupations evolved, rather than as something which was inherently unstable or associated with conflict. It was acknowledged that occupations which had complex and important knowledge-bases differed as to the degree to which they currently possessed 'professional' attributes, but this was explained in terms of the age of the occupation, and the chance it had had to develop attributes. Hickson and Thomas (1969) succeeded in correlating the age of 43 professional associations with a cumulative score for the extent to which the corresponding occupation possessed various attributes of professionalism. Unfortunately, this was only made possible by leaving scores for "autonomy" and "extent to which the occupation is theoretically based" out of the analysis.

The 'taxonomic' approach appears of limited value in studying 'organisational' professions. It is based on a model of professionalism, derived from law and medicine as paradigms, where practitioners are predominantly in independent practice, relating directly to clients, but regulating their affairs as an occupation. The 'taxonomic' approach has also been criticised heavily as a sociological account of such 'traditional' professions, on two main grounds.

First, it is ahistorical, in that it proposes a model of professions and professionalisation that is supposed to be of universal applicability and not dependent on historical contingencies. In fact, as Johnson (1972) points out, occupations which have acquired the central attributes of professionalism have generally done so quite rapidly, rather than as a gradual process, and as a result of considerable conflict with other interests and in quite specific historical circumstances. A second problem is that the approach takes as fact the content of professional associations' codes of ethics, which declare ideals of disinterested service to the community, without considering whether these ideals are actually professed by members, still less whether their behaviour can be construed as being in accordance with them. Daniels (1975: 305) argues that many professional attributes can be seen more as ideologies that justify status and rewards than as fact, and need to be empirically verified: "Social workers and school teachers, for example, often complain that they learn little or nothing in their formal training, but it is an ordeal required of them in order to acquire the certification that enables them to practice."

Saks (1983) attributes the development of more critical views of professionalism partly to the intellectual shortcomings of the 'taxonomic' approach, but also to the development of an 'anti-professional' climate, particularly in the U.S.A. The foundations of a new approach were laid by Becker (1962) and Hughes (1958). Departing from a notion of 'professionalism' as an objective set of characteristics, Becker found it more useful to

conceive of it as a symbol, or 'folk concept', which members of an occupation appealed to in claiming certain privileges for themselves.

The nature of the 'folk concept' was as follows. A profession is an occupation that possesses a monopoly over a complex and esoteric body of knowledge, the application of which is of social importance. Its members are guided by altruism and ideals of service, and subscribe to a code of ethics in their relations with colleagues, with members of other professions, clients and the public in general. Most importantly, on the basis of their monopoly of knowledge and their ethical code, members of a profession also have control over their occupation. Collectively, they and only they, control training and accreditation, define the nature of problems to be tackled, the way they should be tackled, and evaluate each other's performance in doing so. Finally, "Members of the profession are usually thought of, as they often are, as people of sizable income and high community prestige ... (they) are considered to be entitled to an important voice in community affairs ... (and their) associations are thought of as important public institutions" (Becker 1970: 46).

According to Becker, the importance of this 'folk concept' is in its use as a standard for judging the worth and status of occupations. Occupations commonly present themselves to the world as though they are organised and behave according to it, even when the reality is very different. This presentation justifies their autonomy and control over their own work. Hughes also sees the idea of professionalism as 'not so much a descriptive term as one of value and prestige' (1975: 210). According to both Becker and Hughes, what marks 'professions' from other occupations in reality is simply that the former have been able to gain acceptance for their use of a badge called professionalism. They delight in showing that in terms of actual behaviour and organisation, similarities are as common as differences between so-called professions and non-professions.

This approach opens up possibilities for understanding 'professionals' working in the context of organisational employment. If, for example, the claim to autonomy and control over work is more ideology and a bid for social status than reality, it may be that occupations calling themselves professions can co-exist with considerable employer control over recruitment, work goals or practices. The field is opened for considering 'impure' forms of profession, which have some of the pure 'traits' but not others, as phenomena in their own right, instead of mere stages on the way to true autonomy and occupational control.

'Social Closure' Approaches

Neither Becker nor Hughes consider in any systematic way the conditions under which occupations gain the title of 'profession', the forms of occupational control that are actually established, and their relation to forms of control of recruitment, training, work, and evaluation from outside the profession. By introducing the idea of 'professionalism' as an ideology that an occupation can adopt for essentially political ends, they do however lay the basis for what Saks (1983) called a 'new orthodoxy' in the sociology of the professions. This school deals directly with such issues, although it also came to do so by concentrating initially on the traditional professions, predominantly independent practitioners dealing directly with members of the public as clients.

According to Saks, the Weberian concept of 'social closure' has either explicitly or implicitly underpinned a number of accounts of 'professionalisation'. Social closure refers to a process by which a collectivity achieves regulation of market conditions in their favour. In this literature, professionalism is seen in terms of occupational groups mobilising to gain legalised control over significant aspects of their work or work situation. Professions are occupations which are "legally privileged groups which have managed to monopolise to a considerable degree social and economic opportunities" (Saks 1983: 6). There have been a

number of different formulations as to exactly in which respects professions have achieved control or monopoly.

Closest to the Weberian emphasis on controlling markets is the analysis of Parkin which sees professionalism as a strategy for restricting and controlling entrance to an occupation responsible for administering an important knowledge-based service. Such a strategy protects or enhances the market value of the occupation (Parkin 1979: 54). Larson offers a very similar picture of the rise of the 'traditional' independent practicing professions such as medicine and law: "Professionalisation (is) the process by which producers of a special service sought to constitute **and control** a market for their expertise. Because marketable expertise is a crucial element in the structuring of modern inequality, professionalisation appears **also** as a collective assertion of special social status and as a collective process of upwards mobility ..." (Larson 1977: xvi-xvii). This notion of professionalisation is elaborated by Larson in detailed analyses of how the medical and legal professions organised themselves in nineteenth century America, using ideologies of professional autonomy. They built alliances with the new higher educational institutions, in order to close off access to their occupation and control the training of entrants. Thereby, they protected their value in expanding middle class markets for their services, at the same time giving guarantees as to the quality of practitioners' skills as justification for their monopoly. This strategy ensured respectable middle class levels of reward and status. The same emphasis on professionalism as upward mobility appears in the work of Parry and Parry, concerned with the medical profession in the UK (1976).

Johnson (1972), Friedson (1970), and Davies (1976) offer slightly different versions of the social closure theme, but still related primarily to an analysis of the nature and rise of the 'traditional' professions. Although he subsequently modifies his position, in his earlier work (1972) Johnson focusses on 'traditional' professionalism as one way of resolving the inherent 'social distance' or uncertainty between suppliers and

consumers of services based on specialised knowledge. He sees professionalism as a form of control of service provision where the provider is legitimated in defining the need of the consumer, and the way the service should be provided. He, too, emphasises the importance of understanding the ideological and material weapons mobilised by an occupation to gain such control over its work. Friedson and Davies concentrate on the control of the conduct of work as the aim of professionalism. Again, professions are seen as occupations which have made their control legally legitimate, backed by an ideology that the profession assures its own standards and has a commitment to public service. According to Friedson, once it was accepted that a profession was merely an occupation that regulated its own work, one could then investigate whether or not other attributes it claimed, such as adherence to its ethical code or a complex knowledge-base, were in fact true.

The persistence of much of this literature with ideas of closure, justified by the status of the occupation and at the same time instrumental in maintaining it, is arguably one reason why the 'organisational' or 'technical' professions are poorly represented within it. Professions such as engineering are recognised as having, or aspiring to, some of the trappings of the established professions, particularly in terms of the social status and levels of reward associated with applying a complex body of technical knowledge. However they are also recognised as having much lower degrees of legalised control over recruitment, training and standards, in both the USA and the UK, than medicine or law. Professional engineering institutions exist in both countries, but employers are clearly at least as influential in controlling most of these areas (Whalley 1986: 5). The idea of social closure, of legally legimitated areas of occupational control simply does not fit that well with the reality of industry-based engineering occupations.

Writers in the 'social closure' school have again related professional technical workers, such as engineers working in industrial corporations and software producers, to the notion of

professionalism by depicting them as half-way there. These occupations are seen as weakly organised, less evolved versions of true professions. This association of professional technical workers with established professions often occurs in the context of discussions of the nature of the relationship of professionals to organisations that might employ them, to which we now turn.

Professional Organisation in Conflict with the 'Administrative Principle'

During the 60's and 70's it was recognised that many traditional professions as well as the majority of 'professional' groups emerging during the twentieth century were commonly employed by business or public administrative organisations, managed by non-professionals, as opposed to offering their services to the public from independent practice. It was also recognised that 'professionals' often worked from within 'professional organisations' such as legal or architects' practices. The increasing importance of organisational employment for even the traditional professions of medicine and law led to a number of attempts to consider the relation between professions and bureaucratic organisations. Generally these discussions started from data gathered in relation to the more clearly self-regulatory professions. Newer 'technical professions' such as the various branches of industry-based engineering were often also included, regardless of whether or not they had attained the same type of legalised occupational control.

Blau (1969) and Drucker (1970) formulate a fundamental clash between, on the one hand, occupational authority, meaning reference to professional standards, established by occupational institutions independently from any particular employer and, on the other, the bureaucratic hierarchy of authority within a non-professional or 'heteronomous' organisation. According to the Weberian 'ideal type' of bureaucracy, tasks to be performed at a given level in the organisation are specified in terms of objectives and means by people at the next level up, who are also

the arbitors of whether work has been performed satisfactorily. Both Blau and Drucker see professional autonomy as in conflict with such bureaucratic authority.

Gouldner (1957) formulates a similar divergence of loyalties in terms of labour market aspirations. When employees work in a job where standards are set and evaluated by management within their organisation, they look to managerial evaluations, identify with organisational goals and value advancement with the organisation. Gouldner describes them as 'locals'. In contrast, a 'cosmopolitan' labour market orientation is associated with employees in occupations where standards are set and evaluated by occupational bodies. Such employees value their standing in their professional community first and foremost, and are interested in the goals of their employing organisation and their position within it only in so far as it gives them opportunities for professional eminence.

Notions of the nature of clashes between professionals and employing organisations and their consequences are developed further by Daniels (1975). Taking examples mainly from medical, welfare and educational professions, she analyses when there is a potential for conflict between professionals and managers of their employing organisations. She concludes that, provided the goals of professionals' work, the means employed to meet them, and the relationship between the two are well specified and clearly understood, there is no problem in accommodating professionals to the sort of evaluation required by a bureaucratic organisation. However when goals, means and the relationship between the two are not clear, and particularly in times of economic stringency, there is potential for conflict. Conflict may concern the nature of goals, means or the level of resources to be made available by the organisation for professionals to do their work. Bureaucratic managers are, for example, likely to question the validity of goals set by professionals and see the means they require as extravagant. In such cases, "... bureaucratic power and professional power are in contest, and the relative power of the professional group vis-vis

the organisation decides the extent to which bureaucratic control can be enforced" (Daniels 1975: 328). In brief, when times get hard, bureaucracies try to break professional control over goals of work undertaken and the approach used.

Daniels sees the situation of 'computer software experts' in precisely these terms. She refers to the introduction of new software development techniques in the mid-70's as an example of a situation where, because of the fall in the demand for software skills in the early 70's, members of the occupation were "... in a poor position to assert their authority and techniques" (ibid: 329). She sees software producers as a weakly organised profession, which, in a situation of weakened market power, is unable to stand up to the inevitable desires of bureaucratic managers to specify the way their work should be done. Structured techniques become instruments of managerial rationalisation.

Professional Technical Workers in 'Post-Industrial Society'

There is an alternative and influential stream within the sociology of the professions which also starts from an assumption

of a degree of incompatibility between professional control or autonomy and bureaucratic managerial control. It sees however a very different resolution, in terms of strengthening of professional power at the expense of managerial. It is articulated above all by Freidson (1973), although a very similar argument is put forward by Bell (1973).

Freidson explores the relevance of his conception of 'professionalism' - an occupation which controls the means and methods used by its practitioners, and the standards used to evaluate them - to issues in the coordination and control of modern work organisations. On the understanding that advanced western economies and social institutions are undergoing a fundamental transformation towards "post-industrial society", as

described by Bell (1968), he concludes that professional technical workers are likely to become increasingly professionalised. The argument is of particular relevance here because it explicitly refers to systems analysts and data processing. It leads to hypotheses about the future development of such occupations, and their relationship to business management.

Freidson starts from the position that since the industrial revolution, that is, in "industrial society", the organisation of work has been based increasingly around "administrative authority". Workers occupy jobs and positions, rather than belong to occupations. Such jobs are "created, dissolved and reconstituted by management on the basis of changing production goals or needs, changing technology, or further rationalisation for the end of greater productivity or lower cost" (Freidson 1973: 52). By virtue of their positions as "officials of the formal productive organisation" (ibid: 50), managers, administrators or supervisors, define goals and the way work should be carried out to meet them. They also exercise authority over workers in assigning tasks and evaluating performance.

According to Freidson, this principle of organisation is inadequate to cope with the management of the increasing number of knowledge-based workers in the economy. He takes from Bell (1968) and others the idea that in advanced economies complex theoretical skills and knowledge are replacing straightforward manual and craft skills. This is because science and technology are being applied to automate more and more production processes, and knowledge-based services - research and development, education and medicine - are becoming the core of productive economic life. Freidson adds welfare occupations to a list of industrial knowledge-based workers, such as engineers, economists and systems analysts, as characteristic of the 'post-industrial' era. What they have in common is "practising complex skills for which higher education is thought necessary" (ibid: 49).

Freidson sees two problems with managerial direction of knowledge-based workers. Complex abstract knowledge and skills are not easily subject to administrative specification and evaluation, given that they take years of extra-organisational training to master. In addition, during the process of higher education, future employees develop an identification with their skill and occupation. Tasks, the way skills are applied, and the way practitioners are evaluated become formulated by the occupation, not by management. Workers formed by such occupational processes become able to perform their work without management direction. Faced with workers who have an occupational identity in this sense, management have to take skills "more or less as given", rather than attempt to simplify or fragment them.

The problem that emerges is how work should be controlled and co-ordinated by management to meet its goals, once it has lost the ability to direct work within the realm of particular knowledge-based occupational specialisms. Freidson's answer is that professional technical occupations, such as engineers and systems analysts will become increasingly professionalised in the institutional sense, on the model of medicine. They will become organised as occupations to such a degree that members can envisage "a career during which they retain a particular occupational identity and continue to practise the same skills no matter what the institution they work in" (Freidson 1973: 54). The institutions of these professions will then regulate the means used to achieve goals set by management.

Freidson is further concerned about who will coordinate different areas of work, on the assumption that the importance of 'knowledge-based' work means that specialisation and the division of labour continue. He poses the question: "How can any regulation exist between professions without the monocratic authority of management?" His answer is again derived from the example of how doctors relate to other occupations. Professionals themselves will increasingly manage their own boundaries, coordinating their work with that of other occupational spheres.

In sum, professional authority is held by Freidson as a superior alternative to managerial authority for the organisation of 'post-industrial' work. The growth of 'knowledge-based' work is an opportunity for true professionalisation of occupations responsible for it. This view of the responsibility for the management of industrial activity passing from administrative managers to the technically-skilled has many parallels with the shift in power towards the 'technostructure' forecast by Galbraith (1967). Galbraith does not however link the increasing power of engineers and other professional technical workers so explicitly to the rise of occupationally-based institutions. He simply sees technical experts taking over organisational positions formally occupied by generalist business administrators.

This sort of argument suggests the following understanding of relations between software-producing occupations and their managers, and for the evolution of software occupations. Since software producers are undertaking complex and important technical work within organisations, provided that it is not possible for management to simplify their work, members of the various specialisms will continue to require higher education and advanced skills. In the course of this they will develop, at least informally, an occupational identity and set of standards. Management's inability to specify and control this work in detail will be an opportunity for the development of occupation-wide institutions to control training and formally inculcate knowledge standards and values as to the way work should be performed and evaluated. In time, professional modes of organising for each specialism will devise ways of managing the interfaces between themselves. For example, systems software producers, applications software producers and electronic hardware engineers will develop their own standards and understandings for working together.

The result is not so much that bureaucratic authority will have to accommodate to professional authority with which it is in

conflict. Rather, bureaucratic authority will be largely eclipsed by professional control, at least in those organisations where work is intellectually based. According to this view, the introduction of structured techniques might be seen as the technical basis for occupation-wide standards as to how work should be approached, and for defining the interfaces between professions, such as systems analysis and programming in an orderly way.

Bureaucratisation and Professionalisation as Complementary

The views that organisational management seek to bureaucratise professional work and that professional organisation will take over many of the functions of controlling and coordinating work organisations share a common premise, which limits their applicability to professional technical workers. Both assume a basic incompatibility between the occupational identity of technical professional workers and the concerns of organisational management. This has in fact been born out only partially by empirical evidence.

Hall (1968) presents a classic study of the extent to which a number of professions and what he regards as aspiring professions - including engineers and advertising executives - become bureaucratic when employed in different types of organisation. He measures the extent to which their work is bureaucratised in these different settings, according to a set of dimensions derived from Weber (Hall 1968: 95). In general he finds that professionals employed in 'heteronomous' organisations, mainly non-professional managers and in 'professional departments', such as legal sections, of larger organisations have more bureaucratically organised work than professionals working in autonomous organisations - such as legal practices - controlled by members of their own profession. However, in some dimensions of bureaucracy, professional departments are no more bureaucratic than autonomous organisations. Hall concludes that professional

standards can fit within the context of organisational control, rather than clashing with it. Indeed "... the professional's self-regulatory patterns may reduce the need for and utility of organisational rule systems." Hall sees organisations adjusting to take account of the standards that professionals bring in from outside. When new organisations are set up this can probably occur without conflict. However, he acknowledges the likelihood of conflict when professional standards are brought into an existing organisation.

The study also measures to what extent members of the various occupations ascribe to typical 'professional' beliefs about their work. When correlated with the extent to which work is bureaucratised, the picture is again uneven as to the extent to which typically 'professional' attributes clash or harmonise with typically 'bureaucratic' conditions. Whilst the 'feeling of autonomy' was less when work was more bureaucratised, having a professional reference was found quite compatible with working under a rigid hierarchy of authority. The bureaucratic use of technical competence as a basis for selection and promotion was also found to fit well with professional values. Such results led Hall to posit a degree of interdependence between professional standards and bureaucratic organisation, rather than inherent conflict:

> "... the implication is that in some cases an equilibrium may exist between the levels of professionalisation and bureaucratisation in the sense that a particular level of professionalisation may require a certain level of bureaucratisation in order to maintain social control. Too little bureaucratisation may lead to too many undefined operational areas if the profession itself has not developed operational standards in these areas." (Hall 1968: 104)

Studies of technical professional workers by Miller (1968) and Ritti (1971) further provide evidence that their labour market

orientations are by no means at odds with organisational loyalty and goals. Miller finds that graduate engineers in a U.S. aereospace concern lack a definite image of their professional tasks, are willing to work in a broad range of technical jobs, and characterise success in the "technobureaucratic" terms of getting into management. Ritti finds similar orientations amongst development engineers in a large U.S. concern in the late 1960's, engaged on designing large and complex electromechanical control systems for weapons guidance and telecommunications switching.

Ritti finds that engineers on the whole identify with the goals of their employing organisation, because they are oriented towards creating technical systems to be marketed and used, rather than developed for their own sake. He sees the engineer's emphasis on moving into management as the result of "... not so much a desire to manage people - to supervise others - as to achieve an organisational status that permits him a wider latitude of technical involvement" (Ritti 1971: 62). Ritti concludes that the basic harmony in goals between engineers and their employing organisation and the greater technical influence accompanying organisational advancement mean that development engineers are as much 'locals' as 'cosmopolitans'. At the same time he recognises that conflicts can still arise between 'local' engineers and business administrators over how to achieve agreed business goals, e.g. which technologies to use (ibid: 62).

The view of an inherent clash between professionals and organisational management is also based on an erroneous characterisation of 'non-professional' organisations as ideal-typical bureaucracies, unable or at least uncomfortable about controlling activities that cannot be fully specified. The organisational literature indicates that there are many ways that managers have found of controlling tasks too complex or uncertain to be fully specified. Beginning with Burns' and Stalker's (1961) conceptualisation of 'organic' as opposed to 'mechanistic' forms of organising, many alternative modes of organising have been described. For example, sub-units may be controlled

relative to outputs rather than according to the means used to achieve them (Galbraith 1977, Mintzberg 1979) and workers may be coordinated through mutual adjustment rather than hierarchical direction (Thompson 1967). In the 1980's, there has been a concerted move in 'progressive' management theory and education to establish an alternative model of management, more effective than traditional bureaucracy for dealing with uncertain conditions where flexibility is required (Walton 1982). This involves encouraging individuals to feel committed to the organisation and giving them responsibility to organise tasks autonomously. Managerial control and complex, knowledge-based work cannot be assumed to be incompatible.

There is a further problem specifically with Freidson's notion of increasing professional power as the resolution of tension between professional technical workers and organisational management. It assumes a natural link between workers acquiring complex skills through a training in higher education, and the emergence of professional institutions which are influential in the practice of the professions. Whilst it is true that Freidson argues that such a link is desirable, rather than inevitable, the consensus of the literature is that it has still not occurred for most branches of industry-based engineering in the U.K. (Prandy 1965, Whalley 1986). We saw in the last chapter that there is also little evidence of powerful occupational institutions existing for software production.

To summarise the argument of this chapter so far, many theorists of professional-managerial conflict have been guided by some notion of professionalisation as 'social closure', by which an occupation gains autonomy and control over the conduct of its work. These theorists have identified the potential for clashes with managers over goals of work, over the means to be employed, over the level of resources to be allocated, over standards to be used in evaluating work, over the way work is to be controlled, over loyalty to the organisation as opposed to the profession, and over the kind of advancement desired. The evidence from studies of 'professions' working in heteronomous organisations

and particularly from studies of professional technical workers such as engineers is that none of these are in fact necessary points of conflict. Goals can be shared between professional technical workers and their organisation, and organisational advancement seen as bringing occupational recognition. Managerial evaluation and control can be seen as valid and indeed necessary. At the same time, most studies conclude that conflicts can arise in these areas.

The issue emerging is of how to conceptualise the nature of interdependence and harmony between professional technical workers and their employing organisations so that an understanding of the conditions under which conflict arises can be gained. Before addressing this, in the third section of this chapter, I will examine the separate neo-Marxist school of theorising about professional technical workers, which has also focussed on fundamental conflict with organisational managers. Once more, the general direction of development has been from a premise of fundamental conflict over goals and means to a recognition of considerable areas of compatibility.

4.2 MARXIST ACCOUNTS OF PROFESSIONAL TECHNICAL WORKERS

We have seen that the neo-Weberian sociology of the professions has a lot to say about the nature of professional managerial conflict 'in the present'. Apart from Freidson and Bell, the theorists we have just reviewed offer little by way of broad analyses of the evolution of professional technical occupations over time, as these conflicts are worked out. In contrast, neo-Marxist analysts have on the whole been more interested in the historical development of these occupations, and to see conflicts between technical professional workers and managers in the context of 'longitudinal' or 'diachronic' processes.

Most neo-Marxist accounts of professional technical employees have concentrated on their class position in society as a whole, and the way this is determined by their relations to managers within capitalist enterprises. As Whalley (1986) points out, there are three distinct strands of Marxist thought on professional technical workers' relations with managers. First, writing in France, Mallet (1975) and, in his earlier work, Gorz (1967) see professional technical workers as in conflict with capitalists because of the restrictions placed on the exercise of technical skills by commercial considerations. Professional technical workers are opposed to managers because of the higher rationality inherent in advanced technology itself, and are likely to look for alliances with traditional working class organisations in opposing capitalist management. Mallet sees technical workers employed in industry as the 'new working class'. Given the sorts of positive orientations to performing technical work in a capitalist context described by Ritti (1971), this perspective would seem to have little to offer, at least in an Anglo-Saxon context, for understanding the nature of conflict between managers and professional technical workers.

At the opposite end of the spectrum, Poulantzas (1975) sees professional technical workers such as engineers as performing technical tasks within the framework of capitalism, and often concerned with designing deskilled labour processes for blue-collar workers. They simultaneously maintain capitalist control over production processes and their own privilege by employing their knowledge in the interests of management. Their expertise is subordinate to management, rather than a challenge to it. According to this view, minor conflicts may occur between managers and professional workers, but 'professional' alliances with the working class are unlikely. This view would appear to have much in common with the empirical evidence obtained within the sociology of the professions. Its abstract nature is however of limited use in understanding the detailed nature of organisational conflict.

The third position identified by Whalley is that of Wright (1978) and Carchedi (1975). Carchedi sees professional technical workers as allied in some senses to both blue collar workers and managers. When they are designing supervisory systems, they are serving a function of capitalist management, and are part of the 'global function of capital'. However, they also perform labour as wage-labourers for capitalists, and so are part of the 'global function of labour'. Carchedi sees this as a basic reason why the extent to which professional technical workers have control over their own labour processes will ultimately be limited by capitalist managers. The number and importance of professional technical workers in an increasingly technologically-based industrial structure mean that capitalists will be as concerned to simplify the tasks professional technical workers perform and replace them where possible with cheaper labour - a process of 'proletarianisation' or 'deskilling'. This view has provided the basis for a good deal of consideration of the development of software-producing occupations. Software has been seen as an opportunity to observe whether or not proletarianisation is, in fact, taking place, for a newly-emerging sphere of professional technical work.

Professional Technical Workers and the 'Deskilling Thesis'

The first two extensive considerations of the organisation and evolution of software-producing occupations in the Anglo-Saxon world were both carried out in the late 70's and in the USA. Both authors were heavily influenced by the exposition of the deskilling thesis as applied to blue collar work in Braverman's Labour and Monopoly Capital (1974). Both Kraft (1977) and Greenbaum (1979) start from Braverman's premise that the prime objective of management in a capitalist enterprise is to make sure that the amount of labour or value obtained from workers is greater than the price of their labour power, so that surplus value is extracted to at least the 'socially average' rate. They also subscribe to a Bravermanian view that the demands of competition have led managers progressively to break up and

simplify the originally integrated process of producing software.

According to Braverman, craft forms of blue collar work organisation, where workers both conceive and execute production tasks, using complex skills with discretion, are a remnant of a previous mode of production. They put limits on the levels of productivity that can be obtained. Over time, management attempts to destroy craft control of work, in three ways. The first is for management to appropriate from the shop floor knowledge as to how to operate an existing production process and to use it to specify in more detail exactly how workers should perform their jobs. The second is for management to introduce automated machinery which, apart from making workers more productive in a straightforward technical sense, also enshrines knowledge and decision-making formerly in the realm of the operator. Consequently it further removes acts of conception from the worker, making them more at the mercy of management moves to intensify work. The final element can be combined with either of the first two. It is to take advantage of the fact that work is increasingly limited to acts of mere execution, and to replace skilled workers with 'complex' labour power with less skilled workers, who sell their labour power more cheaply.

Kraft and Greenbaum start from the position that there is a conflict between software producers and managers of a precisely analogous nature, that managers have been intent on breaking up the control of conception and execution possessed by early software producers. Such early software producers are seen as exemplars of an original craft that has been mutilated by capitalist managers and capitalist forms of production technology. Greenbaum sees the emergence of programming standards in corporate data processing departments in the 1960's as a first attempt by managers to establish control at the expense of software producers' discretion. Kraft sees the process starting even earlier, when the invention of high-level languages permitted people to program computers without understanding machine code, so that the expensive labour power of

the low-level programmer could be replaced by that of the less costly applications programmer.

We have already seen in the last chapter how Kraft interprets the introduction of structured programming as a further stage, the routinisation of applications programming itself. He sees structured programming as analogous to an automated piece of machinery for producing software. It breaks down programs into modules and gives each programmer one part of the whole product to code, according to a restricted set of logical procedures which restrict discretion and judgement (Kraft 1977: 99). According to both Kraft and Greenbaum, in the USA it has given rise to possibilities of employing less educated 'technician' level workers for the detailed coding of each module, whilst higher level design work remained the province of university educated 'creative' programmers.

The idea of a basic conflict of interest between workers and managers leading to a universal management strategy of removing discretion from workers and taking control of the labour process has been eloquently rebutted by many authors who nonetheless remain basically sympathetic to Marxism (Friedman: 1977, Cutler: 1978, Elger: 1982, Wood: 1982, Littler and Salaman: 1982). There

are three main strands of qualification to the 'deskilling' thesis. The first concerns the way that labour processes and forms of management control are shaped by various forms of worker resistance as well as by management initiative. Management is not in general able to increase its direct control over work in a straightforward fashion. The second concerns the way management may find methods other than routinisation of work, close monitoring of progress, and replacement of complex labour to further an aim of establishing greater control over labour. Greater direct control may not always be the best way for management to increase levels of productivity. The third challenges the notion that managers in a capitalist firm are always primarily concerned with increasing the intensity of work and productivity of the workers they employ.

The remainder of this section examines the evidence for the applicability of each of these criticisms to an understanding of conflict between software producers and managers and the shaping of software occupations. In connection with the first two criticisms, we see how both Kraft and Greenbaum are themselves led to qualify the simple notion of 'deskilling' which they profess to support. We discuss how most neo-Marxist accounts of professional technical work organisation are forced to recognise that 'professional autonomy' can be acceptable and functional for capitalist managers, so that the problem is then to specify the conditions under which managers choose to attack it. Under our discussion of the third criticism, we return briefly to a consideration of the sorts of development in technique and work organisation occurring in software production, described in Chapter 3. This leads to conclusions that an understanding of developments in the organisation of software production requires a consideration of management priorities other than the control of the performance of workers at the individual level.

Modifications to the 'Deskilling Thesis' (1): Worker Resistance and Management Mechanisms for Producing Consent

A first weakness of the 'deskilling thesis' identified by critics is that production processes are not shaped solely by managers, just because it is they who supply the machines to be used or set down formal procedures by which work is to be carried out. Workers can influence the way work is done, either by organised resistance to change formal procedures, or by informally subverting management intentions. Kraft pays little attention to this. He is clear that in the 70's, in the USA at least, worker organisation and trade unions had been kept out of data processing, and argues that managers were actually achieving the fragmentation of tasks and direct control they were seeking in data processing. Greenbaum however is of the opinion that these

management intentions are checked by the informal patterns of work organisation that emerge 'on the shop foor' amongst software producers. She describes how, particularly for large and complex projects, software producers find it preferable to communicate informally and divide tasks up in a way that contradicts the formal procedures, division of labour and patterns of communication set down by managment. She quotes a programmer:

> "At least once a year there is some major productivity crisis and their response is to organise us into different groups and make the groups report to different management functions. We've come to take these changes as commonplace. Sometimes we don't even talk about it: we just go on doing what we always did."
> (Greenbaum 1979: 114)

It appears possible for workers to hold on to areas of discretion and autonomy in both how they do their individual tasks and how they relate them to the work of others in spite of management attempts to direct work. According to Greenbaum, such attempts are often counterproductive, because workers continue to aspire to attaining control over their work and to having more responsibility. They react to a standardisation of their work by a withdrawal of initiative, and devise ways of reducing their productivity within the framework of bureaucratic standards. For example, they use their knowledge of programming to convince managers that a task will take longer than it really will, and then use the formally-recorded estimate as sanction for withholding effort.

This recognition that workers retain areas of control even when subject to routinisation, leads both Kraft and Greenbaum to conclude that managers have not found structured techniques sufficient for establishing control. They both argue that managers have supported it with organisational devices for encouraging workers to behave and work in standardised ways, analogous to the management strategy for establishing control over blue-collar work identified as 'bureaucratic control' by

Edwards (1978).

Kraft and Greenbaum describe the creation of 'job ladders' within each segment of the workforce, where minor promotions and pay increases are awarded at regular intervals, on the basis of evaluatory procedures. Evaluation is primarily concerned with adherence to managerially-defined rules for the execution of work and with adherence to more general standards of discipline, such as timekeeping, attitude to authority and dress. One common element of such evaluations is the emphasis placed on judging the extent to which workers have a 'professional' attitude. This means that they behave responsibly in following procedures, and are prepared to work until a job is done, rather than that they show any great initiative or desire to prove themselves according to occupationally-based standards of technical excellence.

In Kraft's eyes: "An internalised ideology of professionalism suggests that ideologies are important for producing certain forms of consent to management demands, even for the most rule-bound workforce segments, consisting of 'detail workers' or 'coders'. Littler and Salaman (1982) have similarly drawn attention to the importance and also the variety of the usually

informal mechanisms by which workers' consent is reproduced even in the most routinised work situations, where management control appears to function at a very formal and direct level.

Modifications to the Deskilling Thesis (2): Alternatives to Direct Control

Greenbaum and Kraft further acknowledge that management-defined structured techniques, an increased division of labour and bureaucratic control, may not under all circumstances be the most efficient way of controlling data processing software productivity. Greenbaum holds that the 'Henry Ford' of software production has not yet arrived, and particularly for more complex development tasks, managers often recognise that workers must be

left to exercise judgement and discretion themselves (ibid: 106). Whilst it is possible to specify in great detail how maintenance and modification of existing programs should be carried out, it is simply not possible to achieve this in projects concerned with developing new programs.

In such cases, Greenbaum recognises the survival of the form of organisation typical of the "knowledge worker", where groups of workers control their own techniques, division of labour and forms of communication. Kraft too acknowledges that managers are compelled to let areas of discretion continue, but sees them as controlled through a version of "professionalism" created by managers rather than workers. He sees managers as making discretionary areas of work predictable by inculcating standards and loyalty to the firm, achieved by putting workers through common training programmes leading to certification by the company rather than an independent occupational body. Work is controlled not so much through a general 'professional' commitment to work, but through an emphasis on rewards for loyalty to the firm, for the responsible exercise of discretion. Kraft is here describing a form of management control similar to that of 'responsible autonomy' identified by Friedman for some groups of workers in the UK car industry (Friedman: 1977).

Greenbaum sees the control of workers who exercise discretion and judgement as problematic for managers, however, and as their less-preferred option. Whilst workers allowed to organise their own work may actually be more productive, she argues that their possession of knowledge as to how to conduct work is a threat to management, because it means that worker productivity is ultimately beyond the control of a bureaucratic structure. The performance of such workers is not exactly predictable, whereas that of the worker under a rigid system of bureaucratic control is:

> "The 'predictable performer' may not always be creative or the most productive, but his or her behaviour fits within the predetermined guidelines of the bureaucratic

structure." (Greenbaum 1979: 107)

Consequently, according to Greenbaum, managers are always attempting to bring groups of workers with discretion within a framework of specified work techniques and procedures. They are however sometimes prevented from advancing further in this direction, either by worker resistance, or else by the inefficiency of the result. Once tasks have been fragmented and specified by management, and allocated in an increased division of labour, 'natural' systems of co-ordination and communication between workers are lost. The management hierarchy is rarely adequate to the task of re-integration, and must be constantly redesigned to improve project reporting to senior management. The result is that the jobs of lower and middle managers are also made more standardised and routine, and less suitable for handling the sort of co-ordination that is required.

The understanding of the nature of conflict between software producers and managers and the forces shaping the evolution of software occupations emerging from the work of Kraft and Greenbaum is then as follows. There is fundamental conflict between software producers and managers over the need of the latter to control productivity and make it predictable. The originally integrated 'craft' knowledge of software producers and autonomous organisation of production is an impediment to this, and has been progressively transformed by management. They have first of all used modular and structured programming to separate off a less skilled area of applications programming, where work can be simplified and specified, and worker performance controlled by bureaucratic standards combined with internal promotion ladders. Within such areas there is evidence of worker resistance in the form of informally agreed divisions of labour and ways of working that contradict management's specifications.

At the same time, there are areas of complex work which it has not been possible to routinise. According to Kraft, managers have here contrived to establish a company-sponsored 'profession-

alism'. This exhorts software producers to undertake to exercise judgement and discretion in the service of management goals. Greenbaum also sees the persistence of areas where software producers exercise discretion as important for management, but sees them as under threat in the long term. Management is constantly trying to overcome its reliance on worker knowledge and its associated unpredictability.

Kraft and Greenbaum emphasise the way that fundamental conflict has given rise to management initiated 'vertical' segmentation of software workforces. 'High-skill' and 'low-skill' segments of the workforce have different levels of educational attainment, and different types of labour control. The existence of some aspects of control on the basis of an ideology of professionalism as well as managerially-defined rules and procedures for the 'low-skill' 'detail workers' however suggests that the differences in labour control between segments of the workforce is one of emphasis rather than absolute. 'High-skill' people are controlled rather more through encouraging and rewarding their commitment to exercise discretion responsibly, and 'low-skill' people rather more on the basis of encouraging and rewarding rule adherence.

In many ways this account of the relationship between organisational managers and groups of technical workers who continue to control their own work because of its complexity echoes the themes emerging in the literature on professionals and organisations. On the one hand, technical workers are seen as in conflict with managers who would basically prefer their work to be conducted according to bureaucratic norms; and on the other, it is acknowledged that these technical workers can be made subservient to management goals even though they control the way their work is performed, by inculcating a positive orientation towards organisational goals. What is distinctive about the arguments of Kraft and Greenbaum is the notion that the former dynamic is inherently the stronger, meaning that managers are constantly seeking to find ways of replacing the complex knowledge of professional technical workers with simplified

procedures, and to replace them in at least some areas of their work with less skilled, cheaper labour.

One problem with the sort of analysis presented by Kraft is that it does not really explain why it is that management chose to abandon a strategy of encouraging responsible professionalism, and undertake 'routinisation' and 'dequalification' of professional work. If professionals can be made subservient to capitalist goals, why is it necessary to break their control of work? Greenbaum's answer to this is that capitalist management is ultimately never content with any group of workers controlling its own work, and is looking only for the technical basis to install uniform bureaucratic control. The issue of the conditions under which professional technical workers are 'proletarianised' appears in several more generalised neo-Marxist accounts of professional technical workers. Before turning to the third type of criticism of the 'deskilling thesis', and considering the implications of management priorities other than labour control for conflict between software producers and managers and the development of software occupations, we first of all examine how these other theorists deal with the issue of understanding when and why 'professional autonomy' is attacked.

'Professional Autonomy' and 'Proletarianisation'

Although not strictly a Marxist one, one of the most influential accounts of the 'proletarianisation' of professional work is that of Jamois and Pelloile (1970). According to them, any production process is characterised by an 'Indetermination/Technicality' ratio, which expresses:

> "the part played in the production process by 'means' that can be mastered and communicated in the form of rules (T) in proportion to the 'means' that escape rules, and at a given historical moment are attributed to virtualities of producers (I)." (1970: 112)

Professional work is characterised by a high I/T ratio.

Professionals base their status within an organisation and within society on the 'indetermination' characteristic of their work: the extent to which only they can perform it on the basis of their qualifications and experience, and at the same time restrict access to such work to their fellows possessed of these same "virtualities". This is similar to Freidson's account of the nature of professional autonomy, except that Jamois and Peloille emphasise that there are also aspects of 'technicality' implicit in any professional production process. This is because professional production such as that of design engineers is based on the application of rational scientific techniques which, they argue, are used by professional practitioners to make their work more predictable, so as to extend their own control over it. Paradoxically, such efforts create the conditions for another group to take over some or all of such work, performing it according to explicit rules and procedures, as a production process in which 'technicality' predominates. Professionals are thus likely to have to defend their 'indeterminism' against incursions by less skilled workers guided by managerially defined procedures. Jamois and Peloille thus see professionals less in conflict with managers than with other groups of less skilled workers.

Apart from indicating that professionals will generally resist the removal of areas of work from them, their framework does not however contribute a great deal to an understanding of the conditions under which areas of work are taken away from them. Cooley (1987) presents a rather different argument. Rather than hypothesising a process by which some areas of 'professional' work are taken over by less skilled and educated labour, in describing developments within engineering design work, he indicates that a process of "Taylorisation" is occurring within such professional work. The principle manifestation of this process is the growth of Computer Aided Design, by which design decisions are limited to an increasingly narrow range of choice within a schema presented to the designer by the computer. Cooley postulates a general compulsion under capitalism for initiative and discretion to be removed from professional

technical workers by means of increasingly complex technical systems, usually computer-based, which regulate work. His position is very similar to Greenbaum's in that he sees a general deskilling of professionals themselves. The difference is that he emphasises increased technical as opposed to bureaucratic control. Cooley makes no explicit analysis, however, of how increased technical control affects the development of the division of labour.

In his account of developments in the division of labour and work organisation in modern accountancy, Johnson (1980) takes great pains to relate management initiatives to the evolution of the division of labour. In analysing the class position of accountants, he argues that there is a distinction between, on the one hand, "... the 'housed' accountant (working as an employee) or the partners of an independent firm who create, install and supervise control systems and regulatory procedures performs the routine day-to-day recording of stock or cash flows (Johnson 1980: 357-8).

It is the command of technique of the first group that permits the regulation and dequalification of the work of the second group. Johnson sees the profession of accountancy not as a homogenous community where all members have a similar command of technique and autonomy in the conditions under which they are employed, but as a segmented hierarchy, in which the upper rungs provide the technical means for subordinating lower ones. In confronting the problem of defining the conditions under which the upper rungs manage to maintain their autonomy and discretion as to how to organise work, Johnson argues that showing that esoteric knowledge is involved is insufficient. Rather, the degree to which the knowledge that needs to be applied by an occupation remains esoteric and under the control of individual professional workers is itself determined by:

> "... the degree to which it functions to promote and maintain capital in the course of its appropriation and expansionary dynamics ... . The ideology of

> professionalism will be an effective strategy only when its claims coincide with and draw upon the dominant ideological processes of capital." (ibid: 359)

Saks (1983) has pointed out that this argument contains a considerable element of tautology. To say that managers permit professionals to keep control over their work only when they directly serve the interests of capital is only of use if a framework is also supplied for specifying when autonomy does or does not conflict with the needs of capitalist competition. It would appear, then, that none of Kraft, Greenbaum, Jamois and Pelloile, Cooley, or Johnson provide a convincing account of the conditions under which professional autonomy is acceptable to managers, and when it comes under attack. They all tend to see it as under constant pressure of 'proletarianisation' - being replaced by less educated labour performing simplified tasks.

Modifications to the "Deskilling Thesis" (3): Management Priorities other than Labour Control

In some ways the most far-reaching criticism of the 'deskilling' thesis' is that it is simply mistaken to interpret all management initiatives as concerned with labour control (Littler and Salaman: 1982). According to this view it cannot be assumed that all changes in production technology and work organisation, even when concerned with the introduction of rules and procedures, are attempts to obtain greater direct control over the productivity of individual 'professional' workers. It is in fact possible to conceive of two different types of initiative, concerning software producers or professional technical workers more generally, which are quite compatible with the persistence of 'professional autonomy' and moreover not motivated by intentions of establishing greater control over the productvity of existing workers. The first type is concerned with increasing labour productivity by using cheaper labour, but without displacing skilled professional labour. The second type is not in fact concerned with increasing labour productivity in a

straightforward sense at all, but with improving the way work is performed in some other sense, such as its quality or predictability.

An example of the first type is where routinised forms of work and less skilled workers are introduced in the context of an expanding demand for the services of skilled professional technical workers, in order to permit 'professionals' to concentrate on the more complex tasks. It is, for example, possible that structured techniques have been introduced in situations of skilled labour shortage without experienced programmers losing their jobs or having any elements of their work removed against their will. The techniques may simply have permitted firms to employ more less skilled workers, to take on more work and to expand more rapidly. Child and Fulk suggest a similar sort of situation when they argue that the occupational control of the 'knowledge-base' of a profession is not necessarily eroded just because less skilled workers take over some of the "technicalities" previously performed. Provided that the "indeterminate" aspects of professional knowledge remain within the sphere of the professionals, and particularly if new complex knowledge is being generated, no significant loss of occupational control results (Child and Fulk 1983: 161).

An example of the second class of management initiative is suggested by the software management literature advocating the introduction of structured techniques, reviewed in the last chapter. It is conceivable that structured techniques have been introduced as a way of making sure that programs are written in a more standardised style, and become easier to maintain, rather than as a way to simplify the work of programmers and make it more susceptible to direct management control and intensification. Such an initiative could be quite compatible with the persistence of professional control over complex areas of system design, and would not even require the creation of less skilled labour market segments. It could, for example, be combined with a system where the more routinised tasks are performed by newly recruited software producers. They would then

move on to higher grades and less codified types of work. Ritti (1971) and Whalley (1986) found exactly this sort of division of labour, with an absence of labour market segmentation, amongst hardware design engineers.

To summarise, the value of neo-Marxist approaches for understanding conflict between professional technical workers and their managers lies in their account of the way managers may seek to undermine the autonomy and control of professional technical workers in order to establish control over their productivity. They also bring ideas of labour market segmentation to bear on an understanding of the development of professional technical workforces. Basically, they argue that managers seek to simplify areas of work when they can and replace qualified professional technical staff with cheaper 'technician-level' staff. These approaches also recognise that professional autonomy can however persist and be functional for capitalist firms, but they are not able to clarify the conditions under which such a situation is stable against the tendency toward 'proletarianisation'. The emphasis on antagonism between professionals and managers and on management's need to control the productivity of individual workers has also led to a neglect of other reasons why management may introduce codified techniques. Neo-Marxist accounts have tended to ignore possibilities that codified techniques are not necessarily associated with the replacement of professionals by less skilled labour, or with segmentation of labour markets. Arguably, then, such theories are not adequate for illuminating all aspects of conflict between managers and professional technical workers when codified techniques are introduced.

We now turn to a more promising strand of theorising on professional technical work, typified by the work of Larson (1977). In the light of evidence from empirical studies of the relations between professionals and bureaucracies, this approach starts by formulating a relationship of basic compatibility between the goals of professional technical workers and managers. The next section outlines this view, and examines how it can be integrated with some aspects of the neo-Marxist approaches for

developing and understanding the shaping of software occupations.

4.3 PROFESSIONAL TECHNICAL WORKERS AND BUSINESS ORGANISATIONS: "AUTONOMY WITH RESPECT TO MEANS BUT NOT GOALS."

Larson characterises professional technical workers as groups of workers who have been educated in such a way that they can be relied upon by management to "make the use of discretion predictable" (Larson 1977: 198). They have acquired complex technical skills and an orientation towards using those skills in the service of a capitalist work organisation. They are prepared to use discretion and judgement to complete tasks set by management, without close supervision, provided that they have autonomy in the way they organise their work and in the selection of technique. Their 'professional autonomy' is concerned with the realm of technique, rather than goals.

According to Larson, these new industrially-based professions are distinguished from the older 'pure' professions most notably in that they have much weaker professional institutions and occupational control of recruitment, training and standards. For the newer 'professions', autonomy means controlling the organisation of work and the selection of techniques within the firm, rather than doing so on an organised basis across the whole occupation. There is also no real issue of a clash between 'professional' and 'organisational' loyalties for the new professions. This is because they have emerged, gained usefulness, status and high levels of financial rewards entirely because business organisations began to demand their services. Corporations needed workers they could rely on to perform their most complex functions, which Larson characterises, in Perrow's (1968) terms, as those requiring most discretion, with high numbers of exceptional cases and unanalysable search procedures (Larson 1977: 197). Universities provided them with the right sort of material:

> "The decentralisation and delegation of authority inherent in the multidivisional structure compelled the monopolistic corporations to rely on professionalism as a pre-existing means for limiting the potential abuse of discretion. Not surprisingly, these giant corporations were the most hospitable to the standardised professionals - that is the university graduate who could safely be placed in discretionary, though subordinate and increasingly specialised capacities." (Larson 1977: 199)

Larson links the 'trusted' character of technical professional workers with their class position, in a Weberian sense. On the one hand, she sees them as trusted by corporations because of the way their education serves not only to give them complex technical knowledge and skills, but also a 'responsible' orientation towards the business goals. On the other hand, an individual's entry into such an occupation and their openness to such socialisation is bound up with the prospect of a level of rewards and status which stem from their value to business corporations. Professional technical workers can be trusted because of what they stand to gain from business corporations, which stems in turn from the vital role they play in the functioning of corporations. Salaman makes a similar point, in Marxist terms. He argues that only some occupational groups are 'trusted', in that their work is characterised by discretion, lack of close surveillance, and co-ordination by mutual adjustment rather than rules and procedures. He sees these as occurring in functions that are vital to corporations, and as filled by people who have, through education or selection within the firm, demonstrated a class identification with employers' goals (Salaman 1980: 127).

Larson supports this picture of mutual dependence between organisations and professional technical workers by presenting data, similar to the findings of Ritti already described, that engineers typically see promotion into management positions

within their employing organisation and professional success as one and the same thing. The creation of a 'professional' or 'technical' promotion ladder in addition to a managerial one is typically at the instigation of management rather than professional technical workers themselves, and is a way of compensating individuals if limited management opportunities are available (Larson 1977: 193).

Technical Professional Workers' Work-Related Goals

Mumford (1972) describes a set of orientations in a sample of data processing software producers employed in seven different organisations in the late 60's which illustrate the basic harmony between the needs of employers and the work goals of 'trusted' workers. She analyses the relationship between software producers and their employing organisation in terms of a series of 'contracts':

- the 'knowledge contract', by which individuals expect to be able to use and develop skills,
- the 'psychological contract', by which individuals expect to experience motivation, achievement and visibility in their work,
- the 'efficiency contract', by which people expect to have to work at a certain level of effort and receive a certain level of reward,
- the 'ethical contract', by which people expect to work in an organisation whose goals they can broadly agree with, and
- the 'task structure contract', by which people expect to experience diversity, autonomy and identity in their work.

In all seven organisations Mumford finds most aspects of these five 'contracts' remarkably well met in terms of both the needs of employers and the goals and expectations of software staff. The efficiency needs of managers are well provided for through

supplying 'responsible' workers with jobs that give them the opportunity to utilise skills with autonomy and visibility.

Ritti's findings on the nature of development engineers' work goals permit a more in-depth understanding of the relationships between the desire for organisational advancement, the desire to use skills autonomously, and a positive orientation to the business goals of the enterprise. Ritti identifies four main elements of work-related goals, which are basically similar to the expectations implied in Mumford's 'contracts':

- the desire to utilise skills,
- the desire to perform work of importance to the business goals of the organisation,
- the desire for autonomy and individual responsibility for technical decisions, and
- the desire to perform work where the individual's contribution is noticeable.

These indicate that such workers can be relied upon to bring a level of effort and commitment to their work provided that it is of a suitably demanding and responsible nature. They constitute a form of control over the quality and pace of work. Ritti sees

such orientations as stemming not simply from processes of socialisation involved in receiving an engineering education, but also from the fact that promotions are perceived as stemming from their fulfilment. Promotion also gives rise to opportunities for even greater fulfilment. Being given the opportunity to utilise skills is likely to lead to recognition and promotion, and promotion is likely to be to a position that gives wider opportunities for technical influence and further recognition.

It would appear, then, that the compatibility of this sort of professional technical orientation and organisational goals is very much bound up with the existence of career opportunities. Larson observes that organisational professions are heavily stratified according to rank within the employing organisation.

"Profession as a community of equals" is an ideological fiction for such occupations. This idea of commitment to work being reproduced through promotions up a hierarchical job ladder is to be found in many more general analyses of the ways workers are tied to organisational goals. Williamson (1975), for example, argues that an internal labour market provides a way of assessing workers over time and rewarding them for 'consumate' as opposed to 'perfunctory' co-operation. What appears distinctive about the picture of the workings of the promotion system for professional technical workers in Ritti's study is that the qualities rewarded in evaluations and sought after promotion concern the responsible use of discretion.

In Ritti's study it is not clear whether orientations to work were reproduced through the operation of a closed internal labour market, as in Williamson's model, or through an internal promotion ladder used in conjunction with recruitment from and some loss of staff to the external labour market. The latter arrangement can be seen as functioning just as well as the former for motivating people to achieve, on the assumption that their achievements may count for advancement on the external labour market as well as within the organisation. The works of Doeringer and Piore (1971) and Loveridge (1982) suggest that the existence of a true internal labour market, with recruitment only at the lowest position or 'port of entry', may occur only in cases of firm-specific skills.

The Nature of Conflict

This view of the relationship between professional technical workers and their employers moves away from the notions of fundamental conflict in much of the literature reviewed in this chapter. If the aspirations of professional technical workers have grown up in the context of industrial organisations and are well served by them, they may have little cause to desire to be organised as an occupation. Likewise, managers should have little cause to want to establish direct control over their work,

or to replace them with less skilled workers. The relationship between managers and professional technical workers described by Larson is one of a negotiated trade-off of interests, and this basic harmony in goals is born out by the empirical work of Mumford and Ritti.

It would however be unwise to argue that the types of conflict described in the literatures of the Marxist labour process and the sociology of the professions do not occur. These do, in fact, suggest two fundamental contexts in which conflict can arise between professional technical workers and their managers, in spite of a basic harmony of goals and class interests.

It is possible to imagine first of all conditions under which management might find the norms of productivity established by professional technical workers as no longer adequate, and set about routinising their tasks as a prelude to intensifying work and replacing them in some areas with less skilled workers. Child and Fulk (1983) suggest that such a move would be resisted by 'professionals', unless the removal of work is compensated for by emergence of new areas of complexity. It could also be compensated for by a straightforward increase in the volume of work, so that 'professionals' become fully occupied with a narrower range of more complex tasks. Drawing on the characterisation of 'high-skill' and 'low skill' workforce segments found in the work of Kraft and Greenbaum, the result would be a segmentation of career structures. A more highly educated segment of 'professional' technical workers would retain control of areas of work that could not be simplified, and seek promotion up a career ladder that rewards skill utilisation and the ability to meet goals set by management without close surveillance. A less-educated 'technician' segment would be subject to more bureaucratic, rule-bound forms of control and closer supervision, seeking promotion up a separate and inferior promotion ladder, with greater emphasis on rewarding rule compliance.

Second, Larson's formulation allows the possibility that conflict

between professional technical workers and managers can also take the form of a renegotiation of the details of the bargain of a responsible attitude in return for challenging work. This may occur, for example, when management wish to change the nature of the work process for reasons other than a desire to establish direct control over workers' productivity. The introduction of structured programming in order to ensure that software products are more maintainable would be an example of this. Whilst not intended to confine the technical discretion of software producers, these techniques could be experienced in these terms and lead to conflict.

Ritti's study of development engineers provides an example of exactly this sort of conflict within the terms of a basic bargain which neither management nor technical professionals want to upset in any fundamental way. Ritti describes systems development laboratories where two important elements in the routinisation of work appear to have stemmed from business concerns not directly related to the productivity of individuals. The first was the use of standard procedures or 'formatted tasks' for the design of certain elements of electromechanical control systems. The second was the retention of many engineers without

enough to do, or engaged, for instance, on formatted tasks which were well below their technical aspirations and capability.

Ritti found that such engineers experienced themselves as wanting to work not only on more challenging work, but longer hours and with greater intensity than their allotted tasks provided.

In this case, routinised tasks were clearly not introduced by management for reasons of controlling productivity and intensifying work. Rather, Ritti attributes the need for 'formatted tasks' to a need to establish uniformity in the way that the elements of a large control system were designed. The hoarding of labour is attributed to the need experienced by the corporation in question to retain a staff of engineers so that it could respond rapidly to new contracts. Management concerns with

uniformity and of being able to land new contracts were more important than ensuring that every individual was working to maximum productivity all the time. There was tension over the implications of routinisation for the expectations of professional technical workers, but no basic threat to their control of an area of work, or any intention of segmenting their career structure.

Ritti's work suggests other related tensions that can arise within a basically stable relationship between 'trusted' professional technical workers and their managers. Management may wish to take staff off development work and redeploy them before they have had the chance to 'see the project through'. In general, Ritti found that his engineers accepted the commercial nature of their work, and were not inclined to question the standards to which they were asked to work. It was however possible for staff to feel that commercial concerns were being given too much weight by management, at the expense of technical thoroughness.

CONCLUSION

This chapter has indicated the importance of taking into account the full complexity of management and worker concerns in order to understand the nature of conflict between professional technical workers and managers. We have seen in the first section that there is on the whole little evidence for positing a fundamental conflict between professional technical workers as 'professionals' and organisational authority. In the second section, we examined accounts of the 'proletarianisation' of professional technical work. We saw that it has been associated with the segementation of career structures and has generally been linked to management concerns with establishing greater control over labour productivity, although the literature is not

on the whole very successful in establishing when it is that managers turn to 'proletarianisation' as opposed to encouraging responsible professional autonomy.

In the final section, we saw that Ritti's characterisation of the work goals of engineers and their relation to promotion structures provides an account of the way that 'professional autonomy' serves the needs of business organisations. We argued that conflict between professional technical workers can occur in two different contexts: when managers attempt 'proletarianisation' and the introduction of a 'technician' career structure, and when management concerns other than labour productivity lead them to organise work in a way that does not meet the work goals of professional technical workers, for example by introducing routinised techniques.

The contribution of this chapter to the development of a theoretical framework adequate for understanding the nature of conflict between software producers and managers and the forces shaping the evolution of software occupations is as follows. In terms of conflict, Ritti's account of work goals provides a basis for understanding the orientations of software producers and their reactions to two different types of management initiative where tensions and conflict may result. In terms of the way occupations are shaped, the work of Larson articulates how the emergence and persistence of 'professional autonomy' without formal professional institutions is linked to the way that professional technical workers internalise the goals of business corporations. The work of Kraft and Greenbaum suggests how, in the USA at least, management concerns with establishing direct control over labour productivity have led to a segmentation of career structures and labour markets in software.

None of the literature covered so far provides concepts, however, for a comprehensive understanding of how management concerns give rise to different types of initiative in the field of work organisation. It is the objective of the next chapter to review the literature on the relationship of systems of management

control and co-ordination to business environment in order to provide such concepts. These concepts must take account of the range of management concerns implied by the different business situations for software-producing organisations described in Chapter 3.

CHAPTER 5

MANAGEMENT CONTROL CONCERNS AND MECHANISMS IN PROFESSIONAL TECHNICAL WORK

INTRODUCTION

The purpose of this chapter is to explore ways of conceptualising what management are concerned to control in software-producing organisations, beyond the productivity and quality of output of individual direct producers, and the mechanisms they use for doing so.

The position adopted here on the nature of management control in a capitalist enterprise is in essence an elaboration of that advocated by Salaman (1979). According to this view (Salaman 1979: 108), managers are concerned with two interrelated but conceptually distinct aspects of what happens in a work organisation.

On the one hand they are concerned with ensuring that the division of labour within the organisation is **coordinated** sufficiently to meet performance goals. The need to coordinate encompasses issues of effectively matching and harmonising, or 'integrating' the division of labour within the organisation. There are issues of coordination at many different levels within an organisation. There is a need to integrate the work of sub-units or departments engaged on different aspects of the overall organisational task, and to recombine specialised jobs produced by the division of labour between workers within each sub-unit.

On the other hand, managers are concerned with the **control of individual performance**, "...to ensure that members perform their differentiated tasks, and that they operate at the organisationally adequate level of competence or quality" (Salaman 1979: 109).

This view makes the weakness of much neo-Marxist writing in the 'deskilling' school clearer, in that it has focussed predominantly on the second aspect of management control. In the writing of Braverman (1974), for example, the increasingly fragmented division of labour and oppressive forms of direct

management control are seen as stemming from management's need to control individual productivity. With this as the primary dynamic, Braverman then merely refers to management's need to coordinate the fragmented division of labour they have created. This school does not on the whole provide concepts for understanding the nature of coordination concerns, the organisational mechanisms used to deal with them, how these mechanisms are related to those used to control productivity and quality, and what effect they have on the nature of management-direct producer relations and evolution of occupations. The neo-Marxist 'labour process' literature and the neo-Weberian accounts of professional technical workers, as typified by the work of Larson (1977), together give us a rich typology of mechanisms by which managers may seek to establish control over levels of productivity and competence or quality. They have much less to say about management needs for coordination and how these may give rise to conflict.

This chapter is concerned with the part of the organisational behaviour literature which deals with the determinants of organisational structure and modes of functioning, to see what concepts it offers for understanding issues of coordination within software-producing organisations. In particular, the focus is on how various types of coordination concerns and mechanisms can lead to conflict with direct producers. They may clash with preferences and expectations stemming from the 'professional' socialisation of such workers, which is at the same time in effect a mechanism for the control of productivity and quality.

The remainder of the chapter has four main sections. The first discusses in general terms the nature of conceptions of management control in 'contingency' theories of organisation, briefly examines the shortcomings of this sort of approach for analysing software-producing organisations, and summarises some of the more general criticisms that have been made of contingency theory, particularly in respect of the way it neglects processes of choice, negotiation and the exercise of power.

The second section summarises a body of literature which has sought to combine a recognition of choice and negotiation with an awareness of the constraints posed by contingencies. It derives a framework for understanding how mechanisms of control and coordination are shaped at two different levels within an enterprise - the selection of strategic objectives and choice and shaping of mechanisms of control and coordination within this context.

The third section looks at the literature on strategic typologies to derive concepts useful for analysing the first set of processes within software-producing organisations.

The fourth section explores the sorts of management control and coordination mechanisms that may be used in various strategic contexts, the sorts of tensions that may arise between managers and direct producers, and the forces shaping the evolution of the division of labour. This final section paves the way for Chapter 6, which sets out a framework for understanding the circumstances under which various types of control and coordination mechanisms are likely to be used, the nature of conflicts associated with them, and their impact on the evolution of occupations.

5.1 CONCEPTIONS OF MANAGEMENT CONTROL IN CONTINGENCY THEORY

In the 60's a number of theorists used studies of organisations operating under contrasted conditions to argue that various 'contingencies' had definite implications for the types of organisational structure and systems of control and coordination that should be adopted in order to ensure effective performance and organisational survival. Implicitly, they were articulating a notion that the things organisational management needed to control or coordinate and the mechanisms they used to do so depended on the task and environmental context of the

organisation.

A common assumption underlying much work within this school was that organisational structure, systems of control and coordination had to be adapted to the various types of uncertainty in decision-making faced by an organisation. Different theorists focussed on the effects of different contingencies and the types of uncertainty they produced. Two main classes of uncertainty-producing contingency were considered, the 'environment' within which the organisation functioned and the 'technology' or production system it utilised. Whilst there has often been a good deal of overlap between what different theorists have referred to under these two headings, it is still worthwhile using them to illustrate the sorts of insights which contingency theorists can offer into the nature of management control and coordination concerns and mechanisms.

Management Control and 'Environmental' Uncertainty

The classic accounts of the relationship between environment and organisational structure and systems of coordination and control were those of Burns and Stalker (1961) in the UK, and Lawrence and Lorsch (1969) in the USA. Burns and Stalker examined management practices in manufacturing firms operating in a variety of industries. They argued that the crucial determinant of the type of management system required for successful performance was the rate of change in the environment, in terms of the rate of technological change in products and the associated frequency with which the nature of product markets changed. They identified two contrasted **management systems**, the mechanistic and the organic, which they saw as suited to **stable** and **unstable** environments, respectively. They recognised that most companies used both systems to some extent, but held that notably stable or unstable environments required one or other to predominate. They characterised the two systems and their appropriateness to their environments as follows.

Under stable conditions, it is possible to break down direct production activities into well specified, specialised and independent sub-tasks, allocated to individuals. The responsibilities and methods of each individual can be specified by management. Such an organisation is 'mechanistic' because the actions of its members are programmed according to plans and procedures drawn up by a management hierarchy. Coordination of sub-tasks and control of individual performance are both achieved by a hierarchy of managers collecting information on performance and making sure that procedures are adhered to. Problems within the production process that cannot be dealt with according to procedures are referred up the hierarchy for resolution, on the assumption that only more senior managers have the knowledge of the firm and its situation needed to resolve them. Management control is thus mainly concerned with ensuring performance according to procedures, on the basis that procedures have been devised to meet a set of stable and predictable performance requirements for the organisation as a whole.

In contrast, unstable conditions mean that actions to be taken within an organisation cannot be specified precisely in advance, nor can they be allocated rigidly into well-defined and specialised jobs. The 'organic' type of organisation is based on some understanding of the organisation's task and environment lying with those involved in direct production, as well as with senior management. Individuals have basic responsibilities allocated to them, but changing conditions and unforeseen problems mean that methods have to be constantly changed. Lateral communication is required for individuals to work out exactly what they should be doing and what their colleagues should be doing and to decide on the approach to be taken.

In such an organisation, management control no longer takes the form of a management hierarchy setting and monitoring performance according to procedures and making occasional adjustments when exceptional circumstances emerge. The mechanisms by which control and coordination are achieved become much more diffused

through the organisation. Implicitly, as well as monitoring overall performance, management is concerned with making sure that organisational members have certain levels of expertise and commitment to organisational goals.

In many ways, this understanding of how managers control the work of individuals in changing environments is similar to that of the neo-Marxists discussed in the previous chapter, who argue that control of complex tasks that cannot be routinised must be achieved by giving discretion to workers who have been socialised to be responsible. What Burns' and Stalker's framework brings out, however, is an understanding of how the issue of coordination becomes more complex in uncertain environments. Workers are seen not simply as responsible for maintaining levels of effort and quality, but also for coordinating their work.

Lawrence and Lorsch (1969) used data from a larger sample of research sites to elaborate further a model of how the nature of an organisation's environment raises issues of coordination which put requirements on its structure and management systems. This model involves a more detailed conceptualisation of both decision-making uncertainty associated with an organisation's environment and the forms of organisational structure and management systems needed to coordinate work.

With respect to environments, they go beyond Burns' and Stalker's single dimension of environmental stability or change, recognising that uncertainty in the environment of an organisation can also be multi-faceted or diverse in nature. According to Lawrence and Lorsch, an organisation designing, manufacturing and selling technologically sophisticated products faces decision-making uncertainty or unpredictability as to what scientific knowledge is relevant to new products (its 'scientific environmental sector'), as to how product markets will behave (its 'market environmental sector') and as to how the current production system for manufacturing existing products would behave (its 'techno-economic environmental sector'). Moreover the levels of uncertainty in each of these three 'environmental

sectors' can be different. They measure 'decision-making uncertainty' in each environmental sector in terms of the perceptions of organisational members of the clarity of information received from the environment, the time-span of definitive feedback and the level of uncertainty regarding cause-effect relationships.

Lawrence and Lorsch see a major issue for organisational managers in their need to devise structures and systems to coordinate or 'integrate' parts of the organisation dealing with different types and levels of uncertainty in the three different 'environmental aspects'. For organisations dealing with low levels of uncertainty in each area, departments can be structured and controlled along 'mechanistic' lines. Such rule-bound departments were relatively simple to coordinate, by a further set of rules governing their interactions. For organisations in the specialist plastics industry, however, they observe much greater levels of uncertainty in the 'scientific' environmental sector than in the 'market' or 'techno-economic' sectors. Successful firms in this industry are characterised by greater 'differentiation' in terms of the types of management system used in research and development, marketing and manufacturing departments.

Their data on the types of structure leading to successful performance also indicate that it is important that firms operating in a 'diverse and dynamic environment' adopt structures and systems specifically designed to integrate their well-differentiated organisational sub-units. Such structures and systems cannot however be based on such things as the rules and procedures used to coordinate activities in organisations operating in more certain environments, because the non-standardised ways of operating of the departments concerned produce too many exceptional cases. Lawrence and Lorsch identified a range of "high grade" mechanisms able to integrate unpredictable sub-tasks, such as cross-departmental integrator roles and joint teams.

The studies of Burns and Stalker and of Lawrence and Lorsch thus emphasise management concerns that stem from uncertainty in various aspects of an organisation's environment, and highlight issues apart from the need to control the productivity and quality of output of individual workers. They lead first to an understanding that organisational sub-units may need to control and coordinate workers in diverse ways, so that each unit or department is effective in dealing with a particular aspect of the organisation's environment, and its type of uncertainty. Secondly, they indicate management's need to coordinate such sub-units with special integrative mechanisms if sub-units have to deal with uncertain or 'dynamic' environmental aspects.

Management Control and 'Technological' Uncertainty

Perrow (1968), Thompson (1967) and Woodward (1965, 1970) are the classic exponents of the effect of 'technology' as a contingency upon organisational structure and functioning. Their work implies that there is a relationship between the type of uncertainty in the production process or 'technology' employed by a work organisation, and the forms of control and coordination required within it.

Thompson sees mechanisms of coordination as shaped by the 'technology' of the 'operating core' of an organisation, in the sense of **the type of interdependency** existing between the tasks of its workers. He distinguishes three basic types of interdependence, which exhibit increasing levels of uncertainty from the point of view of coordination. They are seen as requiring increasingly complex and costly forms of coordination.

Easiest to coordinate are tasks which have 'pooled' dependence, where an organisation mediates between or provides a standard service to a large number of independent clients. The only form of coordination needed is to ensure that like customers receive the same service. This can be done simply by categorising customers and specifying rules and procedures for workers to

follow for each class. The second form of interdependence between tasks is that found in a standardised manufacturing operation, where one operation succeeds another, so that there is 'sequential interdependence'. Such 'long-linked' technologies require more costly coordination mechanisms than mere rules to guide the action of each worker individually. They require planning of the production process to make sure that materials flow from one position to the next. Finally, Thompson identifies organisations with 'intensive technologies,' where tasks are concerned with complex transformations of a single object, as in building an engineering prototype or conducting scientific research. Here the results of actions are not predictable, and so coordination of work involves feedback from the object, and between one worker and another. The relationship between tasks is one of 'reciprocal interdependence'. Such tasks require 'mutual adjustment' as an integrative mechanism.

Perrow (1968) provides another framework for understanding how organisational managers are concerned with issues other than controlling individual productivity. He describes types of management system appropriate for coordinating tasks in production processes characterised by different types and levels of uncertainty. He starts from a very different conceptualisation of the nature of uncertainty in 'technology'. He focusses on the the extent of 'non-routineness' in actually transforming materials or providing services, which he refers to as the 'materials technology' of an organisation. He identifies two dimensions: the frequency with which exceptional cases are encountered, or the extent to which tasks are variable; and the extent to which 'search procedures are analysable', or how complex the activity of finding a response to exceptions is. These two dimensions allow Perrow to divide a continuum of types of 'operation technology' into four cells.

Perrow describes the systems of control and coordination appropriate to the four extreme cases of 'operations technology'. For example, organisations dealing with frequent exceptions and unanalysable search procedures require that workers in the

operating unit coordinate by mutual adjustment. The technical staff who plan and control work have also to coordinate by mutual adjustment, and there is a high degree of interdependence between the two functions.

Again, although the focus is on forms of coordination, there are assumptions in each of the four types of management system described by Perrow as to the way that workers' productivity and quality of workers' output should be controlled. In the case just described, there is an assumption that workers need to have a responsible orientation to organisational goals so that they can be trusted with discretion in their own tasks and in relating to the work of others.

The work of Reeves and Woodward (1970) is explicitly concerned with the relationship between uncertainty in manufacturing production processes and forms of management control, although based on a very different conceptualisation of production technology than those of both Perrow and Thompson.

In an earlier study (1965), Woodward detected differences between the types of organisational structure and management system found in firms with the three different types of production system. She identified three basic categories of production process, of increasing technical predictability and complexity: unit and small batch production; automated large batch and mass production of discrete products; continuous-flow production of bulk material. She found that the most successful firms in each category tended to have near the average value for a number of measures of organisational structure for their category. She concluded that the type of production system was a significant constraint on the type of organisational structure and system of management required for control and coordination.

For firms engaged in unit and small batch production, which is what we have seen to be characteristic of software production, Woodward described a changeable production process, little regulated by automated machinery, although sophisticated but

flexible machinery might be used (Woodward 1965: 25). She identified a form of organisation in many ways similar to the 'organic' form described by Burns and Stalker. Coordination was achieved at one level by production managers who allocated people to jobs, but issues within a job were resolved through mutual adjustment between workers, or else were referred to first line supervision, who were technically skilled and worked closely with the operators, typically in small groups.

Woodward presents a picture of a form of organisation based on skills applied with flexibility and discretion, necessitated by the need to coordinate an unpredictable organisational task. She also sees it as addressing the control of individual performance, although she observes that unit and small batch production firms rarely seem to experience a great deal of pressure from their clients, and so the control of group or individual performance to time and cost is rarely an explicit concern. She concludes that an important element of the sort of management system she describes is that worker performance is assured through 'self control'. She sees this as not simply the product of individual socialisation, but the result of commitment to group performance stemming from the satisfying 'network of relationships demanded by production' (Woodward 1965: 159).

Reeves' and Woodward's later (1970) consideration of management control starts from an investigation as to why there appears to be much greater variation in organisational structure and type of control system in large-batch and mass production firms. It leads to a deeper understanding of the nature of management control systems, as being concerned both with coordination and control of individual performance, and of their relationship to different types of production process. The study is based on Eilon's notion that a system of management control contains four basic elements: setting objectives, planning activities to meet them, execution, and control of activities through feedback and new planning (Eilon 1963). Reeves and Woodward distinguish between what they term 'performance' and 'adjustment' mechanisms. The former are mechanisms for motivating performance of workers

and directing their activities i.e. they serve to make outcomes predictable and according to plan. The latter serve to gather data on what has actually been done, and to make adjustments in plans accordingly. Conceptually at least, Reeves and Woodward are aware that management control is concerned with control both of worker performance and coordination. They are also of the opinion that in many cases the same management control mechanisms serve both purposes.

In terms of how management control systems change in nature across the three types of production process, they identify two important dimensions of variance. The first is the extent to which planning and control are carried out through personal, as opposed to impersonal or mechanical means. 'Personal' refers to situations where planning and control are achieved through direct contact between workers and supervisors or managers. They observe that in unit production and small batch firms, controls are largely personal.

The second dimension is the extent to which planning and control are unified, as opposed to fragmented, in two senses. Planning and control can be fragmented across departments in the sense that different mechanisms are for different departments. They can also be fragmented within one department, in that standards can be set by different functional specialists for cost, time and quality of production, and seperate mechanisms can be used to monitor these different aspects of performance. They observe, however, that unit production and small batch firms tend to have unified control systems.

Shortcomings of Contingency Theory: The Importance of Managerial Choice, Power and Conflict

In the previous chapter we characterised professional technical work organisation in terms of workers with complex skills and a commitment to organisational goals, who exercise discretion and judgement, work to certain performance standards and also

coordinate their work with each other. Contingency theory would see organisations or sub-units with this form of structuring as having a system for control and co-ordination of work well adapted to cope with the uncertainties inherent in dynamic environments, work processes that have frequent exceptions and analysable search processes, and work instruments or machines which do not regulate the conduct of work.

Whilst contigency theory provides concepts for understanding management concerns and mechanisms for coordination as well as individual performance control, in relation to a wide range of organisational circumstances, it has shortcomings from the point of view of understanding how control systems are shaped over time, and the nature of organisational conflict over their form. A brief consideration of the sort of management control concerns and conflicts described in Chapter 4 can be used to illustrate points made by organisational theorists who have argued out such a case.

Ritti's study of system engineers suggests that tensions can arise because of the use of elements of task standardisation to coordinate the work of professional technical work. A coordinative mechanism of this type appears to clash with the predominant mechanism for controlling individual performance. The commitment of engineers to their work requires interesting work and promise of advancement to be mobilised. This raises the possibility that the various uncertainties encountered by an organisation may make conflicting demands on its management control system. It is also possible that managers at different levels within the organisation or in different departments may feel more concerned with addressing one particular set of uncertainties. In Ritti's case it is conceivable for example that first line supervisors were concerned above all with ensuring individual performance, and wanted to maximise the challenge and autonomy given to their team. They might then resent the imposition of standards imposed by the next line of technical management, who were concerned above all with integration.

The issue raised here is basically that the existence of a unitary set of operating objectives cannot be taken for granted even amongst the managers of an organisation. Dawson and Wedderburn (1981) summarise many different strands of theorising about organisations, including the work of Marris (1964), Weber (1948), Cyert and March (1963), and Thompson (1967), which have all emphasised the limitations of a simple concept of managerial rationality. It is possible for functional departments or different levels within an organisation to pursue operating priorities that differ in their emphases. Further, even with shared goals, Cyert and March emphasise that it is often impossible to evaluate in advance the consequences of a course of action for organisational performance, so that rationality is often 'bounded'. No course of action is automatic and several different ones appear possible.

What this means in practice is that different levels of management or functional departments may see the mastery of different uncertainties as central issues in achieving their differing performance goals. Even if there is agreement as to what contingencies need to be addressed, there may be different views as to the types of structures and control sytems appropriate for dealing with them. In addition to the 'bounded rationality' argument, Child (1972) has pointed out that, even if it is known exactly what the consequences of different organisational structures and control systems are for an agreed version of desired organisational performance, the level of performance given by the most effective structure may be higher than that which is considered necessary. In this case, there is scope for powerful factions to try to bring about organisational forms which they find preferable for reasons other than their contribution to organisational performance. Alternatively, they feel able to allow less powerful members of the organisation to pursue their favoured arrangements.

There is then a strong argument that organisational structures and systems of management control are determined by political

processes where different perspectives come to some sort of accommodation, rather than by unequivocal perceptions of what is necessary, or by dictates of efficiency.

There are also problems with contingency theory for understanding why managers may want to introduce new work techniques, such as structured programming, and the forms of conflict that may arise. Although Woodward, for example, was interested in the implications for organisational change of a move from a less-regulating to a more-regulating production technology, on the whole, contingency theorists start from an assumption that the 'technical system' or the set of instruments for conducting work is given. It is necessary to conceptualise management priorities at some higher level in order to understand how choices of technical system are made in the light of the needs of both coordination and control of individual performance. Many theorists, such as Thompson (1967), Perrow (1970), Aitken and Hage (1971) and Child (1972) have argued in a similar fashion that the market and environment in which an organisation moves can also be shaped to a considerable extent from within the organisation.

The next section reviews the 'strategic choice' strand of organisation theory, which retains the insights of contingency theory as to how management control systems must address issues of coordination and control subject to a variety of forms of uncertainty, in order to ensure effective performance. It also incorporates an understanding that management control concerns may be different at different levels within an organisation, that there may be conflict between managers over what control mechanisms are appropriate, and that the contingencies of environment and technology may themselves be subject to managerial choice in the light of higher order priorities.

5.2 THE SHAPING OF MANAGEMENT CONTROL SYSTEMS

Strategic Choice and Organisational Adaptation

Child's (1972) critique of contingency theory gives a central role to the concept of strategy for understanding how work organisations are structured and controlled. Child takes a broad view of the nature of strategy, beginning from Chandler's definition as "the determination of the basic long-term goals and objectives of an enterprise, and the adoption of courses of action and the allocation of resources necessary for carrying out these goals" (Chandler 1962: 13, Child 1972: 14). He sees the structure, systems of management control and general functioning of an organisation as shaped in the first instance by the basic strategy or goals adopted by powerful decision makers making up the 'dominant coalition'. These are generally, but not invariably, those who have formal positions of authority. However, Child does not replace the determinacy of the 'contingencies' with the idea that strategy at this general level determines organisational arrangements. Goals themselves are seen as the object of choice to a considerable extent, and once they are fixed there is still a realm of choice as to what sorts of organisational arrangements can be used to meet them.

Child proposes the following model for the processes by which organisations are shaped. Strategic decisions stem from the dominant coalition's evaluations of the organisation's position and performance in relation to expectations placed upon it by resource providers such as shareholders or a parent company. They may also evaluate the organisation in terms of what it is like to work in, the "congeniality of its present internal configuration." From this evaluation an overall strategy or set of basic goals emerge, which provides ground rules for decisions as to actions to be taken in two fundamental areas, which Child terms 'environmental strategy' and 'organisational strategy'. The first refers to choices or adjustments made in the markets or areas of activity of the organisation, which shape the

environment it has to work in. The second covers decisions as to the internal arrangements needed to secure the level of performance required by the goals and environment chosen. Such decisions concern the type of production technology, organisational structure, type of manpower and training, and the size of the organisation.

According to this model, organisational structure and systems of management control are no longer seen as determined in any strict sense by environment and technology, or, as some contingency theorists have hypothesised, by organisational size (Pugh et al 1968). What is necessary for the achievement of the chosen goals is that there is an effective 'fit' between these contingencies and the way the organisation is structured and controlled. Child includes the type of 'human resources', and, implicitly, the way workers are controlled as individuals and coordinated, in his picture of organisational arrangements that need to be 'fitted'.

The notion of contingencies determining effective organisational arrangements is abandoned because, first of all, not only structure and systems of control but also environment, technology and size are now seen as objects of 'strategic choice'. Members of the dominant coalition may choose to keep an organisation small rather than change an organisational structure which gives managers a great deal of personal control over activities, or to move into a market which will allow a more routinising automated technology, a less skilled workforce and a more rule-bound mechanistic structure. Second, even accepting environment, production technology and size as given, there may be considerable scope for value-based preferences in structure and forms of managment control to be exercised.

Finally, it is possible to conceive of different contingencies pulling organisation design decisions in different directions. According to the work of Pugh et al (1968), large organisations require more formalised structures, whilst we have seen the arguments of Perrow and evidence from Woodward that a production process with a high degree of 'non-routineness' and a technical

system which is not highly regulating require an organic structure. In such circumstances, Child's framework allows organisational design decisions to be understood as a weighing up of the implications of adopting one type of structure or the other, or some combination of the two.

To summarise, this model implies that the mechanisms used for management control in an organisation have to be understood not as functionally determined responses to external contingencies, but as the product of a process of decision-making, mainly involving the dominant coalition. There are two parts to this process. First of all there are fundamental strategic decisions by which the overall goals of an enterprise are set. These set the context for the second part. Decisions are made on the one hand as to what markets the organisation should operate in, and how it can shape its business environment within its chosen market. On the other hand, decisions are made as to what production technology it should use, what size it should be and what sort of workers it should employ, and how it should be structured.

Different Levels in the Shaping of Management Control Systems

Child stresses that strategic choices are not made solely by senior managers or even by members of the dominant coalition. In general, middle managers and other organisational members also have an input, because they are involved in the implementation of strategic choice. The roles of organisational members other than senior managers in making choices concerned with shaping the system of internal control and with the mechanisms used to shape and control the external environment, are brought out more explicitly in Mintzberg's conceptualisation (1979: 25-30) of the elements of an organisation.

Mintzberg sees the management structure of a work organisation essentially in terms of a first line of supervision responsible for the control of the 'operating core', a 'middle line' and a

'strategic apex'.

According to Mintzberg, managers in the strategic apex are responsible for monitoring overall organisational performance and developments in the organisation's environment. This includes surveying the state of product markets, the behaviour of competitors, suppliers and distributors. On this basis they clarify strategy in the sense of setting goals.

In the light of these they on the one hand make decisions as to how the environment itself can be shaped. This may take the form of entry into new markets, for example by extending geographical scope or by introducing new products, or of intervening in the behaviour of other agencies, for example by setting up cooperative agreements with suppliers or competitors. Decisions as to how to deal with the environment are based on their perceptions of it, which are shaped by their 'perceptual set' as well as by the way they have collectively experienced markets, suppliers, customers and distributors and governmental regulatory bodies in their interaction with them. Weick calls this 'environment', as perceived by organisational members in the course of their interactions with it, the 'enacted' environment (Weick 1979).

On the other hand, they make decisions as to how to tailor the basic internal parameters of the organisation - its products or services, size, workforce and technical system - in order to meet the demands of the environment. They define resources, tasks and performance criteria for middle managers responsible for various aspects of the overall organisational task stemming from the organisation's strategy.

In practice, their decisions on internal structure and tasks constitute definitions of overall tasks and performance criteria for the organisational sub-units headed by the next line of managers reporting to them. Mintzberg sees these definitions of tasks for sub-units as a set of constraints within which middle managers can, in turn, make choices as to details of the

technical system to be used, the structure of their departments and the local system of management control. In so doing they have to take account of the need to achieve a fit between the technical system, structure, and systems of control used and the types of environmental uncertainty impacting their particular unit.

According to Mintzberg, whilst middle managers shape management systems at a lower level, their level of discretion and the type of decisions they have to make are, in many ways, just like those of their senior colleagues in the strategic apex:

> "In general, the middle manager performs all the managerial roles of the chief executive, but in the context of managing his own unit. He must serve as a figurehead for his unit and lead its members; develop a network of liaison contacts; monitor the environment and his unit's activities and transmit some of the information he receives into his own unit, up the hierarchy, and outside the chain of command; allocate resources within his unit; negotiate with outsiders; initiate strategic change; and handle exceptions and conflicts." (Mintzberg 1979: 29)

The important point for our present purposes is that there appear to be two basic levels of management concerns and decision-making that shape the structure, functioning and systems of control and coordination in an organisation. First there is the 'strategic' level which deals with overall goals, and the management or shaping of external environments. This is largely the responsibility of the senior managers in the 'strategic apex'. Then there is shaping of internal arrangements in the context of the definitions of the tasks for organisation sub-units that stem from these strategic concerns. These definitions are referred to by Galbraith (1977) as the 'task environment' of the organisation, that which has to be addressed for the organisation to achieve its strategic goals in its chosen business environment. Size, technology, structure, systems of control and

the use of human resources are in general shaped by management decisions at a number of levels in order to meet the requirements of the task environment. The strategic apex may make broad decisions as to what forms of technology are to be invested in, how the performance targets are set for sub-units and in what form they must report their performance, but the details of these aspects of organisation and their interaction are shaped by lower-level organisation members, most strongly by middle managers.

The remaining two sections of this chapter examine these two different levels at which systems of control and coordination are shaped.

5.3 THE IMPLICATIONS OF BUSINESS STRATEGIES FOR TASK ENVIRONMENTS

The way that strategic goals are set in business organisations and how they affect organisational structure is the subject of very large bodies of theorising and research, a thorough treatment of which is beyond the present scope of inquiry. Here, we consider the literature highly selectively, in order to draw out the ways that decisions made at the strategic level in a software-producing organisation define task environments.

A seminal work in the field of the relationship between strategy and structure is that of Channon (1962), although the focus is limited, being concerned with approaches adopted in multi-divisional firms. Two more recent formulations of the link between business environments, strategy and structure which have more to say about the structure and control of 'single' business units are those of Miles and Snow (1978) and Porter (1980, 1985).

Defenders, Prospectors, Analysers and Reactors

Miles and Snow (Miles and Snow 1978, Miles et al 1978) conceptualise the process of strategy formulation in an organisation in terms of the solution of the 'entrepreneurial problem' for senior managers. They must decide what the domain of their organisation is to be, in terms of the goods or services to be supplied, and the target market or market segments. The way these strategic decisions shape organisational structure and process is seen in terms of the 'engineering problem' and 'administrative problem' that must be solved in order to bring into existence the chosen solution to the 'entrepreneurial problem'.

The 'engineering problem' is concerned with finding a production technology able to deliver the chosen products and services to the chosen markets, and with devising systems of information flow for the communication needed to operate and control it. The 'administrative problem' is concerned on one hand with rationalising and reducing uncertainty in the solutions to the entrepreneurial and engineering problem already put into practice. On the other hand, it is concerned with "formulating and implementing those processes which will enable the organisation to continue to evolve (innovation)" (Miles et al 1978: 550).

Miles and Snow are however able to go a good deal further than Child in conceptualising specific types of overall strategy and the way they constrain organisational structure and process, including systems of management control. They offer a four fold 'strategic typology', characterised by different types of solution to the 'entrepreneurial problem', which in turn call for different types of solution to the 'engineering' and 'administrative' problems. The four strategic types are labelled defenders, prospectors, analyzers and reactors.

The strategy of a **Defender** is concerned with establishing a stable market or market niche in which it can be assured of a predictable demand for an established product or service or set of them. Senior management deliberately select a stable product

market environment and ignore what is happening elsewhere. The strategy for maintaining control in the selected market is to maintain competitive prices and product quality, together with some limited product development.

The basic emphasis is however on cost reduction, and overall this strategy leads to a stable task environment. The organisation is basically concerned with cutting costs on well-defined and established activities. The organisational consequences of the 'defender' strategy are 'mechanistic' administrative procedures combined with machine regulation to achieve both control of individual performance and coordination.

At first sight, this strategic type does not seem of great relevance to the software industry, for two reasons. First, software producing organisations are basically engaged in the development of new systems rather than in the production of existing designs. The task of production is by-and-large trivial - consisting merely of making a copy of the prototype code, usually stored in some form of digital memory. Secondly, Chapter 3 indicated that product markets in the software industry are highly unstable, with a high rate of change in products. It would thus be difficult for an organisation successfully to specialise in one product or set of products without considering fundamental changes. However, an organisation could conceivably mark out a particular application area, and market a fairly standard packaged system, with minor enhancements to suite the needs of particular customers, carving out a stable market. It could then seek to defend its position by virtue of its efficiency in terms of cost and quality in delivering a fairly standard and predictable system.

The **Prospector's** approach to the 'entrepreneurial problem' is in most respects the opposite of the defender's. Senior management aim above all to find and exploit new product opportunities and new markets. According to Miles and Snow, management may even put the company's reputation as an innovator above its record for

actual profitability. Consequently they choose, 'enact' and perceive a dynamic and varied task environment. The fact that prospectors are often concerned with introducing new products and opening up new markets mean that their managers are also likely to perceive a good deal of environmental uncertainty. They are

likely to employ considerable resources in scanning and assessing the environment for opportunties.

The rapidly-changing nature of computer hardware and software product technology, and of markets for software products and services imply that strategies of this type should be common in software-producing organisations.

According to Miles and Snow, the **Analyser** combines some of the characteristics of both defender and prospector, lying at a point on a continuum with the first two strategies at the extremes. In terms of its 'entrepreneurial problem', it is concerned with locating and exploiting new product or service opportunities whilst maintaining revenue from established products or services and markets. It grows through a combination of market penetration on existing lines and the addition of innovative ones. Its engineering problem is therefore to develop an equilibrium between demands for technological flexibility and for stability. It must have technical systems that are efficient in producing standard lines, but flexible enough to adapt to new ones. The 'administrative problem' is that of differentiating and integrating the organisation's flexible, innovative and dynamic areas of operation on one hand, and its stable, routinized parts on the other. In effect, Miles et al (1978: 555) see analyser organisations as dealing with two different task environments simultaneously: one stable, one dynamic.

It is highly conceivable that a software producing firm could adopt an analyser-type stance, marketing standard packages with minimal customisation, as well as developing new products - bespoke and packaged. In such cases, according to this model, the different requirements put on different parts of the

organisation would amount to a differentiation in the type of task environment perceived in each.

Miles and Snow's final category, that of the **Reactor** is that of organisations which do not respond successfully to the environment they have selected. This may occur because there is no clearly articulated strategy, because organisational structure and process are not in harmony with strategy, or because the environment to which the organisation once adapted has now changed: as when demand for an established product falls off because of the sudden emergence of a substitute and strategy is not adapted. In such cases, it seems reasonable to assume that management control systems at the level of direct production will be shaped by either an inconsistent set of strategic priorities, or else by a consistent but inappropriate set. There are thus no general task environment characteristics associated with this strategic type.

Cost-Leadership, Differentiation and Focus

Porter provides a strategic typology in many ways similar to that of Miles and Snow, although he bases it on a notion of competitive strategy, the approach taken by an organisation in seeking to establish competitive advantage, which "grows fundamentally out of the value a firm can create for buyers that exceeds the firm's costs of creating it." (Porter 1985: 3) According to Porter, there are two alternatives for achieving competitive advantage, cost-leadership and differentiation, and three 'generic strategies' for obtaining it: cost-leadership, differentiation and focus.

Cost-leaders seek to become the low cost producers in their markets, and are essentially similar to Miles' and Snow's defenders. Porter points out that for such a strategy to be successful, lower costs must be achieved without incurring any significant differentiation of the product from that of higher priced competitors. The strategy of **differentiation** seeks to

achieve the opposite, to get its products or services perceived as unique along some dimension that it is valued by customers, so that they are prepared to pay more for them. For such a strategy to work, firms must ensure that their costs do not rise proportionately. Miller argues that there are in fact two sub-types amongst differentiators. There are those that seek differentiation by achieving product innovation, by creating the "most up-to-date and attractive products by leading competitors in quality, efficiency, design innovations or style" (Miller 1988: 283). Firms with this sort of strategy are recognisable as Miles' and Snow's prospectors. Then there are those that seek to achieve a unique image by marketing rather than through the nature of the product itself.

These first two generic strategies may be applied to gain competitive advantage across a number of different markets within an industry or even across several industries. Alternatively, either may be combined with a strategy of **focus**, referring to the selection of a narrow market segment. The focusser seeks to obtain advantage over more broadly-targeted competitors either through better serving customer needs within the particular segment, or else through exploiting differences in the costs of production that apply only to the particular segment. There are thus differentiation and cost-leadership variants of the generic strategy of focus, which is the opposite of 'strategic breadth'.

Porter (1985: 17) sees the three generic strategies as fundamentally different, from the point of view of deciding how a company is to actually achieve competitive advantage. He acknowledges that, up to a point, a firm can achieve differentiation and cost-leadership simultaneously, although usually only until a competitor outperforms in one direction or the other. Empirical research has shown that this normative stance is not applied in practice (Hambrick 1983, Miller 1988). Actual strategic priorities appear to place their emphases on cost leadership and differentiation in a variety of proportions, and vary considerably in the 'strategic breadth' of the markets they serve. The diagram shown in Figure 5.1, which Porter uses

to illustrate the three generic strategies, can thus be taken as representing a two-dimensional continuum of strategic approaches. Effectively it adds the dimension of 'strategic breadth' to Miles' and Snow's one-dimensional continuum from pure defender to pure prospector, with the analyser somewhere in between.

FIGURE 5.1: A TWO-DIMENSIONAL CONTINUUM OF COMPETITIVE STRATEGY

	Type of Competitive Advantage	
	Cost Leadership	Differentiation
Wide	Cost Leadership	Differentiation
Strategic Breadth		
Narrow	Focus Cost Leadership	Focus Differentiation

The way that progress along the top of this continuum affects task environments has already been dealt with in the discussion of defenders, prospectors and analysers. In keeping with the same general notions, Porter too, for example, sees differentiators as above all concerned with dynamic task environments. They will seek technical innovations to give their products or services some qualitative edge. In contrast, cost-leaders are concerned with a stable set of conditions and with innovation that will lead to cost advantages. Whilst the sorts

innovation sought by cost-leaders are most likely to be in processes, product innovation can also be targeted on cost advantage rather than differentiation. The software house that develops a package product for an applications market where it has been seeking cost-leadership is an example of this.

Business Environments, Business Strategies and Task Environments

In this section we have seen a number of ways in which the business strategies of software-producing organisations can be characterised, and explored their implications for the sort of task environments which lower-level management control and mechanisms must address. This is not, however, to imply that all software-producing organisations have fully articulated business strategies that fit neatly into one of the categories described. Strategies may exist at the level of unspoken assumptions as to the sort of thing the organisation should be achieving. They may also not be appropriate to the business environment, as in reactors, and can even have contradictory elements. They may also be subject to rapid change as various powerful actors in an organisation try to swing the goals to be pursued one way and another. The use in the strategic typologies outlined here is however in terms of providing concepts for analysing whatever strategic choices emerge from the essentially political process of strategy formulation, and in understanding how they are associated with task environments which constrain the equally political processes by which internal systems of control and coordination are shaped. Child, Miles and Snow, and Porter, all imply that choice of business strategy is also to a large extent choice of task environment. Defenders or cost-leaders seek out stable task environments, whilst prospectors or innovative differentiators seek out dynamic task environments. Strategy would thus seem to be a major determinant of the degree of task environment stability experienced.

Hrebiniak and Joyce (1985) argue that in some industries there is much greater scope for choice of strategy than in others. Some

industries offer little choice other than to pursue cost-leadership, whilst others offer a wide range of choices. Still others may be highly regulated by government agencies, but still offer a range of strategic options, most commonly in the realm of focus. Hrebiniak and Joyce conclude that the degree of choice of strategy depends on both the nature of the industry, and the degree to which managers exploit the available opportunities for strategic choice. The extent to which business strategies are the products of business environments or the other way around need not concern us any further here. The point is that business strategies are linked to task environments. Together these two are the fundamental contingencies that organisational structure and systems of management control must address for successful performance.

5.4 MECHANISMS OF CONTROL AND CO-ORDINATION

This section is concerned with understanding the mechanisms of management control that may be used within software-producing organisations, in the context of various types of task environment and strategy. Just as the previous section outlined typologies for describing outcomes of processes of strategy formulation, this section provides typologies for analysing the outcomes of the political processes by which organisational structures and management control systems are shaped. It returns to the insights of the contingency theorists set out in the first part of this chapter to examine how choices of organisational structure, systems of control, types of human resources and systems of production technology may constrain one another. The discussion is based on the work of two authors. Galbraith (1977) has extended and enriched the range of contingency theory in analysing choices of organisational forms appropriate to environments involving considerable levels of uncertainty. He explicitly examines the sorts of mechanisms that can be used to control and coordinate professional technical work. Mintzberg

(1979) offers a synthesis of contingency theory, including the work of Galbraith, which also explicitly refers to the sorts of mechanisms used to control and coordinate professional work and the way various contingencies affect them.

We begin by examining how Galbraith and Mintzberg characterise 'professionalism' as a form of labour control appropriate to uncertain task environments. This is in reality a more precise formulation of Lawson's notion that professional technical workers are trusted employees who 'make the exercise of discretion predictable'. We move on to how Galbraith and Mintzberg see other mechanisms of control and coordination as needing to be added to 'professionalism' in order to cope with demands of different types of task environment. In this way we build up a picture of the various management control and coordination mechanisms that may exist alongside 'professional autonomy' in software producing organisations. We also examine what potential there is for manager-producer conflict with the various combinations of control and coordination mechanisms, and how they can be seen as shaping the division of labour.

'Professionalism' as a Mechanism for the Control of Work in 'Uncertain' and 'Complex' Task Environments

Galbraith sees professionalism as a form of work organisation suited to task environments where the level of uncertainty is too great for work to be conducted by the more traditional means of direct supervision through a hierarchy of authority, coupled with the use of rules, programmes and procedures, which serve to formalise behaviour. In this model, uncertainty is defined as the difference between the amount of information an organisation needs to accomplish a task, and what is available within it when the task is begun. The greater this information gap, the greater the amount of information that must be acquired and processed during execution, to deal with aspects of the task that the rules, programmes and procedures cannot. This leads to a greater burden on the management hierarchy in processing information and

making decisions.

According to Galbraith, professionalism is an alternative way of organising more uncertain work. This is first of all because professionals have dependable skills that allow them to use discretion in gathering and processing information and making decisions without the need to refer exceptional cases up the hierarchy. Second, they have responsible attitudes which mean that they can be trusted with decision-making. Galbraith thus sees professionalism as a mechanism for controlling the performance of individual workers charged with a certain kind of task.

Mintzberg also sees professionalism as a mechanism for performance control, but relates it to complex task environments. He defines a complex task environment as one which requires a considerable degree of sophisticated knowledge. Application of sophisticated knowledge requires decentralisation of decision-making to experts who are specialised in dealing with particular aspects of the task environment. All the knowledge needed to deal with the task environment cannot be enshrined in rules and procedures for less skilled operatives to apply, nor mastered and applied by managers in a controlling hierarchy. Professional training and orientation is a way of controlling individual performance in such circumstances. According to Mintzberg, the greater the complexity of the task environment, the greater the breadth of knowledge required to cope with it, and the greater the range of professional specialisms that must be brought to bear.

Both Galbraith and Mintzberg thus relate professionalism as a means of performance control to characteristics of the organisation's environment. Mintzberg's connection of professionalism with task complexity rather than uncertainty is significant. Another way of putting this is to say he relates it to technology in Perrow's sense of 'materials technology'. 'Professionalism may be seen as occurring as a form of labour control in software producing organisations because they have

complex task environments, or a 'materials technology' where search procedures are unanalysable. Mintzberg thus sees professionalism as related to an aspect of the task environment that is distinct from its degree of unpredictability. He sees it as linked more to the kind of knowledge needed to perform tasks. Since business strategy affects unpredictability of the task environment rather than its complexity per se, the emphases in business strategy would not then be expected to be a major determinant of whether or not an organisation uses 'professionalism' as a form of labour control. Accordingly, Mintzberg's understanding of what is involved when an organisation moves away from 'professional' control of labour concerns changes in the type of knowledge required rather than in the predictability of the task environment.

He sees professionalism as a mode of control as related to the absence of sophisticated or regulating instruments for the conduct of work. If management succeed in finding a technical system which can be used to perform the organisation's task by regulating the way work is done, the task environment is no longer complex (Mintzberg 1979: 268). Such a technical system enshrines knowledge previously exercised by professionals and allows work to be broken down into easily-comprehended components. It may therefore require skilled professionals in a technostructure to design and maintain it, but it no longer requires them to operate it. It becomes possible to control and coordinate work through specifying routine tasks in terms of rules and procedures, monitoring operator performance directly and rewarding rule compliance. The introduction of a routinising technical system therefore shifts an organisation based on 'professional' work organisation to one based on control and coordination through 'behaviour formalisation', a form which Mintzberg calls the 'Machine Bureaucracy'.

In the case of software-producing organisations, the evidence summarised in Chapter 3 indicates that, although structured techniques are to some extent regulating of the conduct of work, they do not on the whole eliminate the need for 'professional'

discretion. Nor is their introduction on the whole motivated by a desire to establish behaviour formalisation as the main form of individual performance control.

An understanding of reasons for introducing structured techniques thus most probably lies in terms other than as a mechanism for reducing task complexity so that the mode of control can be shifted to one appropriate to less skilled labour. It is however conceivable that a software-producing organisation operating with a strategy of cost-leadership in a stable market, customising packages in fairly predictable ways could utilise structured techniques in this way. This would however imply the retention of some professional-level employees in a technostructure, developing the packages and the structured techniques to be used in customising them, such that their use by less-qualified labour would not affect product quality.

Coordination Through Standardisation of Skills

According to Mintzberg, in stable but complex environments, for example in a general hospital, a viable form of work organisation consists simply of professional specialists left largely to their own devices. Standards of skill and knowledge acquired in professional training will ensure that different members of the same specialism deal with similar cases in similar ways. Professionals can be assumed to work to the same goals, and even though they have autonomy in choosing the means to be used, they will also choose similar means. In stable environments, they are likely to develop rules for the application of their complex knowledge. The common variants of the tasks they do and problems they solve become classified or 'pigeon-holed.' Each case has a routine that meets its needs. Complex skills are still needed, however, to make sure that pigeon-holing is accurate, and to deal with anomolous cases.

In describing this form of work organisation, Mintzberg for the most part assumes that professionals work closely with clients on

a one-to-one basis rather than with colleagues. Interdependencies between tasks are largely pooled, and the professional is in effect only dependent on the organisation for resources needed to practice. The need for coordination stems only from the requirement that standards are uniform, that similar cases are dealt with in similar ways. If there is also a need for cases to be dealt with by more than one specialism, there is some sequential interdependence between tasks. However, in a stable environment, the standardisation of skills of each specialism means that one sort of practitioner can know exactly what is expected of them by another, and also exactly the sort of areas of a case that a colleague from a different specialism will have dealt with. There is very little need for communication. Coordination is achieved through rules, but those that originate from the exercise of professional autonomy along familiar channels. Mintzberg calls this form of work organisation 'Professional Bureaucracy'.

It is then likely that the system of coordination in a software-producing organisation employing specialists engaged on sequentially dependent tasks in a stable task environment would consist simply of the standardisation of professional skills. This would be combined with professional control of individual performance, and a system of rewards required to reproduce professional commitment to work, and to achieving organisational success. Such mechanisms would mean that managers would only need to maintain overall performance control of groups or departments in simple financial terms. This sort of organisation would appear most likely to be associated with a strategy of 'defending', or cost-leadership, and the stable task environment implied. Examples would be an organisation employing a few different types of specialists to supply fairly routine minor customisations of a standard package in a stable market, or consultancy and bespoke development in an established and stable market in a fairly predictable application area, such as financial and administration systems for small and medium sized businesses.

In such a context, the introduction of structured techniques can be understood as a form of standardisation of skills, as a way of achieving coordination between systems analysts, systems designers, programmers, and those responsible for maintaining programs. Such techniques make the output of one type of software producer more readily comprehensible to a colleague. This fits well with the indications from the software management literature referred to in Chapter 3, as to the motivations behind introduction of these techniques. Their introduction would then involve an expectation that professional technical workers adhere to a framework of rules, but without abandoning basic discretion in their work and moving to total behaviour formalisation, or replacing the system of control of individual performance based on commitment to use complex skills with discretion.

According to Mintzberg, there is no fundamental conflict between the principle of professional autonomy and such a use of rules for standardisation of skills. This is because the rules stem from the development of professional practice itself under stable conditions, rather than from the management of particular employing organisations. Professionals develop 'pigeon-holes' and 'recipies' to fill them, which become diffused through the professional community and accepted by practitioners through initial training and subsequent professional gatherings. How well this describes the situation for professional technical workers where reference to a professional community is weak is an open question.

In the case of structured techniques and software engineering in general, the evidence from the literature is that initiation in their use tends to lie with organisational management. On one hand it is conceivable that direct producers would resent their introduction, seeing them as an infringement of desired autonomy. On the other, they may see them as the arrival of a truly professional approach. It may be conjectured that reactions will depend a great deal on the light in which these techniques are presented, and the way they are related to ideas of professional autonomy. The issues of whether structured techniques function as

a mechanism for coordination of professional work in this manner, or of whether they are used to shift the functioning of an organisation to that of a 'machine bureaucracy' by eliminating professional discretion, and the way software producers react to them, will form of a major focus of the empirical research outlined in the next chapter.

Coordination Through Action Planning

Using Mintzberg's framework, as task environments get more unpredictable or dynamic, coordination through standardisation of skills is no longer adequate. Sequential interdependencies between tasks can no longer be handled by one specialism knowing in advance exactly the sort of thing the previous one will have done, because the sorts of tasks or projects undertaken are much more variable, and each one involves considerable novelty. It is necessary for management to introduce 'action planning'. This means that individual professionals, or functional or product-based groups of professionals have tasks set in terms of technical specifications for what is to be achieved, together with targets of time and cost. Specifications are designed so that each sub-task will fit into a larger task, product or project. Time and cost targets communicate commercial criteria, and also coordinate the sequence in which sub-units of a large project are to be produced. In Mintzberg's terminology, as a mechanism for coordinating professional work action planning is akin to standardisation of outputs, and is used in addition to standardisation of skills. It is less restrictive of professional discretion than the third form of standardisation which can function as a means of coordination, that of work processes themselves (Mintzberg 1979: 154).

Galbraith describes an exactly similar form of coordination as essential for the control of professional work in a business organisation. He sees the use of specifications and targets as the basis for ensuring that professional discretion is used to achieve commercially viable outcomes, as opposed to the sort of

technical novelty or perfection that might be valued according to the standards of a professional community (Galbraith 1977: 45). It is worth noting that, whilst Galbraith sees 'professionalism' and 'planning' as a response to 'uncertainty' in a task, Mintzberg sees them as called for by a combination of two independent dimensions of task uncertainty, complexity and variability. When there is only complexity, standardisation of skills without action planning is sufficient for coordination.

Mintzberg indicates a number of ways in which the system of action planning may be linked to the system of overall performance control for units or departments. For an organisation based around a few large projects, senior management may start from action plans for what a set of product-based or functionally-based units should be doing. They may use them to derive a set of performance targets and budgets for projects or units within them. Alternatively, an organisation might consist of several different sub-units engaged in different markets, each with a large number of smaller projects. Each unit might then have its own budget and financial performance criteria set by senior management. It would carry out internal action planning on its projects, so as to meet its financial performance targets (Mintzberg 1979: 157). Whether an organisation starts with action-plans or financial performance criteria as a mechanism for controlling overall performance, if it is using action planning as its basic co-ordinative mechanism, specifications and targets given to individual professionals or groups can be used as a basis for monitoring performance and replanning if slippage should occur. They also serve as a means of motivating performance. In Mintzberg's terms, such specifications serve as both 'action plans' and 'performance control' mechanisms. In Woodward's, they are both 'performance mechanisms' and 'adjustment mechanisms'.

This sort of combination of control of individual performance through 'professional' commitment with coordination through action planning might then be expected in a software-producing organisation with a moderately unpredictable or variable task

environment. An example might be an organisation developing new systems in an application where requirements are fairly variable and there is incremental technical innovation which has to be taken account of in the sort of systems users expect. Another could be an organisation developing and customising packages in a similar environment, where the extent of customisation required would be fairly extensive. Such organisations could be 'focussers' or 'innovative differentiators' working moderately dynamic markets.

In terms of the sorts of conflict expected in such organisations, Galbraith's analysis of the need for action planning suggests that there is the potential for professional technical workers to resent the specifications posed on their work for commercial ends. Ritti's (1971) analysis of systems engineers, suggests however that professional technical workers see specifications of function, cost and time as an inevitable part of the work they must do if they are involved in developing technical systems to be marketed, sold and used. At the same time, Ritti's work indicates that morale suffers if professional technical workers are given specifications to work to that do not extend them sufficiently, in the sense not only of technical challenge but also time demands. The issue of the conditions under which specifications lead to manager-worker conflict in software production is then an issue which requires empirical exploration.

Coordination Through Lateral Relations or Mutual Adjustment Between Functional Groups

In Mintzberg's terms, as task environments become more dynamic, action planning itself becomes less effective for coordinating professional technical work. As the work to be done becomes more unpredictable, interdependencies between the work of one professional and another become more and more complicated. Work may have to be passed backwards and forwards between a number of specialists as a problem discovered by one has consequences for what another has just done or proposed to do. Interdependencies

between tasks become more and more reciprocal. There is then a need to use co-ordinative mechanisms that permit mutual adjustment. Galbraith offers a similar analysis, but in terms of increasing task uncertainty rather than environmental dynamism. As the information gap involved in performing tasks increases, organisations can increase their capacity to process information by introducing lateral relations between individual professionals in different specialisms or functional groups, so that exceptional cases can be dealt with directly, rather than being referred up the management hierarchy.

Galbraith offers a rich typology of mechanisms for establishing lateral relations between functions, starting from mechanisms that permit less intensive interaction at lower cost to the organisation, moving to those that permit more intensive interaction at higher cost. This runs as follows: direct contact between different specialisms or functional groups, establishing liaison roles, setting up temporary task forces, employing permanent product- or project-based teams or standing committees, creating linking roles, and creating managing linking roles. The most complex and intensive form of lateral relations is to set up a matrix organisation where professionals report in two directions at once. They have a functional manager who assesses their individual performance, development needs and career progress. At the same time they report to the manager of a multi-disciplinary project team.

Vertical Information Systems

Galbraith identifies a further mechanism by which organisations can increase their capacity for processing information to deal with uncertain tasks. They can invest in systems for gathering information and processing it, increasing the capacity of managers for dealing with exceptional cases, which fall outside the provisions of plans and which cannot be dealt with through the knowledge of individual direct producers. Galbraith sees this as an alternative to the creation of lateral relations, which

fulfils the same purpose. Creating vertical information systems is essentially a way of increasing the capacity of managers to digest performance relative to action plans, and to replan in the light of unpredictable task characteristics. In the case of software-producing organisations, it would appear to be applicable to firms operating with differentiation or prospector strategies in dynamic environments, where tasks change and are unpredictable. It would take the form of the use of computerised information systems to record progress and schedule the allocation of work. A possible area of conflict with 'professional' orientations is the way that such a mechanism would require formal reporting and could place the responsibility for scheduling the work of individuals with middle management. This could clash with 'professional' preferences for organising the allocation and monitoring of individual's work within a team, with performance monitored by middle management in aggregate terms only.

Galbraith identifies three further alternative 'organising modes' which can be used to improve the effectiveness of control and coordination if task uncertainty is great. They are all ways of reducing the amount of information the management hierarchy has to process, as opposed to increasing its capacity to process information.

Creation of Self-Contained Sub-Tasks

According to Galbraith, if several different functions are involved in producing a number of different products, the information-processing required for each product can be reduced by putting those functions in a product-based grouping. This reduces the diversity of calls on a single function. It also means that coordination between functions and the handling of exceptional cases occurs nearer to source, handled by managers in a product grouping. Galbraith is in fact describing the sort of differentiated organisational structures identified by Lawrence and Lorsch (1969).

This form of organisation would appear to be an option for software-producing organisations with business strategies directing them to multiple products or projects, with considerable uncertainty in each one. The improved coordination achieved by product-based groupings does however have costs and tensions (Galbraith 1977: 167). From the management point of view, it can require duplications of specialists in two groups when one might have done otherwise. It also ties professional technical workers to products rather than their profession, and can lead to frustrations with lack of professional development and contact with people from the same occupational group. On the other hand, this mechanism permits a great deal of local discretion in groups or teams, and seems highly compatible with typical orientations of professional technical workers. Mintzberg describes a project-based organisational form suitable for the most unpredictable task environments. He labels it 'Adhocracy'.

As environments become more and more dynamic and one project less and less like the previous one, Mintzberg sees the whole basis for coordination shifting from action-planning implemented through a conventional functionally-based management hierarchy to mutual adjustment between professional specialisms in project groups. In such an 'Adhocracy', each successive project requires a different combination of professionals. Organisational structures are fluid, and matrix based. The composition of product-based teams varies as projects come and go. The main form of overall control of project groups takes the form of setting financial performance targets in terms of revenue and costs. Within this framework, groups must be allowed autonomy in deciding the sort of outputs they should be producing and the means employed to do so. Action planning is only used close to the realm of direct production, and is in constant need of revision within the project team as the unexpected is encountered.

Such organisational forms might be expected in software-producing organisations that are 'prospectors' or 'innovative differentia-

tors', which have strategies of developing new systems in markets characterised by rapid innovation in software technology. Whilst the use of mutual adjustment for coordination would appear to harmonise well with a professional preference for discretion and autonomy, Mintzberg's discussion of 'adhocracy' also indicates the nature of tensions that may be experienced by direct producers.

Professional technical workers may feel considerable ambiguity in their position in a matrix organisation, as to who their boss is, as to who is evaluating their performance for promotion, and by which criteria. They may worry about what they will be doing next, and how well the organisation is going to serve their ambitions for career development. A characteristic of highly unstable environments is that the position of a firm in the market cannot be taken for granted, nor can the evenness of demand itself. The supply of orders to a particular firm is unpredictable. Professionals may not know what they will be doing beyond the end of their current project.

In general, in such circumstances it is a major concern of the 'strategic apex' to be monitoring the environment and managing the image of the firm so that an even supply of orders is achieved. However, staff are likely to find themselves underutilised at times, overstretched at others, and asked to do kinds of work they do not value at times, and given considerable challenges at others. Overall, unless functional management place a great deal of emphasis on supplying training and looking after professional development, direct producers in this sort of organisation are likely to acquire skills in a haphazard fashion according to the projects they are assigned to. There may be tensions between the fluidity the organisation requires and the need of professionals for consistent challenge and career development.

Creation of Slack Resources

Galbraith's second mode for reducing the amount of information that has to be processed during task performance is deploying more resources for a sub-task, in terms of time or numbers of people. This gives more room for dealing with deviations from the expected without having to replan or refer problems up the management hierarchy. Uncertainty and the need for tight coordination are reduced simply by lowering performance standards. This mechanism would appear to be another possible mode of coordination appropriate to 'innovative differentiator' or 'prospector' software-producing organisations, operating in dynamic or variable environments. Ritti's study, however, suggests that using slack resources has its contradictions from the point of view of direct producers' morale. Professional technical workers who do not feel fully utilised are likely to be less motivated in their work than those that are.

Environmental Management

Galbraith's third way of reducing the uncertainty facing an organisation refers to a variety of mechanisms by which it can seek to change the behaviour of organisations in its environment, making its own task environment less uncertain, or, in Mintzberg's terms, more stable. One particular aspect of this is relevant to understanding how tensions can arise between managers and direct producers in software-producing organisations. This concerns how organisations can deal with the 'production smoothing' problem, caused by the variability in demand which is likely in a changeable and dynamic product market environment, in comparison to their more fixed resource of direct producers.

When describing the 'adhocracy,' we referred to the problem produced by unpredictable demand when managers feel they have to keep a certain proportion of staff available in order to take advantage of new orders. At times these staff will be either underutilised, or as Ritti (1971) puts it, misutilised - employed

on 'make-work', tidying up the end of projects that are basically finished - whilst the organisation waits for new orders. Whether professional technical workers do not have enough to do, or have work which is not challenging enough or 'going anywhere' in project terms, dissatisfaction and tensions are likely to result. This unused capacity can however function as a slack resource available to assist other projects that get into difficulty. At the same time, it represents less than optimal utilisation of the workforce.

According to Galbraith, there are a number of mechanisms for dealing with the 'production smoothing' problem apart from the use of slack. The simplest method, which does not actually amount to managing the environment, is to develop input or output buffers. Output buffers are only possible for a product which can be completely standardised i.e. for software-producing organisations producing standard packages which require no customisation whatsoever. For any other organisation, software-producing or customising personnel could only be buffered against a variable demand by building up stocks of orders. This strategy would however depend on customers being prepared to wait for systems longer than the time actually taken to develop them.

In addition to buffering, or in cases when it is not possible, the following smoothing mechanisms are available, which can more properly be regarded as forms of environmental management. First, the effective size of the workforce can be varied so that the 'core' of producers can be kept smaller and fully-utilised for a larger portion of time. Producers can be asked to work overtime as a way of coping with demand peaks, or sub-contractors can be used. The latter option depends on suitably skilled producers being available for short-term contract work at viable rates. Second, the organisation can attempt to influence demand itself. It can promote its products or services when a period of slack appears imminent. If the problem is too much demand at particular times, it can simply ration what it takes on. The third set of options involves cooperating in some form with key customers. A large software-producing organisation can enter into an implicit

agreement with a set of smaller ones that they will deal with business peaks that it cannot handle. The larger organisation thereby gains a very stable workload, at the expense of great uncertainty experienced by the others. Alternatively, a software-producing organisation can enter into a contracting relationship with a customer with persistent demand for software, where a continuous flow of work for the producer can be negotiated in return for greater influence over what is produced on the part of the customer. Finally, a software producer can merge with a large user organisation, so that evenness of workload can be negotiated internally. This is the situation for a corporate in-house data processing department, for example.

The Shaping of Systems of Control and Coordination

Both Mintzberg and Galbraith see the array of organising mechanisms they describe as essentially tools available for managers or designers of organisations to choose from in dealing with the demands of their task environment. For any given task environment, managers' choice will be guided first of all by what they see to be the relative costs and benefits of different mechanisms. For an organisation using professional control to deal with a complexity in its task environment, 'uncertain' or 'unstable' conditions will make managers consider whether the costs of creating lateral relations or investing in vertical information systems are worth the return, or whether it will be more effective to utilise slack resources, set up self-contained units, or utilise some form of environmental management. All of these alternatives have costs as well as benefits in coordinating and controlling activities.

However, drawing on Child's (1972) argument, if the demands of the environment are not very rigorous, then there may be room for value-based choice between organising modes.

There is recent evidence to support a modified version of contingency theory, which sees the character of business strategy

and the task environment stemming from it as loosely determinant of organisational form. Miller (1988) presents a study which suggests that successful organisations with strategies of differentiation are driven towards particular organisational features.

Using a sample of 89 Canadian firms, he investigates the relationship between firms' competitive strategy - in terms of its position on scales of cost leadership - as opposed to differentiation, and strategic breadth as opposed to focus - with both business environments and the organisational structure utilised. His findings on the whole confirm that organisation is in fact related to strategy rather than business environment per se, and that this relationship is stronger for better performing organisations. Since several different strategies may be viable in the same environment, several different types of structure may also be viable, provided that they are associated with different competitive strategies. This reaffirms the importance attached here to understanding business strategy as a basis for understanding management control concerns and mechanisms in software-producing organisations.

In terms of how specific competitive strategies shape structure and process, Miller found that a strong strategy of innovative differentiation was associated with the use of highly-trained technocrats, and the use of 'organic' liaison devices. Miller hypothesises that breadth in competitive strategy should be linked to the use of formal controls such as budgets to control sub-units, delegation of authority to sub-units, and the use of information systems devised by technocrats and liaison devices to ensure coordination amongst diverse product units, whilst focus should produce much simpler administrative structures and forms of control. His data do not however lend any support to this characterisation of the organisational impact of the strategic breadth - focus dimension.

It is nonetheless possible to advance a hypothesis concerning the implications of a strategy of focus in software-producing

organisations. A strategy of focus in a labour-intensive industry such as software, implies that a firm would seek to establish either cost or differentiation advantage within a particular area of computer applications through skills of its direct producers and managers that are specific to that application area. Since the strategy of focus is based on a firm becoming established as the best supplier in the target niche, it is unlikely to have competitors which value these specialised skills to the same extent. As a consequence, it is very likely that many of the skills developed within it will be effectively firm-specific. Doeringer and Piore (1971) have argued that a firm which seeks to develop firm-specific skills is most likely to recruit inexperienced practitioners, invest in their training, and then fill vacancies for more senior staff through an internal labour market. In this way, its staff can develop expertise through training and on-the-job learning, and are enticed to stay by the promise of internal promotions. Meanwhile the company preserves the investment it has made in training.

It would appear likely that this sort of arrangement would be set up in a 'focusser' software house. It is also possible that a lack of promotion opportunities at higher levels and a lack of comparable outside job possibilities would lead to resentment.

CONCLUSION

This chapter has explored available typologies for describing the strategic concerns of software producing organisations, and the sorts of mechanisms of control and coordination that may be associated with task environments stemming from different types of strategy. It has also discussed the types of tension and conflict that may arise between management and software producers in the context of different types of control and coordination mechanisms.

It has focussed primarily on the way that various organising mechanisms may be used alongside 'professional autonomy' as a form of labour control. It has provided concepts for understanding how the second type of conflict context referred to in Chapter 4 comes about, when management concerns other than labour control act against professionals' preferences, and undermine the working conditions associated with reproduction of "professional responsibility" as a form of labour control.

The next chapter uses the concepts of this chapter and the one preceding to formulate a set of questions for empirical research into the nature of conflict and the shaping of the division of labour in software production.

CHAPTER 6

RESEARCH FRAMEWORK AND METHODOLOGY

INTRODUCTION

Chapters 4 and 5 have developed concepts for understanding the nature of professional technical workers' goals and orientations, and the nature of management concerns in software-producing organisations. In this chapter, a conceptual framework that summarises the discussion of the preceding chapters is set out in the first section. In the second section, we explore the implications of this framework for our basic research questions leading to a more detailed set of questions for empirical research. The third section describes the research methodology and approach to data analysis.

6.1 RESEARCH FRAMEWORK

The research framework starts from the position that an understanding of the nature of conflict between managers and software producers and of the forces shaping the development of software occupations at the level of the enterprise has to be rooted in analyses of two interacting sets of phenomena. On the one hand there are managerial goals within enterprises and the systems of control and coordination serving them. On the other hand there are the orientations of professional workers, and the way they react to mechanisms of management control and coordination and initiatives for change. In this first section of the chapter, we look at ways of conceptualising these two phenomena. The second section sets out the issues for research arising from a consideration of their interaction.

The Orientations of Software Producers

Taking the second aspect first, the orientations of software producers are expected to be along the lines of those of other groups of professional technical workers described by Ritti (1971). Software producers are expected to desire to use their skills with discretion and responsibility and without close supervision, to make visible contributions to organisational success, and to have a 'local cosmopolitan' orientation to their careers. This means that they seek technical challenge in their work, but respect the commercial as well as the technical aspects of the targets set for them by managers, and are motivated by the promise of organisational advancement because it leads to greater technical responsibility and 'professional' eminence.

The Nature of Management Control and Coordination Concerns and Mechanisms

With regard to the way that systems of management control and coordination are shaped under different conditions, we start from the position that they are the products of two basic levels of negotiation and choice, in which organisational actors come to grips with constraints. First, business strategies are chosen. Together with any strategies for shaping organisations' environments, they determine the task environments which the internal functioning of organisations must address. Managers make choices constrained by their perceptions of business environments and the expectations of resource providers, but also find scope for exercise of preferences and values in deciding between alternatives (Child 1972). Second, internal systems of control and coordination are shaped to meet the demands of task environments, again with scope for values and preferences to manifest themselves.

Considering, first of all, the forms that systems of control of individual performance can take there are two extreme emphases. Control can be achieved through giving professional technical workers discretion in how they approach their work, within specifications and targets set by management. Provided that workers feel that targets are reasonable, but that the work is challenging and will lead to advancement in terms of greater rewards and greater technical responsibility, they will be motivated to meet specifications and targets without the need for intensive monitoring. Alternatively, control can be achieved by emphasising behaviour formalisation in a system of 'bureaucratic control' (Edwards 1978, Greenbaum 1979). Actual work procedures are specified to a much greater extent, and commitment to their application is ensured by close monitoring of progress and rule-adherence, and a system of internal promotions which rewards rule compliance.

Bureaucratic control can only be effective when the task of software production can be made simple or routinised, and when tasks do not change rapidly i.e. when the task environment is on the whole simple and stable (Mintzberg 1979). The rules and procedures on which bureaucratic performance control is based can also serve as a way of coordinating the work of different producers. If the work of individual producers is specified in considerable detail by managers, coordination can be built into the rules and procedures by which producers work. In contrast, the 'professional' form of performance control relies on other mechanisms for coordination. The mechanisms that are effective are likely to depend on the complexity of the task environment - how much non-codified knowledge needs to be applied during its execution - and the extent to which it is dynamic or changing.

Figure 6.1 summarises the part of the framework concerning the relationship between management control and coordination mechanisms, business strategy and task environment developed in the last chapter. It shows how the two types of control of individual performance may be related to different mechanisms for coordinating work and controlling performance above the level of

FIGURE 6.1: MANAGEMENT CONCERNS AND MECHANISMS FOR CONTROL AND COORDINATION

Competitive Strategy	Task Environment		Organisational Structure	Mechanisms for Control of Individual Performance	Mechanisms for Coordination and Higher level Control
	Stability (Predictability)	Complexity (Comprehensibility)			
Cost Leadership	High	Simple (structural techniques used for defining work procedures)	Functional	Bureaucratic Control	Behaviour Formalisation Rules and Procedures, Hierarchy of Authority (Machine Bureaucracy
		Complex (producers decide work procedures)	Functional	Professional Control	Standardisation of skills, Control of departments in terms of outputs and costs
Increasing emphasis in differentiation through innovation	Increasing unpredictability	Degree of complexity reflected by areas of knowledge required for task execution	Functional	Professional Control	Action Planning, plus control of departments through outputs and costs
			Functional	Professional Control	Lateral Relations, plus control of departments through outputs and costs
Innovative differentiation	Low (Rapid changes in type of product, greater uncertainty/ unpredictability in technical problems encountered)		Product or Matrix	Professional Control	Creation of self-contained tasks, controlled through outputs and costs. Plus any of: 1. Vertical Information Systems 2. Slack Resources 3. Environmental Management

the individual. It also indicates how different emphases in business strategy of software-producing organisations and the types of task environment associated with them are likely to influence the combinations of control and coordination mechanisms that are effective.

Software-producing organisations whose business strategy can be characterised as cost-leadership (Porter 1980, 1985) will be associated with stable task environments, in that there will be little change in products or the types of technical problems that are encountered in their production. Whether or not they have task environments that are simple or complex will depend on the availability of structured techniques which can make it possible to specify in advance how work should be performed, removing the need for discretion on the part of the direct producer. If such techniques are available and used, the control of individual performance and coordination can be achieved through bureaucratic control, consisting of behaviour formalisation according to rules and procedures administered and monitored by a hierarchy of authority (Mintzberg 1979). This is likely to involve the employment of less educated and cheaper labour power than would be used for 'professional' control (Kraft 1977).

If it is not possible to specify in advance how work should be performed, or thought by management not to be desirable to do so, the accomplishment of tasks depends on professional discretion. Control of individual performance is then most effectively carried out according to the 'professional model', where commitment to work is mobilised by technical challenge and the promise of advancement. In a software-producing organisation that has introduced 'bureaucratic control' for the control of some portion of its software workforce, it is likely that not all the tasks of software production will be able to be routinised. People will be needed who can apply knowledge with discretion to design the procedures used by those subject to bureaucratic control (Johnson 1980). These producers will retain their 'professional' autonomy, and be controlled by means of their commitment to work. Since the efficiency of bureaucratic control

depends on utilising less costly labour than 'professional control', the two different emphases of performance control imply the presence of two different types of worker, with different orientations and expectations. This implies two separate career structures, differentiated 'vertically', and drawn from two separate labour markets.

In Chapter 4 we saw that the use of internal promotion ladders was generally associated with the use of both 'professional' and 'bureaucratic' principles of control of work in organisations. To the extent that there are internal promotion ladders within each segment of the workforce, and that higher positions within a segment are filled by junior staff rather than by external recruits, these career structures can be regarded as internal labour markets (Loveridge and Mok 1979, Doeringer and Piore 1977). Internal labour markets are more likely to arise where there are firm specific skills, because it is in the interest of the organisation to seek to keep staff by offering opportunities for advancement.

For organisations using 'professional' labour control, provided that the task environment is stable, so that the sort of work being undertaken does not vary much, coordination can be achieved by the use of structured techniques as a form of standardisation of skills. As work is passed from specialist to specialist, provided that what one does conforms to the structured standards, it is readily comprehensible to the next. These mechanisms of control and coordination at the level of direct producers are likely to be used in conjunction with overall control of performance at functional department level in financial terms.

As business strategies of software-producing organisations are concerned more and more with innovative differentiation (Porter 1980), their task environments become more and more dynamic, in that products change more rapidly and the new types of technical problems are encountered more frequently, with less predictability. Task environments in such cases are likely to be both dynamic and complex. The more complex the task, the more

areas of knowledge that will be needed to execute it, and the greater the number of specialisms that will need to be involved, economies of scale permitting.

The lower section of Figure 6.1 illustrates the sorts of organisational structures, systems of control and coordination that are likely to be effective as the emphasis on innovative differentiation in strategy increases, and the 'dynamism' or unpredictability of the task environment increases correspondingly. For moderate levels of change in the task environment, a system of action planning for a functional organisational structure is likely to be sufficient for its coordination, with the specifications and targets produced in such a system serving as a mechanism for monitoring and controlling the performance of departments or units within them.

As dynamism increases, there will be a need for more and more intensive lateral relations between functional departments (Galbraith 1977). Functional departments may also need to become differentiated in their systems of control and coordination due to different levels of unpredictability and complexity in their sub-tasks (Lawrence and Lorsch 1967). Alternatively, there may be a move towards differentiated project or product structures, which combine different functional specialisms. Project or product groupings may be permanent or temporary, for extremely changeable product market conditions, and may be formed in the context of a matrix organisational structure. Such forms of organisation represent the creation of self-contained sub-tasks as a mode of coordination, which will be used in combination with action-planning within the project or product groupings. Overall control of the performance of each grouping will be achieved using performance targets set in financial terms where a grouping is dealing with a large number of small projects, or in terms of project targets where it is assigned a small number of large projects to develop.

The combination of action-planning with lateral relations or creation of self-contained sub-tasks may be supplemented by the use of slack resources, vertical information systems and environmental mananagement to deal with conditions that are highly changeable in terms of products required and technical problems encountered (Galbraith 1977).

The Purpose of the Framework

The concepts set out in this framework provide a basis for understanding both the nature of conflict between software producers and their managers, and the way that software occupations, in the sense of divisions of labour and the nature of career structures, are shaped at the level of the enterprise. They permit analysis of forms of control and coordination that managers may adopt under different circumstances of business strategy and task environment. They raise issues as to the sorts of conflict that may be associated with the combination of 'professional' labour control with various coordination mechanisms.

They also provide a basis for analysing how managers may react to changing conditions, such as the emergence of new products and markets, or the 'maturation' of markets for previously innovative products. They permit an understanding of how divisions of labour, systems of control and coordination are shaped at the level of the enterprise in response to such industry-level developments. For example, they suggest the circumstances under which managers may move from 'professional' labour control to vertical segmentation of the workforce and the introduction of 'bureaucratic' control. They also raise issues as to how conflicts occurring within a framework of 'professional' labour control may lead to developments in the division of labour.

We now turn to the empirical investigation designed to explore these issues.

6.2 RESEARCH DESIGN AND RESEARCH QUESTIONS

In the introductory chapter we identified some fundamental research issues: the nature of conflict between direct producers and managers in software-producing organisations; the forces shaping the development of software occupations and the stability of 'high trust' forms of work organisation and the relationship of 'trusted' occupational groups to professional organisation. In the empirical work of this thesis the framework set out above is used for developing an understanding of these issues at one particular point in time within a small number of contrasted organisations. In each case-study the following issues are examined: business strategies and task environments; the current state of occupations in terms of the division of labour and the labour markets from which positions within it are filled; systems of management control and coordination; worker orientations; the problems, tensions and conflicts experienced by managers at various levels and by different categories of direct producer; and any developments in divisions of labour and job structures that appear to be imminent as a consequence of strategies adopted by organisational actors to deal with problems and tensions experienced.

It is recognised that this research design is limited in two important senses for the purpose of addressing the overall research issues. First, it cannot be claimed that a small number of case studies are representative of the great diversity of software-producing organisations described in Chapter 3. Second, an understanding of the current state of software occupations at the level of the firm within a few organisations, even when combined with an indication of tensions and developments that may stem from them in the near future, cannot be claimed to give a picture of the long-term development of software occupations as a whole.

The rationale for a case-study focus is twofold. First, it will test the adequacy of the framework for understanding 'synchronic' processes at one point in time within particular organisations and allows a development and refinement of theoretical categories. Second, it should provide a basis for conclusions to be drawn as to the sorts of factors within enterprises which affect the nature of conflict, developments in the division of labour, and the prospects for occupational control. These can be used to draw up some hypotheses as to the way developments in software occupations, labour markets and occupational organisation will be affected over time by developments at the level of the firm. These should provide material for future research on a more comprehensive scale, using survey methodologies, to identify how widespread the phenomena and developments uncovered in the case studies are.

The rest of this section sets out a more precise set of research questions for the case studies, stemming from the conceptual framework of 6.1. Questions concerning processes at one point in time within organisations fall under five headings. It is not proposed to formulate the issues concerning the longer term development of software occupations in any more detail at this point. An understanding of how to conceptualise developments in this respect will be sought in the final chapter.

I. Business Strategies and Task Environments

The framework suggests that management control systems in a software-producing organisation have to be understood in the context of its task environment. The first issue is then to identify types of task environment and the way they are shaped by business strategies. The framework suggests definite relationships between the type of business strategy being pursued and the type of task environment. This leads to the following research question:

What are the characteristics of the task environments of the organisations studied, and how are they shaped by the type of business strategy being pursued?

II. Organisational Structures, Career Structures, Mechanisms of Control and Coordination

The framework implies that the way a software-producing organisation is structured, the form that career structures take within it, and the systems of control and coordination used, can together be understood as responses to the characteristics of its task environment. An understanding of these phenomena is important first of all because they shape the work experiences of software producers and are the immediate context of any conflicts or tensions. Second, an understanding of career structures in particular is important because it gives a picture of the current state of software-producing occupations, for example whether there is any vertical segmentation of labour markets. An understanding of the current state of career structures and labour markets is vital for analysing the nature of conflict over control of work in the present, as well as for assessing the likelihood of various lines of future development.

The following research questions stem from the framework:

What are the types of (a) divisions of labour and career structures, and (b) organisational structures, and mechanisms of control and coordination currently existing? How can they be understood as responses to task environments?

III. The Nature of The Orientations to Work of Software Producers

In Chapter 4 we outlined the nature of the work-related goals attributed to professional technical workers by Ritti (1971), which can be seen as underlying 'professional control' as a form

of labour control. We also noted the need to understand better how the work-related goals described by Ritti are turned into a commitment to targets set by organisational managers. The questions raised for research in the case studies are as follows:

> To what extent are the work-related goals of direct producers in the various job ladders for direct producers in each organisation in line with those described for 'systems engineers' by Ritti? How are they related to the establishment of commitment to project targets, and what are the issues in reproducing this commitment?

IV. The Nature of Conflict

The framework suggests that there are two distinct contexts in which conflict may occur between managers and software-producers who enjoy discretion and autonomy in the conduct of their work. First, managers may be dissatisfied with the level of performance

achieved from direct producers at the individual level. If they can see the technical feasibility of replacing an emphasis on 'professional' discretion with bureaucratic control of cheaper labour through behaviour formalisation, at least in some areas of work, conflict may arise from 'professional' workers' desires to retain their control over work. Second, conflicts can arise even when managers are satisfied with the levels of productivity and quality obtained through 'professional' control and are content to retain this form of labour control. There may be clashes between the sorts of worker orientation bound up with 'professional' control of individual performance, and the sorts of working practices called for by coordination mechanisms, such as standardisation of skills, required to supplement 'professional' control under different circumstances. There may also be clashes over standards of quality sought by management, as expressed in specifications, and the sorts of quality of work aspired to by direct producers.

The basic questions for investigation in the case studies concerning conflict may be formulated as follows:

> What forms of conflict exist between direct producers and managers, and how are they related to the nature of worker orientations and the systems of management control and coordination used to address business conditions? Does producer-manager conflict take the form of struggles for the control of work stemming from the desire of management to increase individual productivity, or are conflicts the result of working conditions stemming from mechanisms for coordination of work?

A subsidiary set of questions concern conflict over the introduction and use of structured techniques. The framework implies that they can function as either a means of behaviour formalisation, part of establishing 'bureaucratic control' over work processes, or as tools for the 'standardisation of skills' in the context of 'professional' control. The questions raised are as follows:

> What are management views of the function of structured techniques or other software engineering methodologies, and how do these vary according to business conditions? What are their consequences for direct producers in terms of the respects in which they control the conduct of their work? Are such techniques the site of conflict between managers and direct producers?

V. The Shaping of Software Occupations at Enterprise Level

The framework implies two different dynamics behind the shaping of software occupations at the level of the enterprise. One the one hand, there are developments stemming from within the enterprise. 'Professional control

is likely to be associated with a variety of types of conflict between workers and managers. Managers may seek to eradicate them by changing the organisation to make professional control subject to fewer contradications, or by moving away from it towards bureaucratic control. On the other hand, changing products and markets may lead to new business strategies and task environments, and new forms of control, co-ordination and divisions of labour.

The basic questions for research here are as follows:

> How can any strategies for change articulated by managers within the organisation be understood as a response to the tensions and conficts associated with 'professional' control of labour? To what extent can strategies for change be seen as a response to changes in business environments? What are their likely impacts in terms of modes of labour control and the nature of career structures? Does 'professional' control of labour appear to be stable, or unlikely to last?

More particularly, the framework raises the following questions about motivations behind the introduction of structured techniques and their impact:

> Are intentions of introducing structured techniques associated with intentions of shifting the mode of labour control away from professional autonomy, or with concerns of improving coordination? In either case, what are the likely impacts of the techniques proposed on the control of work and the nature of career structures and labour markets drawn on?

Overview of the Case Studies

These research questions serve to provide a focus for investigation in the case studies, which aim to explore links between competitive strategy, management control systems, worker

orientations, tensions and conflicts, and developments in work organisation and occupations. The final section of this chapter describes the areas of data required to do so, the methodology employed to obtain them, and the approach to data analysis.

6.3 RESEARCH METHODOLOGY

Selection of Cases

The decision to research in-depth relationships between business strategies, mechanisms of control and coordination, conflict and development in work organisation and the division of labour, limited the number of cases it was possible to attempt within the scope of a doctoral research project. Cases were selected on the basis that they should be software-developing organisations contrasted in terms of the nature of their products and types of business environment. No attempt was made, however, to make representation comprehensive on the basis of any exhaustive typology. It was decided to focus on applications software, so that all the organisations involved would have tasks of comparable complexity in that they would include assessment of user requirements, where users were not computer professionals or even necessarily 'computer literate'. It was anticipated that each study would provide opportunities to investigate how more 'technically oriented' software staff relate to their more 'user-oriented' colleagues, and provide insights into the pressures and conflicts experienced by each.

It was decided to conduct an equal number of studies of data processing organisations and organisations producing more 'technical' types of software. This was justified on the basis that the two types appear to draw on largely distinct labour markets and types of software staff.

In the case of organisations producing data processing software, it was decided to include at least one example in each of the two basic business environments identified for data processing organistions in Chapter 3. One case was to be of an independent software house selling software on the market and one of an in-house corporate data processing department, providing services to the operating divisions of a business organisation. The initial direction for selection of the more 'technical' cases was that they should be differentiated in terms of the size of systems developed and the degree of customisation involved in the systems sold. It was anticipated that these differences would be associated with differences in task environment and management control concerns.

In the event, cooperation was secured to conduct four in-depth studies, differentiated according to these very broad criteria. Three were in fact separate divisions of a single corporation which developed and manufactured a variety of electronic and telecommunications systems in the U.K., many of them software-based. Access to these three was gained through a recently-appointed Manager of Software Engineering, a new corporate staff position reporting directly to the Technical Director. This manager was known to be considering the issue of management in software, and expressed interest in the project. He secured cooperation for some exploratory interviews with four operating divisions engaged in developing and manufacturing products incorporating a significant amount of software. Two of these led to an agreement to proceed with in-depth studies. The third case where agreement was obtained was the corporate data processing division which came under the Finance Directorate. It supplied data processing services to support the activities of the corporate centre and its various operating divisions.

The remaining case was a division of a large independent UK software house, concerned with selling customised data processing packages to manufacturing companies. Access was obtained through a middle manager within this division who was known to be interested in the issues concerned, and who was prepared to

convince his colleagues that an in-depth study should be undertaken.

Data Collection

Each organisation studied was found to contain a number of different business units concerned with different markets or products. It was decided to focus on one unit or area in each case.

Data in six main areas was collected from managers and team leaders within the selected business units:

A. Products, markets and business strategy and performance goals of the organisation as a whole, and of the selected business unit.

B. Type of software development projects being undertaken, in terms of size, whether one-off systems, package developments or customisations.

C. Organisational structure of the whole company, and detailed composition and structure of the selected business unit. Types of software development methodology used.

D. Mechanisms for controlling and coordinating work to meet performance goals associated with the task environment, including use of: hierarchy of authority; behaviour formalisation; standardisation of skill; performance targets and action-planning; vertical information collecting and replanning; lateral relations; product-based groupings; environmental management. Differences in control and coordination mechanisms used at different hierarchical levels.

E. Problems and issues identified by managers at various levels in the selected business unit.

F. Strategies for change or improvement being pursued by managers at various levels.

These data were obtained through unstructured interviews with managers and development team leaders in the business unit, using the *aide memoire* in Appendix 1. Each interview was begun by a description of the broad aims of the research, by giving guarantees of confidentiality, and offering the interviewee the option of not proceeding further if they were concerned with the nature of the project.

In general, strategic information was obtained from the more senior managers, and details of products, organisational structure and systems of control were filled out at middle management and team leader level. In the cases of the two 'product divisions' of the large corporation, there were some issues in terms of gaining access to managers responsible for overall strategic decisions of the selected business unit. In each case, the point of entry was a middle manager responsible for a set of development teams, who reported to a manager of the business unit as a whole. This sort of 'technical' manager was the level at which the corporate Manager of Software Engineering usually liaised. Managers at this level could provide information about forms of organisation and management control and arrange interviews with team leaders and direct producers within their area, as well as provide a considerable amount of data on strategic considerations. They were however in neither case willing to persuade their superior with strategic responsibility to give time for an interview, at which a more direct view of strategic concerns and performance objectives might have been obtained.

This was not considered a major inadequacy in the data collected. Each case study was concerned with understanding management control concerns and mechanisms as they impacted direct producers, and the way management control concerns were linked to strategy and overall performance requirements. It was therefore

quite appropriate to take middle management perceptions of strategic concerns as a basis for understanding the context in which systems of control were shaped. In the cases of the corporate data processing division and the independent software house, direct access to senior managers was achieved.

On completion of data collection from managers and team leaders, a set of part-structured interviews with as many direct producers as possible within the business unit was negotiated. In two cases the middle manager responsible for the development teams within the business unit agreed that all direct producers could be interviewed. Actual coverage was somewhat lower due to people not being available on the days allocated. In the other two cases, middle managers were more reluctant to allocate direct producers' time, and about half were covered, giving a total of 57 interviews across the four cases. In addition, a further 11 interviews were conducted with staff in three other 'technical' software organisations where it had proved possible to obtain only very limited data from management.

The part-structured interviews collected data in four areas:

A. Academic background, career histories, training, current specialism.

B. Orientations to work, goals in terms of type of current work desired and career aspirations, according to the dimensions identified by Ritti (1971).

C. Extent to which experience of current work measures up to work goals. Degree of control over work and autonomy from supervision experienced. Perceptions of effects of structured techniques on achievement of work goals.

D. Anticipated promotions and career path. Extent to which prospects measure up to career goals.

The interview schedule used is shown in Appendix 2. Interviewees were informed of the aims of the research by a memorandum from a middle manager. Staff were interviewed at times arranged by management, within working time and in a spare office near where they worked. They were given guarantees of confidentiality and asked if they wanted to withdraw, but none did so. It is possible that some had asked not to be interviewed when first contacted by management, however. The schedule was piloted on the first six interviewees in the first case study undertaken, and then revised, with some additional questions. The additional questions were put to the first six subjects by post.

In addition, data was sought on patterns of supervision and the extent of discretion and autonomy enjoyed, as well as on the size and variety of projects. The use of direct observation was considered for the first class of data, but rejected because of the time demanded and also because middle managers were not keen on allowing it. They saw it as intrusive, and were convinced that it would not lead to any real data because 'there's nothing to see, just people sitting at screens and talking to each other round the vending machine". It was decided to give interview subjects diary sheets. On the basis of the experiences with diary sheets administered to managers by Stewart (1963), sheets which required simple tick responses were used. Over a ten week period following the interview, subjects were asked to fill in one sheet per week giving details of the sort of work undertaken that week. They were also asked to fill in one sheet per day on a different day of the week for ten weeks, to show the sorts of contacts they had with their team leaders and colleagues. The diary sheets used appear in Appendix 3.

The success of the diaries as an instrument of data collection was mixed. A few interviewees refused to fill them in. Others expressed willingness, but apologised for having forgotten, when visited for collection of the sheets after five and ten weeks. Others still offered completed sheets, but said they had forgotten to fill them in at the time and had made up entries on what they reckoned was a typical day for them. Bearing in mind

Stewart's conclusion that individuals can find it very difficult to judge the pattern of their contacts accurately after the event, these diaries were discarded. Overall, diary sheets for some part of the ten week period were obtained from 34 (60%) of the 57 members of the total sample employed within the four case-study organisations.

Data Analysis

Qualitative data from the management interviews and from the less structured questions on the interview schedule, and qualitative data from other portions of the interview schedule, were initially analysed by writing a description of each case study. It was decided that the best way to understand what mechanisms of control and coordination were in use was to describe in what terms targets were set and monitored at each hierarchical level, and what mechanisms served to make their achievement predictable. This task was guided by a set of questions, listed in Appendix 3.

This initial data analysis led to two main forms of final data analysis.

First, it was apparent that, across the four cases and the three organisations where data was obtained only from producers, work-related goals were very similar in character. It was therefore possible to deal with the issues concerning the nature of work related goals and the extent to which they corresponded to those identified by Ritti i.e. most of the research questions set out under heading III in the previous section, across the whole sample of producers. This sample-wide analysis of quantitative data on work-related goals is set out in the brief first chapter of the data analysis section. SPSS was used for the statistical analysis.

Second, the concepts set out in section 6.1 were used to develop an analytical framework for presenting the data in each case so as to address the remaining research questions. This analytical

framework conceptualises relationships between business environment, business strategy, task environment, and systems of control and coordination on the one hand. On the other, it relates these to producer goals, experiences of work, forms of conflict and developments in the organisation of work and occupational groupings.

The framework takes the form of five sections of data analysis, corresponding to the five sets of research questions. The relationships between the various phenonema described in each section are indicated by Figure 6.2. The sections, and the concepts used in each, are as follows.

1. Business Strategy and Task Environment

This section describes:

- the terms in which resource providers, in each case a corporate parent organisation, set **performance expectations** for the selected business unit.

- the **business strategy** adopted, in terms of products and services, and the degree to which performance goals were set in terms of **cost-leadership** as opposed to **differentiation**, and their **strategic breadth.**

- any **environmental management strategy** adopted, in terms of **cooperation** with other organisations, establishing long-term **contracting** relationships, **co-opting** other agencies, and **coalition.**

- the enacted **task environment**, in terms of the degree of **complexity** and **unpredictability** of its **commercial** and **technical** aspects.

2. Organisational Structure, Career Structures, Mechanisms of Control and Co-ordination

This section describes:

- the **organisational structure** of the selected business unit, in terms of the respects in which it is a **functional**, **product-based**, or a **matrix** type.

- the respects in which **divisions of labour** within sub-units or departments where software-producers work are **vertical** as opposed to **horizontal**, and the types of software producer employed.

- the nature of **career structures**; the number of parallel or **horizontally segmented** job structures, corresponding to different specialist career paths; whether or not there is also **vertical segmentation** within any of these; and the extent to which senior positions are filled from **internal labour markets**.

- the mechanisms used by senior managers to achieve **overall control** of the main departments, in terms of **financial performance control**, and **action-planning**.

- the mechanisms used to **control individual performance** of software producers, i.e. the balance of **behaviour formalisation**, and **professional commitment** to meeting targets.

- the mechanisms used to achieve **lower-level coordination** within software-producing departments, including use of **action-planning**, **standardisation of skills**, and **lateral relations**.

- the mechanisms used to **adjust plans** when unpredictable circumstances were encountered, including use of **vertical information systems**, and **slack resources**.

3. Mechanisms for Producing Commitment to Work

This section describes the mechanisms managers identified for mobilising the general work-related goals of software producers, described in Chapter 7, into commitment to meeting project targets, including:

- offering discretion and technical challenge in work.

- offering promotion as a reward for responsible exercise of discretion.

4. Tensions and Conflicts.

This section describes:

- the **issues perceived by managers** in the functioning of their organisation, in terms of the extent to which they see **quality and productivity of individual performance**, the **reproduction of commitment to project targets**, the **coordination of work** or **future product strategy** as priority issues needing to be tackled.

- the **tensions and conflicts experienced by software producers**, in terms of the extent to which the requirements of the form of organisation adopted meet their work-related goals, in particular the extent to which they experience **productivity and quality targets as too stringent as opposed to not challenging enough.**

5. Strategies for Change and Possible Outcomes

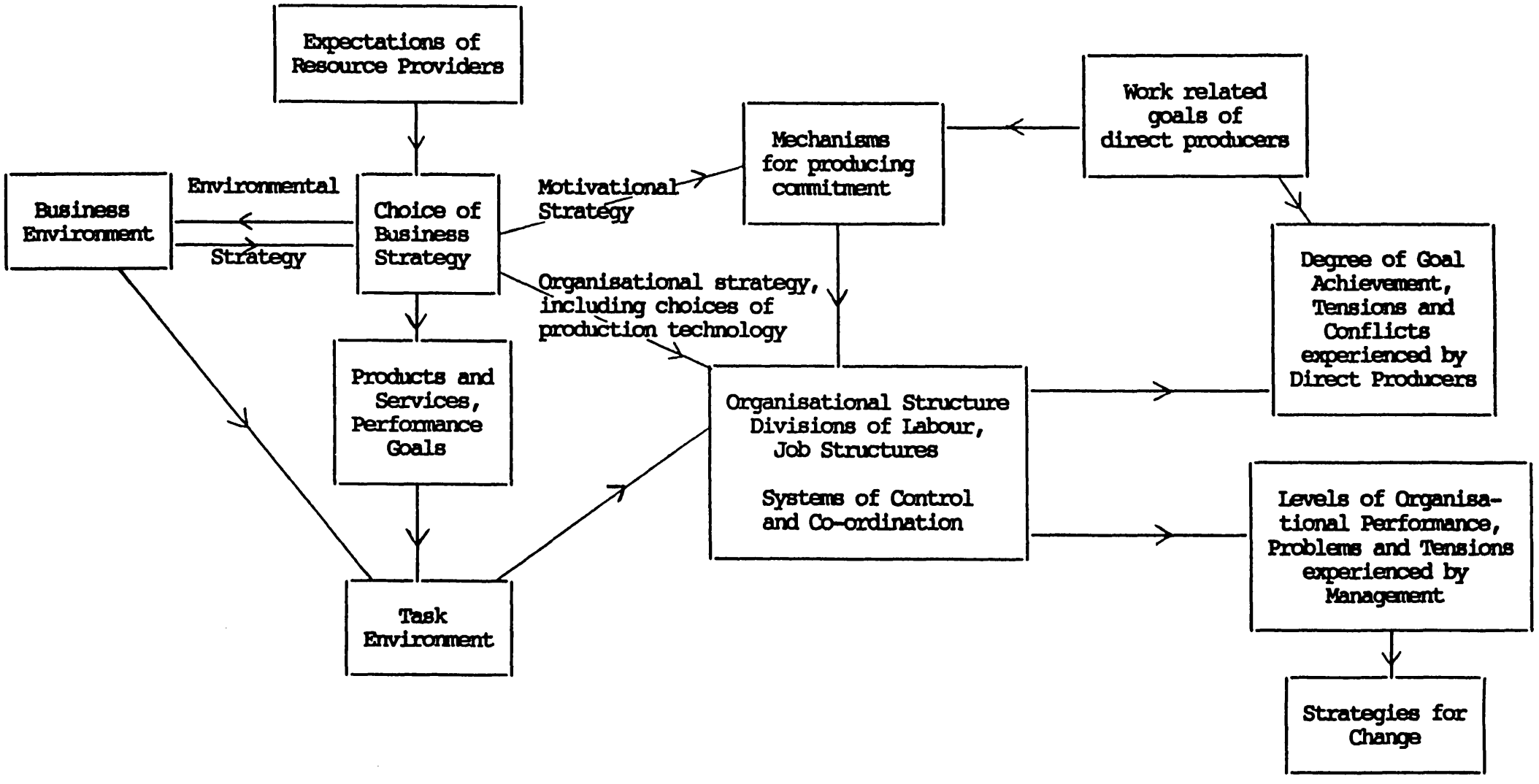
This section is concerned with **management strategies for change**, and their likely impact for the **division of labour, mode of labour control, and the nature of job structures.** In particular it focusses on:

- intentions with respect to new **structured techniques**, and whether these are likely to be associated with **vertical segmentation** of previously continuous career structures.

- issues or problems in the **continuation of 'professional' control of labour**, even where there is no intention of undermining it.

The remaining four chapters of the data analysis section apply the analytical framework to each case study in turn. Chapter 12 draws conclusions as to what answers the case studies provide to the research questions.

FIGURE 6.2: ANALYTICAL FRAMEWORK FOR THE CASE STUDIES



CHAPTER 7

THE WORK-RELATED GOALS OF SOFTWARE PRODUCERS

Introduction

This very brief chapter is concerned with the quantitative data obtained from the sample of 68 direct producers of software as to the importance they attached to various work-related goals. The focus of the analysis is establishing the extent to which their goals were similar to those identified by Ritti (1971) in a much larger sample of 'systems engineers' in the USA.

Ritti found that systems engineers had goals of skill utilisation, having control over technical decisions, making a clearly visible individual contribution, and contributing to organisational success. Such 'local cosmopolitan' goals make them amenable to working diligently and productively within a business organisation, without perceiving a fundamental conflict between organisational goals and their own. We have argued that this commitment to organisational goals can be seen as one side of a bargain, in which such professional technical workers expect challenge and autonomy in their work in return for their commitment.

As part of answering the third set of research questions set out in the previous chapter, and as a basis for understanding the nature of conflict in the case studies that follow, we are interested here in establishing whether or not the software producers in our sample had goals of this sort.

7.1 Goals In Current Work

Table 7.1 summarises data obtained from the sample as to how much they agreed with a range of statements concerning the importance of various aspects of the content of their work. For each statement there were no statistically significant differences between the average scores across the different organisations. This suggests that patterns of priorities reflected by the sample-wide mean scores shown can be taken as indicating the preferences of the sample as a whole. They

indicate that considerable weight was attached to having 'the opportunity to show what I can do', and to having 'the freedom to adopt my own approach', with having 'an impact on business success' also considered to be important , but rather less so.

A Friedman test indicated that the differences in ranking of these priorities across the sample were highly statistically significant at less that 0.001 probability. Utilisation was clearly preferred to having trivial or routine projects, and freedom of approach to having detailed instructions.

Table 7.2 further illuminates goals in current work by correlating the degree to which various aspects of work were actually experienced with two measures of overall achievement of work-related goals. The aspects of work listed down the left hand side of the table were all measured in terms of the percentage of time for which respondents felt them to hold true. The only exceptions to this were the extent of time and technical demands, which were measured on a scale of 1 (extremely light) to 7 (extremely heavy), and the assessment of salary levels. Respondents were asked to compare their current salary with that offered for comparable jobs in the same area, on a scale of 1 (much lower) to 7 (much higher). The two statements of goal achievement were assessed using a scale of 1 (not all) to 4 (satisfactorily) and to 7 (completely).

There were statistically significant correlations between overall goal achievement and having 'the opportunity to show what I can do', the strength of technical demands in the job, the extent to which producers felt they were 'making an important contribution' as individuals, the extent to which they felt they were 'contributing to business success', and the extent to which they were 'making the most important decisions within my area of responsibility'. There was also a significant negative correlation between goal achievement and projects being 'trivial or routine'.

This is reasonable support for the applicability of most of the components of Ritti's 'local cosmopolitan' characterisation of work goals of professional technical workers, i.e. the desire to utilise skills, make a visible contribution, have an impact on business success and control over the conduct or work. The data however suggest some qualifications to the basic pattern of goals identified by Ritti.

First, the importance of skill utilisation in the sense of work offering technical challenge appears to be clearly established. The weak positive correlation of the extent to which the time demands of work were heavy is also interesting. The absence of a negative correlation suggests a preference for wanting to do more rather than less at work. Software producers apparently had norms as to how hard they wanted to work which they were concerned that they should be given the chance to satisfy. There is then evidence that software producers had work-related goals that were conducive to internalised control of both the technical quality of work and productivity.

Second, whilst 'making an important contribution' and 'contributing to business success' were correlated with one measure of overall goal achievement, 'participating in business decisions' was not linked to either measure. This suggests that the sort of organisational impact and contribution being sought was on the whole in technical terms. Producers wanted to be influential in the technical aspects of projects and to feel they contributed to success through doing a technically good job, rather than seeking to influence commercial decisions per se. Ritti's 'local cosmopolitan' label appears to be highly appropriate to these software producers. Their basic goal was to use technical skills on successful projects. The organisational and commercial underpinning of such projects rather than an interest in business aspects per se appears to have led to a commitment to organisational success.

Third, the data on relationships between goal achievement and various aspects of control of the work process are illuminating. Whilst it was important to these producers to feel that they were 'making the most important decisions' concerning their work, they did not appear to attach as much weight to being able to 'use their own approach', to being free from 'specifications that had been decided by others', or from having to 'check with others before they proceded.' Having more control over the approach to work, or being freer from specifications did not lead to greater goal achievement. However, being able to make the most important technical decisions did. The implication is that it was possible to control the most important technical decisions, whilst being to some extent constrained by specifications and methodological standards. There was in fact no significant correlation between the measure of 'having the freedom to adopt my own approach to work', and that of 'making the most important decisions in my area of work'.

7.2 Career Goals

Table 7.3 illustrates producers' orientation to their employing organisation in terms of their career goals. It shows that, whilst these software producers were not particularly intent on developing a career within one company, they nevertheless saw it as important to 'achieve a good rate of promotion' and to 'develop skills and knowledge continuously.' A Friedman test indicated that the differences in ranking between the various career priorities were significant, at less than 0.001 probability across the sample.

These orientations are consistent with those identified in Ritti's sample, where the desire to gain more technical influence and utilise more demanding skills was perceived as being associated with gaining organisational advancement. It seems that for our sample too, the most important career priorities were developing skills and gaining organisational advancement. There does not seem to have been any dichotomy between the two.

People did not, however, see that organisational advancement had on the whole to be within their current organisation. They were interested in organisational advancement, but were prepared to move to find it.

Conclusion

We can summarise our findings with regard to the work-related goals of our sample of software producers as follows. Technically challenging work, influence over important technical decisions, the chance to make a visible individual contribution, and contributing to business success were all important goals. In terms of career development, the chance to develop skills and knowledge further and the achievement of organisational advancement were priorities.

Clearly, this sample of software producers possessed the sort of 'local cosmopolitan' work goals that make people amenable to a 'professional bargain' with the management of a business organisation - diligence and commitment to organisational goals in exchange for responsibility and the possibility of advancement. We may expect to find them in situations where they are controlled on the basis of professional responsibility, and where there is an absence of explicit conflict over norms of quality and effort.

In the case studies that follow, we shall see how these orientations were utilised in the control of work, how they were mobilised into commitment to project targets, and the conditions under which they were not satisfied giving rise to tensions and conflicts. We shall also be able to provide evidence linking desire to gain organisational advancement to desires for developing skills and knowledge, and for influence over technical decisions, rather than to a commitment to the organisation per se. We shall see how promotion ladders offered the chance to develop skills and exert greater technical influence, and so were attractive, but that software producers on the whole also

regarded finding advancement elsewhere as an option.

The data in Table 7.2 permits one further conclusion to be drawn, to set the scene for the case studies. The table shows how various measures for five different variables were correlated with overall goal achievement. No single measure, even those most strongly correlated with goal achievement, explained more than 16% of the variance in goal achievement. This suggests that goal achievement depended on a great variety of factors, beyond those represented by the quantitative data sunmmarised here. In this study we are not concerned with explaining in a comprehensive fashion the determinants of different levels of achievement of work-related goals. We are however interested in what affected feelings of conflict or tension between organisational and individual priorities. In the case studies we shall use qualitative data gained from software producers to understand the nature of factors producing feelings of conflict or tension, additional to those represented by the quantitative data analysed in this chapter.

TABLE 7.1: GOALS RELATING TO CURRENT WORK

Agreement on scale of 1 to 9:

1 - disagree completely, 5 - indifferent, 9 - agree completely.

It's important to have the chance to show what I can do:	7.6 (s.d.= 1.3)
I don't mind a task below my technical capacity:	5.6 (s.d.= 2.1)
It's important for me to have an impact on business success:	6.1 (s.d.= 1.7)
It's important for me to have freedom in my approach to work:	7.3 (s.d.= 1.5)
I mind having trivial or routine projects:	4.1 (s.d.=2. 3)
I prefer detail instructions as to how to proceed:	4.6 (s.d.= 2.3)

TABLE 7.2: GOAL SATISFACTION AND FEATURES OF CURRENT WORK

Kendall Correlation coefficients between two measures of overall goal satisfaction and measures of various aspects of current work (probabilities for purpose of significance tests shown in brackets: * indicates siginificance at less than 0.01 level) for 68 cases.

	Work is fulfilling most important goals.	Job is providing everything expected from a job.
<u>Skill Utilisation</u>		
Opportunity to show what I can do:	0.3951 (0.001) *	0.3300 (0.001) *
Time demands:	0.1316 (0.094)	0.1809 (0.034)
Technical demands:	0.3140 (0.001) *	0.2198 (0.013)
Projects are trivial or routine:	-0.2563 (0.004) *	-0.2594 (0.003) *
Tasks below technical capacity:	-0.1533 (0.058)	-0.2712 (0.002) *
<u>Control Over Work</u>		
Freedom to use own approach:	0.1279 (0.095)	0.2064 (0.016)
Making the most important decisions:	0.2470 (0.006) *	0.2733 (0.002) *
Specifications already agreed by others:	-0.0818 (0.198)	-0.1316 (0.084)
Every step has to be agreed beforehand:	0.0295 (0.380)	0.0150 (0.438)
Given detailed instructions:	0.0474 (0.317)	0.1208 (0.110)
<u>Visibility of Individual Contribution</u>		
Making an important contribution:	0.2672 (0.003) *	0.1390 (0.072)
<u>Contribution to Organisational Goals</u>		
Contributing to business success:	0.3130 (0.001) *	0.1938 (0.021)
Participating in business decisions:	0.2060 (0.021)	0.0756 (0.225)
<u>Pay</u>		
Salary:	0.0848 (0.183)	-0.0424 (0.352)

TABLE 7.3: CAREER GOALS

Importance of various career priorities on scale of 1 to 9, where:

1 = completely unimportant, 9 = extremely important.

Keeping earnings as high as possible:	5.3 (s.d.= 2.1)
Accepting lower pay for the present, to gain longer term prospects:	5.5 (s.d.= 2.0)
Developing a career in one company:	4.0 (s.d.= 2.1)
Continuing to use the same skills and knowledge:	4.9 (s.d.= 2.1)
Developing skills and knowledge:	8.0 (s.d.= 1.1)
Achieving a good rate of promotion:	6.4 (s.d.=1.6)

CHAPTER 8

TELEXCO

SUMMARY: TELEXCO

1. BUSINESS STRATEGY AND TASK ENVIRONMENT

Market: Developing and customising office communications systems.

Businesss Strategy: Focussed differentiation for established products. Innovative differentiation and collaboration for new products.

Task Environment: For established products, complex technically, stable technically and commercially in the short term. For new products, complex technically and commercially, unpredictable technically and commercially.

2. ORGANISATION, JOB STRUCTURES, CONTROL AND COORDINATION

Organisation: Functionally-based departments. Staff with software background in marketing department as well as main production department. Product-based teams within software dept, comprising 'designers' and 'programmers'.

Career Structures: Promotion from 'programming' to 'design' work in an internal labour market; horizontal specialisation only at senior levels, into 'product' (marketing) and 'technical' (development). Staffed predominantly by mathematically-based graduates.

Overall Control and Coordination: Action planning for each project, with plans drawn up in more decentralised way for existing products. Specifications and estimates drawn up by team leaders in direct lateral consultation with marketing staff. Some accumulation of orders as a buffer.

Labour Control: Achieved through 'professional' skills and commitment in the context of targets and specifications, backed up by informal monitoring by team leaders, milestone reviews. Standardisation of skills through structured programming assures quality, as does availability of help from team leaders/colleagues.

Low-Level Coordination: between designers and programmers achieved using structured standards; detailed coordination between team members by direct contact.

Adjustments to Plans: Information concerning occasional technical problems collected 'vertically' through informal monitoring: plans adjusted mainly by team leaders calling for extra effort.

3. MECHANISMS FOR PRODUCING COMMITMENT TO WORK

Finding the 'right people'. Offering skill development and training. Offering pay raises and promotion. Setting tight targets. Leading by example. Giving producers control over the sequencing of their work. Allowing producers to 'see a project through'.

4. ISSUES AND TENSIONS

For Managers: Finding the 'right people'. Keeping producers from leaving a project. Obtaining up-to-date technical and market information for new products.

For Producers: Lack of challenge and development opportunities for experienced people on established products. Lack of opportunity to learn software engineering techniques. Lack of influence in new collaborative project.

5. INITIATIVES FOR CHANGE AND LIKELY OUTCOMES

Initiatives: More rigorous structured techniques.

Possible Outcomes: Greater centralisation to design decisions to more senior people, but no move away from 'professional' control of labour and a single job structure.

8.1 BUSINESS STRATEGY AND TASK ENVIRONMENT

At the time of fieldwork, Telexco was one of four product divisions that made up the Business Systems Group of Telecorp plc. Telexco came into existence in the late 1970's. Under a different name, it had been engaged in developing and manufacturing a wide range of telecommunications switching devices, but shed most of its software development capacity at the end of the 70's. From then until just before the time of fieldwork it focussed its activities on a particular area: the development, manufacture and marketing of 'message-switching' systems. A message-switching system is a type of telecommunications exchange which handles the receipt, transmission and addressing of telex messages. Such a machine becomes necessary for an organisation that has more than one telex terminal for one telex number.

Expectations of Resource Providers and Established Business Strategy

Until a few months before the period of fieldwork, the performance of Telexco was scrutinised by the Telecorp Board through treating the Division as an autonomous profit centre. According to middle managers interviewed, it had been the Board's policy to give the general manager and senior management a free hand in conducting the business within agreed financial targets. Telexco managers decided when to develop new products or enhance existing ones, as well as their own marketing strategy and sales objectives, on the basis that they were in the best position to assess their complex technology and markets.

In the context of these financial performance expectations, Telexco pursued a business strategy of focussed differentation, marketing its products on the basis of its ability to meet the specialised needs of organisations in a selected handful of industries. It sold to large 'communication intensive'

organisations (mainly in the fields of defence, policing, air traffic control, finance, insurance and banking) able to invest hundreds of thousands of pounds in a large message-switching system. Telexco aimed to meet the industry- and firm-specific needs of customers - for example the volume of incoming and outgoing traffic, and the number of terminals - by customising one of two basic product designs. It also provided comprehensive installation and training services.

Changes in Business Strategy

In the twelve months prior to fieldwork, the corporate view of the business environment for business telecommunications systems in general changed, and led to a shift in performance expectations for Telexco. Technological developments were seen as destabilising existing product configurations and product markets, demanding a more integrated corporate approach, and greater corporate direction of product strategies within its divisions.

It became clear that the combined technologies of digital information processing and telecommunications, both based on programmable micro-electronic components, raised the possibility of an 'integrated office system'. Such a system was envisaged as consisting of a network of work-stations offering document preparation, data processing, internal voice telephony and electronic mail, as well as an interface with external telecommunications network. The external link would offer teletex, the successor to telex, along with other transmission services. In order to become established in the market for this sort of system at the outset it was decided to invest in a large scale pioneering product development, involving four Telecorp systems houses with experience in different types of information and communications systems, working cooperatively. Telexco was appointed the 'lead house'.

Together, the four divisions were expected to pursue a strategy of more pronounced innovative differentiation in order to become established in a new market. The corporate centre expected Telexco to lead a product design process coordinated largely by direct lateral relations between the divisions. Relationships and trade-offs between product features suggested on the basis of expertise in each division could then be explored directly, without involving the centre other than in evaluating the overall viability of a product specification once it had been produced.

Task Environment

These two aspects of business strategy, concerning established products and the new venture, led to two rather different task environments for Telexco.

First, there were activities associated with established message-switching products. There were two main systems being marketed. System A had first been developed in the early 70s, and had been through a number of enhancements. Based on a minicomputer with other custom-designed hardware elements and software written in Assembly language, it was becoming obsolete, because of greatly improved cost/processing power ratios achievable using microprocessor hardware. There was however still a steady stream of orders, and it was available in a number of different variants designed for the main classes of user organisation i.e. military, police, air-traffic control and banking. Whilst System A was in the last stages of its life-cycle as a revenue earner, the newer System B was in the middle of an extended central plateau stage. It had been developed four years previously, using software written in the 'technical' high-level language, PL1, running on standard microprocessors and was available in a number of specialised forms. For each one, a selection of features was offered, depending on price. The system as a whole was enhanced every two years, and it was seen as the principle revenue earner for Telexco for some years to come.

Activities on these systems were characterised by predictability in both their commercial and technical aspects. There was stability in terms of the level of demand, and reasonable predictability in terms of the sort of technical problems encountered. Once a product range had been built up, the small number of competitors and the continued demand for message-switching systems internationally meant that a steady stream of orders could be expected, at least in the short and medium term.

As well as selling and customising products, meeting the needs of customers included providing after-sale support to the customer, for as long as their system remained operational. Support was included in the selling price, and meant dealing with faults as well as staying in touch with the user and responding to minor changes in requirements. In both support and customisation tasks, considerable technical complexity was involved, given the intricacy of the real-time switching software involved. The portion of the task environment concerned with these products was complex also in its commercial aspects. Sophisticated knowledge was required to deal with approaches from potential clients, or to respond to tenders.

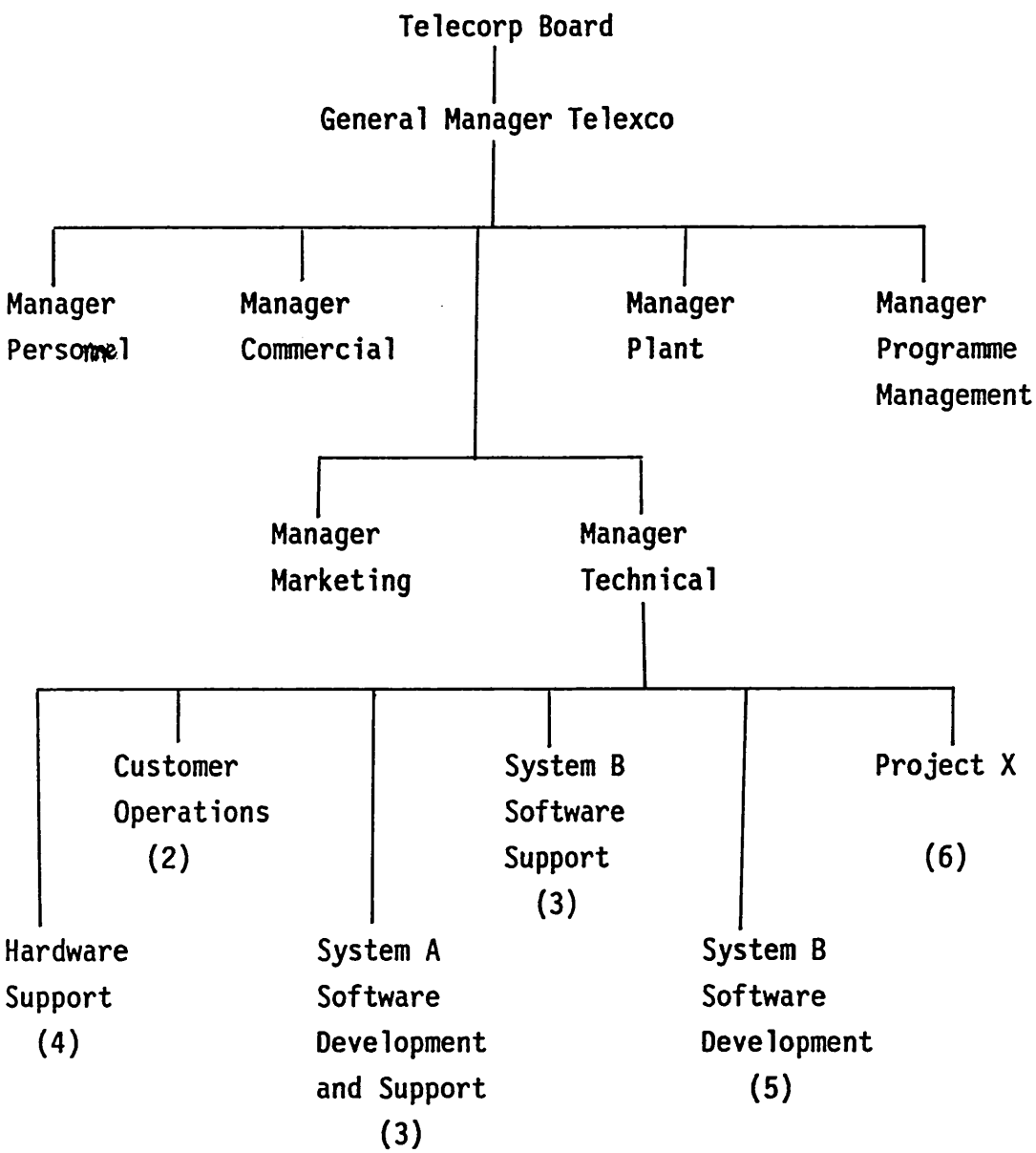
The task environment for the new office automation development, Project X, involved greater unpredictability in both its technical and commercial aspects. It concerned product markets which had yet to be defined, and technologies that were very new and developing rapidly. It also involved greater complexity in both technical and commercial respects. Technical complexity was increased because of the number of sub-systems involved. Commercial complexity was increased because of the number of formerly distinct product markets concerned, and also because cooperative relationships with three other systems houses had to be negotiated. In several respects the new venture represented a step-change in terms of both unpredictability and complexity.

8.2 ORGANISATION, DIVISIONS OF LABOUR AND JOB STRUCTURES

8.2.1 Departmental Structure

Figure 8.1 gives the organisational structure of Telexco.

Figure 8.1: Organisation of Telexco



(Numbers of staff in brackets)

We are concerned here with the organisation and control of the areas where software producers worked. These were the Marketing and Technical Departments. The functional division between the two was a response to the different complexities involved in keeping track of user needs and the ability of the available technologies to meet them, and in developing systems to meet user needs.

In addition to these departments, a number of functions dealt with more routine aspects of the organisation's tasks. A Commercial Department administered customer contracts, and a Plant Department commissioned the supply of assembled hardware for message-switching systems from sub-contractors, usually other Telecorp divisions. The latter was a routine operation because customisation of a basic message-switching system was achieved primarily through altering software. The Programme Managememt department was responsible for coordinating installation of systems and training at the client site. Installation of hardware was carried out by field engineers, whilst software was installed by software development staff from the Technical Department.

The Marketing Department consisted of five sales engineers, who had backgrounds as software producers. They marketed and negotiated sales of customised message-switching systems, mostly in response to tenders put out by client organisations. In addition Marketing had a market research function. On the basis of their experience of client needs, they would advise on features to be included when the Technical Department was designing new systems or enhancing existing ones.

As well as software development work for customisations, enhancements and new products, the **Technical Department** was responsible for maintaining installed systems. The remainder of this section concentrates on its organisation and functioning, since it was here that the bulk of software producers worked. In terms of numbers of staff, Telexco was a small-scale operation. Its Technical Department had 15 software staff engaged on modifying or existing message-switching designs or maintaining

installed systems, and 6 on the early stages of Project X.

8.2.2 Divisions of Labour Within The Technical Department

The Technical Department was organised on a product-centred basis, with teams responsible for each product. Each was headed by a team leader who was a productive member of the team, reporting to the Technical Manager. This product-based organisation can be seen as a response to the need to build specialised expertise in each product, so that enhancements and customisations could be performed by staff experienced in the particular system, and was permitted by the volume of activity called for on each product.

As Figure 8.1 shows, the volume of activity for System B was such that two teams could be justified, one for customisation work and one for maintenance and support, dealing with software faults in systems installed on client sites that proved too complex for Telexco's maintainance field engineers. For System A, one team was responsible for both customisations and maintenance. All but one of the members of the System A and System B teams had worked with their respective systems for the whole of the period they had been with Telexco. Apart from one member of the System B customisation team who had joined the Division six months previously, such periods varied from three to ten years.

The fourth and largest software team was responsible for the early stages of developing the new system that was to result from Project X. This team had been created in the months before fieldwork, and was staffed with experienced development staff transferred from System A teams as activity had run down.

In addition, there were two teams with functional responsibilities for two relatively predictable areas of operation, common to the processes of selling, customising and maintaining both message-switching systems. Hardware support was responsible for maintenance calls on the fairly standardised

hardware of installed systems. Customer Services was responsible for preparing manuals and training materials for users in the client organisation.

(a) Divisions of Labour on Teams Customising Existing Products

Within the customisation teams, once a specification was received from Marketing, projects were conducted in seven distinct phases, which together made up a top-down development methodology.

System design modification: in which the abstract logical design of the system as a whole was modified to take into account the customised features agreed upon for the requirements specification. An outline of the tests the system as a whole would be subject to, in order to demonstrate that it met the requirements specified was also prepared.

Module and Program design: the more detailed logical structure of each module was designed. Tests for each programme and module were also specified, designed to establish that it was performing the functions specified for it in the system design.

Coding: this included low level clarification of logical operations prior to actual coding.

Testing: carried out on a 'bottom up' basis. The code for each programme within a module was tested, and then each module as a whole. Finally an integration test of the whole system was carried out.

Documentation: the technical and operator manuals for the system were modified to take account of customised features.

Product Building: a memory disc giving details of the user's telex network was compiled, customising the system not only with respect to the operating features required but also the current configuration of the user's telex system.

Installation: the software was installed at the client site, and subjected to acceptance tests specified in the original contract. Any modifications arising from small changes in user requirements or errors in development were made.

Within the development teams, there was a division of labour between high level 'system designers' and 'programmers' i.e. those responsible for low-level design and coding. The difference between the two categories was one of experience. Junior team members expected to move from programming to progressively higher levels of design, at the same time advancing in terms of job-title and salary.

For smaller projects, typically customisations of up to six months development time, all of these stages were the reponsibility of one team member. This would be a more experienced member, with the job title of systems programmer or principle programmer (see Figure 8.2 for the progression of job titles in software work). For larger customisations of over six months' development time the first stage might be done by a more experienced team member, often the team leader himself, whilst lower level design and coding would be performed by one or two less experienced team members, under the guidance of the designer, as technical manager of a small sub-team. The designer would perform the final integration testing, and any one of the team might carry out installation. The team leader would however remain the business and administrative head of the project, responsible for its completion to time.

(b) Divisions Of Labour Within The Maintenance Teams

There was no phasing of the way fault-finding and 'patching' work was carried out in response to fault reports received by team leaders responsible for product support. In general, this work was regarded as extremely complex and difficult to systematise, involving an in-depth understanding of each bit of the software making up the product. There was however a division of labour based on experience. Team leaders would make an assessment of the complexity of the problem implied by a fault report, and then allocate it to a team member of the appropriate level of experience.

(c) Divisions Of Labour On Project X

Although in its early stages of requirements specification, it was intended to conduct the Telexco portion of Project X in the same basic stages as customisations to existing products. There was to be the same experience-based division of labour between 'designers' and 'programmers', although with larger numbers of staff. There were however some important differences, in terms of the sort of tasks that were being undertaken during the early stages.

First, since it was a product that aimed to be amongst the first in a new market, requirements anaylsis could not be approached in terms of modifying existing product paradigms. It had to be approached in a much more speculative mannner. The greater complexity of market research, requirements analysis, and specification was approached through a corresponding increase in lateral relations between software engineers within the technical department and marketing staff. Two experienced designers with knowledge of customer requirements were assigned to work with Marketing to evaluate technical aspects of different basic approaches to the product and the functions it would perform in terms of communications. The Technical Manager himself led the requirements specification for the whole project, and was

responsible for integrating decisions made in the other software houses with those at Telexco. The only area of requirements specification conducted as a fairly self contained job was that of the 'man-machine interface' by the most junior of the six system designers on the team.

Second, the development was a much larger undertaking, involving a large number of sub-systems, virtually products in themselves, developed in a number of different systems houses, but which had to be compatible with one another. With the project at an early stage of requirement analysis, a considerable amount of project effort was being devoted to devising techniques and methods by which this complexity could be negotiated. These included software engineering techniques and systems for monitoring and control.

Two members of the team were engaged on this sort of work. One was developing software tools for integrating and testing the different portions of the system one they had been developed. The second was responsible for developing an agreed development methodology with the other systems houses. The remaining two members of the team were engaged on the early stages of actual system design.

Once the requirements of the system had been specified, and at least part of the system design completed, it was expected to add 'low-level' designer/programmers to the team. One estimate was a team of 20 within six months.

8.2.3 Job Structures And Labour Markets

Table 8.1 shows the educational backgrounds and job histories of 16 of the 21 software producers employed within the Technical Department. It indicates that software producers came largely from maths or science degrees which had involved a fair amount of scientific computing. Generally, there was no difference between

TABLE 8.1: JOB HISTORIES OF TECHNICAL DEPARTMENT STAFF

Sex	Job Title	Age	Education	Nature of First Job	First SW Employment at Telexco	Position on joining Telexco	Length of time at Telexco	Professional Affiliation	TU Member
F	Programmer I	25-29	BA French	Programming	Yes	Graduate Programmer	3 years	None	No
M	Systems Programmer	20-24	B.Sc. Comp.Sci.	Programming	Yes	" "	3 years	None	No
M	" "	25-29	B.Sc. Comp.Sci	Programming	Yes	" "	3 years	BCS	No
M	" "	25-29	B.Sc. Comp.Sci	Programming	Yes	" "	5 years	None	No
M	" "	25-29	B.Sc. Mech.Eng.	Programming	Yes	" "	7 years	None	No
F	" "	30-35	B.Sc. Physics	Programming	Yes	" "	7 years	None	No
M	Principal Programmer	25-29	B.Sc. Comp. Sci	Programming	No	Principal Programmer	6 months	BCS,lapsed	No
M	" "	25-29	B.Sc. Chemistry	Programming	No	Graduate Programmer	8 years	None	TASS
M	" "	30-34	B.Sc. Maths	Programming	No	Programmer	8 years	None	No
M	" "	30-35	B.Sc. Maths	Programming	Yes	Graduate Programmer	10 years	None	No
M	Systems Consultant	35-39	B.Sc. Maths	Programming		Programmer	10 years	None	No
M	" "	35-39	Ph.D. Maths	Clerical Work	No	Programmer	8 years	None	No
M	Senior Software Consultant	30-35	Ph.D. Chemistry	Programming	No	Programmer	10 years	None	No
M	" "	35-39	B.Sc. Maths	Programming	No	Programmer	10 years	None	No
M	" "	30-35	B.A. Maths	Programming	Yes	Graduate Programmer	12 years	None	No
M	Senior Product Specialist	35-39	B.Sc. Maths	Programming	No	Programmer	10 years	Institute of Mathematics	No

the level or type of qualifications of senior 'designers' as compared to junior 'programmers'. The difference between senior and junior staff was primarily in terms of experience.

Job titles were more an indication of the level of experience and responsibility achieved by staff than a precise indication of their function. Although only about half the staff interviewed had started their work in real-time programming at Telexco, all those in the most senior positions had joined as 'programmers', and had worked their way up, so that they had been with the Division for at least 8 years. Professional affiliation or membership did not play any significant role in securing advancement. Only one of the sample was a member of the British Computer Society (BCS), and this out of interest only. Likewise, there was only one member of a trade union, TASS, out of a political commitment rather than any reason concerning the workplace.

There would appear, then, to have been no vertical segmentation of career structures, with one continuous promotion ladder, and also an internal labour market for the more senior positions. There was also no horizontal specialisation in career paths until the more senior levels.

Whilst staff stayed with a particular product during its 'life' of up to five years to gain expertise in it, they expected to move on to a new system after a few years. A product thus did not on its own offer a career structure. As more senior positions were reached there was a basic choice in career direction to be made. Staff could concentrate on the high-level aspects of system design as 'senior software consultants', or else move into the area of analysing user requirements, providing technical support to Marketing and managing product support teams as 'senior product specialists'. The next moves up from these positions were, respectively, into technical management or into marketing management.

Table 8.2: Job individuals expect to be in in five years time

	System A team leader	Custom-isation Designers	Custom-isation Prog/ Designers	Maint	Proj. X (All Designers)	**Overall**
N	1	4	2	3	6	16
	EDDDDDDDDEDDDDDDDEDDDDDDDEDDDDDDDEDDDDDDDE					
	3 1	3 1	3 2	3	3 3	7
SYS DES, TECH	3	3	3	3	3	
CONSULTANT	EDDDDDDDDEDDDDDDDEDDDDDDDEDDDDDDDEDDDDDDDE					
	3	3 2	3	3	1 3	3
SW DEV MGT	3	3	3	3	3	
i.e. TEAM LEADER	EDDDDDDDDEDDDDDDDEDDDDDDDEDDDDDDDEDDDDDDDE					
	3	3	3	3	1 3	1
SW SALES,	3	3	3	3	3	
MARKETING	EDDDDDDDDEDDDDDDDEDDDDDDDEDDDDDDDEDDDDDDDE					
	3	3	3	3 2	3	2
PROJ,CLIENT MGT	3	3	3	3	3	
	EDDDDDDDDEDDDDDDDEDDDDDDDEDDDDDDDEDDDDDDDE					
	3	3 1	3	3 1	3 1	3
SW DEPT MGT	3	3	3	3	3	
	EDDDDDDDDEDDDDDDDEDDDDDDDEDDDDDDDEDDDDDDDE					

Table 8.3: Career goals of software producers

Mean scores for importance on a scale of 1 to 9, broken down according to position in the division of labour and overall.

	System A team leader	Custom-isation Designers	Custom-isation Prog/ Designers	Maint	Proj. X (All Designers)	**Overall**
N	1	4	2	3	6	16
Developing a Career in one Company:	3.0	4.5	4.0	5.7	5.3	4.9 (s.d.=2.3)
View on transfer-ability of skills: (Scale: 1= 'not at all' to 7='completely')	6.0	5.0	4.0	6.3	5.7	5.5 (s.d.=1.4)

Table 8.2 indicates the range of expectations of staff as to the sort of work they would be doing in five years time, broken down according to their current roles. It shows expectations on the part of some development staff of achieving or of staying with 'senior designer' work, whilst others anticipated becoming product specialists or managers. The three maintenance staff interviewed all anticipated moving into management in some form.

Table 8.3 indicates that staff on the whole felt that their skills were reasonably transferable, and that it was not essential for them to develop a career in one company. Junior staff in particular indicated that they were open to career moves elsewhere, whilst senior people tended to feel that they had become too highly paid and too specialised in a particular branch of real-time communications software development to be taken on elsewhere. One member of the Project X team who had led customisations and enhancements in the past felt that he nonetheless had transferable skills of being able to lead a team of programmers, devise tests for a new system and 'get it working'. His skills in communications software and message-switching were much more specific to Telexco.

The overall picture in terms of career structures is then one of a single pool of staff at junior and intermediate levels, with some leaving and the remainder becoming specialised at senior levels in two functional specialisms: the more 'technical' development work, and the more 'product-centred' support and marketing work.

8.2.4 Overall Control And Coordination

The mechanism for overall control of the Technical Department was action planning. The teams engaged on customisations of Systems A and B, and on Project X were controlled relative to targets set in action plans for their projects.

(a) Specifications and Targets For Customisations of Existing Products

For the customisation teams, direct lateral contact between Marketing and the development team concerned was used to draw up targets and specifications. This served to ensure that contracts were entered into with clients at a viable price for what was being supplied and a viable cost in terms of the work required within the Technical Department. These targets were the basis upon which the Department was controlled. The Technical Manager experienced pressure on himself and his department from the General Manager of Telexco in terms of whether or not targets were being met.

Projects were initiated by Marketing, usually in response to an invitation to tender put out by a client organisation. A member of Marketing would pass a preliminary analysis of requirements to the team leader of the System A or B customisation, according to which system was more appropriate for the industry and size of system required. The team leader estimated the amount of time required to develop software for various customised features. There was then negotiation between the Marketing and the team leader, in which trade-offs in features and development cost were made. The aim was to arrive at the combination of functions and price most attractive to the client, at a development costs such that the return was up to the level required by the departmental business plan. The total amount of development effort implied had also to be available within the headcount of the Technical Department provided for by the overall business plan, given the rest of the project load of the team in question.

Development team leaders and the Technical Manager agreed that the difficult part of setting targets was not in estimating development costs themselves, but in rating the relative importance and value added by various features and making trade-offs between with development costs. There would appear to have been considerable uncertainties and complexity of information processing in making the trade-offs between the two. According to

the Technical Manager it was often necessary to 'cross-subsidise' an expensive but highly-valued feature by adding more than the usual profit margin to other features that were simple to develop. There was also sometimes a need to charge lower profit margins on different variants of the same system, depending on the industry being served, because the market in some industries was more competitive than others.

Actually estimating the development costs of each feature in isolation was not seen as a problem, and there was rarely any disagreement or conflict between them as to the viability of targets. According to the Technical Manager, if anything, team leaders had a tendency to underestimate targets. However, all parties felt that, whilst it could be done reliably, estimating was too complex a process to be reduced to rules and procedures. Different types of software required very different amounts of time to develop them, and experience was the best guide.

The resulting overall specification for the system, delivery date, price and target development cost, had to be approved first by the heads of the Marketing and Technical Departments, the general manager of Telexco, and then, with the size of a typical bid of the order of hundreds of thousands of pounds, passed on to Telecorp, a fixed-price contract being drawn up with the client. Typically, a customisation project lasted 6 to 12 months from start to finish, and accounted for 6 to 12 months of staff time.

The use of lateral relations between Technical and Marketing departments can be seen as a way of controlling a task of high complexity involving two functions, without involving managers in the hierarchy above the two functions in actually assessing the interaction of the complexities involved. The need for mutual interaction is demonstrated by a decision made in the late 70s to dispense with a separate systems analysis function which worked with marketing to draw up requirements specifications. It was found that development team leaders had to be involved to such an extent in estimating that it was just as effective for them to work directly with marketing staff.

(b) Action Plans For Customisation Teams

Action plans were drawn up for each customisation team by its leader on the basis of specifications and targets for each project. According to team leaders, this was usually done on a fairly definite basis for six months at a time. Three factors made it possible for plans to be drawn up so far ahead. Projects were fairly large, demand was fairly predictable, and it was also possible to control for unevenness in demand by building up stocks of orders through giving delayed delivery dates. The even flow of work meant that headcounts in each customisation team could be matched to the volume of work on an annual basis with sufficient accuracy for incoming work to be negotiated directly with Marketing and scheduled in self-contained and decentralised action-planning, without continually moving people around or building in slack resources to take account of unexpected peaks.

(c) Specifications, Targets And Plans For Project X

Specifications and plans for **Project X** were still in the process of formulation. Together the Technical Department and Marketing had arrived at a definition of the nature of the product and an overall cost for its development over a number of years, based on an estimate of revenue that could be expected. More detailed specification and planning were at an early stage. Requirements analysis was underway, and would lead to an overall specification and more detailed development cost, time and manpower targets. These would provide the basis for more detailed specifications and targets for sub-sections of the overall system, which would be allocated to a number of teams within Telexco and the collaborating systems houses.

(d) Overall Control Of Maintenance Work

It was not considered possible to plan maintenance work on the basis of precise specifications and targets. According to the Technical Manager, the overall target for the System B support team was 'vague'. They were expected to respond effectively to maintenance calls on installed systems, which occurred when problems were too complex for a field engineer. Its headcount was set in proportion to the number of installed systems. The resources for this were budgeted in the annual business plan, and funded from previous sales revenue.

8.2.5 Control Of Individual Performance

Once project targets had been set, monitoring and control of actual development work became primarily the reponsibility of the team leader who was involved in producing the target. The Technical Manager regarded his team leaders as 'their own business managers'. He regarded it as largely their responsibility to ensure that producers met targets. Team leaders identified the following mechanisms as important in ensuring that they did so.

(a) 'Professional' Skills, Commitment and Targets

The basic mechanism for controlling individual performance on all teams was entrusting producers with the responsibility for meeting project targets, without close formal monitoring of their progress between 'milestone' points. This was based on assumptions that they had the skills and experience necessary to make decisions in negotiating the complexities of their tasks, and commitment to adhere to targets. The next section describes the mechanisms identified by team leaders and the Technical Manager for encouraging producers to adhere to project targets.

The basis upon which this commitment could be expected was the fact that project targets were set by team leaders and had built into them norms as to how hard producers could be expected to work.

On the System A and System B teams, team leaders used their project targets to draw up low-level action plans which allocated blocks of work to their staff. This was usually done twice a year, given that the workload was definite for about six months ahead. Overall project estimates were broken down into estimates for each stage of the project, i.e. design, programming etc. Blocks of work based on project stages, or, for large projects, parts of them, were allocated to individuals, matching their level of experience to the complexity of work involved in a block. The volume of work entailed in a block was of the order of several weeks, if not months.

On both the System B support team and the Project X team, targets were much more vague. On the support team, the team leader simply allocated fault reports. On Project X, each team member was allocated an area of responsibility, with a rough timescale, and left to define by themselves what should be involved.

Skilled professional discretion and commitment to work to project targets and specifications was supplemented by three further mechanisms for controlling individual performance, described below.

(b) Structured Techniques

Producers adhered to structured standards, which can be understood as a form of standardisation of skills, making the quality of performance of tasks predictable.

On the customisation teams, system design was carried out according to the requirements of a version of structured design, which specified ways of breaking the overall design into modules,

using a restricted set of logical components, ensuring clarity of the logical structure, and making it easy to modify one module without affecting the functioning of others. In the case of the System A team, design modifications were not usually carried out strictly according to structured standards. The original design had been performed before the standards were adopted. Lower level design and programming was also to be performed according to structured standards, although this again did not apply to System A modifications, because the code was written in assembly language, which did not contain operators that corresponded to the permitted 'structured' logic elements.

For the Project X team, the software engineering methodolgy was still in the process of being shaped. The general intention expressed by the Technical Manager was that a more rigorous set of standards would be required, to cope with the greater size and complexity of the system. We shall return to this and its implications for the control of work in the final section of this case study.

(c) The Availability of Help

Table 8.4 shows the extent to which producers used colleagues and supervisors for help in resolving problems in their work. It shows no significant differences between teams, but shows that on average producers spent as much time getting help from their team leader as Table 8.5 shows they spent in giving progress reports. Table 8.4 also shows that colleagues were as significant a source of help with technical problems as team leaders.

(d) Milestone Reviews and Monitoring

Table 8.5 indicates the basically low intensity of formal and informal reporting for staff in the Technical Department who completed diary sheets. This supports the picture obtained from team leaders, that formal reporting occurred only at the end of a

Table 8.4: Formal and Informal Reporting Patterns

		Formal Reporting		Informal Reporting	
	N	Av. no. of times per day.	Av. Total time per day (min)	Av. no. of times per day	Av. Total time per day (min)
System A Team Leader	1	0.2	0.2	0.8	7.4
System A And System B Designers	4	0	0	0.4	2.7
System A and System B Programmers	2	0	0	0.2	1.1
System B Maintenance Team	2	0	0	0.6	8.4
Project X Team	2	0.6	22	0.6	10

Table 8.5: The Use of Help from Team leaders and Colleagues

		Help from Team Leaders		Help from colleagues	
	N	Av. no. of times per day.	Av. Total time per day (min)	Av. no. of times per day	Av. Total time per day (min)
System A Team Leader	1	0	0	0.8	6.2
System A and System B Designers	4	0.3	1.4	0.2	1.1
System A and System B Programmers	2	0.2	1.0	0.3	1.7
System B Maintenance Team	2	0	0	1.3	15
Project X Team	2	0.8	15	1.1	29

Table 8.6: Discussing Interfaces with Team Leaders and Colleagues

		With Team Leaders		With Colleagues	
	N	Av. no. of times per day.	Av. Total time per day (min)	Av. no. of times per day	Av. Total time per day (min)
System A Team Leader	1	0	0	0	0
System A and System B Designers	4	0.1	0.5	0.1	0.1
System A and System B Programmers	2	0	0	0.8	8.0
System B Maintenance Team	2	0	0	0.7	4.3
Project X Team	2	0	0	1.3	50

substantial block of several weeks' work, and usually took the form of a formal 'milestone' review of the output of a stage of a project. In between times, team leaders would stay in contact to see how work was progressing, and provided advice, but did not monitor every step.

The 'milestone' dates for most developments were the conclusions of system design, program-level design, coding and system testing. The first two of these were particularly important because they were marked by review meetings. These served not only as time-deadlines but also as a means of checking the quality of the system under development. In the presence of the Technical Manager and the team leader, the designer would 'walk through' the design or modifications to it. This involved describing the functioning of each part of the system and the way it interacted with the whole. The objectives were to establish first that the design was logically sound and performed the functions specified for it without ambiguities which might show up later as operating faults; and second that it adhered to structured standards, so that it would be easy to modify and maintain.

According to team leaders, the skills of staff, the use of structured techniques and the availability of help as needed on a day-to-day basis together meant that reviews were very much a 'back-stop', from the point of view of controlling both quality and productivity. Faults discovered at design reviews were rarely fundamental in nature, and over-runs of cost and time at milestone points were also largely avoided. Reviews were however seen as an effective way of detecting errors.

The Technical Manager regarded the use of structured design standards as vital for design reviews to be possible, because they permitted notation and evaluation of designs against requirement specifications. Prior to the introduction of these techniques, it had only been possible to review the technical content of systems once they had been coded. Code had been read by the team leader or a senior team member. The Technical

manager regarded this as a very time-consuming way of detecting errors. Design reviews had permitted its discontinuation. Errors might remain in the design, or be introduced at the coding stage, but these were usually few and not fundamental in nature, and so could be detected easily and eradicated at the testing stage.

8.2.6 Low-Level Coordination of Work

The primary mechanism for coordinating the work of team members engaged on different portions or stages of a project was specifications produced at the end of each stage, for the succeeding one. This was supplemented by two further mechanisms.

(a) Structured Standards

The observance of structured standards was intended to make designs or code produced by one person comprehensible to another, and thus provided coordination through standardisation of skills. In this respect, structured techniques were reinforced by the use of a particular formal notation for designs, 'Keyword design language' (kdl). This ensured that designs were precisely notated and comprehensible to someone other than the original designer.

(b) Direct Lateral Relations

Table 8.6 indicates that direct contact was also required, either to clarify interface issues between producers involved in developing different parts of the same system, or to clarify the meaning of specifications. The table suggests that these interface issues were more often clarified between producers themselves than by reference to a team leader.

8.2.7 Mechanisms for Adjusting Plans

(a) Vertical Information Collection

Whilst slippage was not seen as a major issue in the Technical Department, one of the functions of the informal monitoring by team leaders was to detect it as soon as possible. To this end, team members also filled in weekly time-sheets indicating the amount of time they had spent on each of their allocated tasks. Team leaders used both sources of information to compile a monthly report to the Technical Manager, detailing performance on all the team's projects relative to targets.

(b) Calling on extra effort

If slippage was detected following a technical problem, team leaders' first preference was to call upon a period of extra effort from the team, without changing any schedules. Staff in junior grades (up to Programmer 1) had to be paid overtime, whilst other staff understood that they were supposed to work extra hours as required. At the time of fieldwork, only three members of the Technical Department were in junior grades. Team leaders felt that the way to motivate people to work extra hours was to 'lead by example'. The team leader too had to put in extra hours or work more intensively, and also perhaps help team members with urgent bits of coding, or set up tests for them. To carry the respect of team members and to motivate extra performance, team leaders had to 'get their hands dirty'. They then felt able to 'lean' on the members of their team, provided they had a clear reason for doing so in terms of meeting a project deadline.

(c) Rare use of slack resources

In addition to presenting a written report to the Technical Manager once a month, and meeting formally once a month to discuss current issues and future plans, each team leader spent about five minutes a day with the Technical Manager discussing any problems and the measures the team leader was adopting in dealing with them. On occasions, if slippage proved not to be recoverable using the resources allocated to the project more could be sanctioned by the Technical Manager. This could either take the form of payment for overtime or the drafting in of ex-Telexco staff with experience on message-switching developments, who had moved to other Telecorp divisions. The latter in particular was rare, occurring for a period of two months perhaps once every three years.

8.3 COMMITMENT TO WORK

The mechanism identified by the Technical Manager and team leaders for encouraging staff to be committed to project targets were the following.

(a) Selection and Socialisation

The underlying mechanism for ensuring that staff could be trusted to observe project targets was selection of staff, on technical and attitudinal criteria. Staff were sought who possessed 'technical' programming skills and a basic commitment to using skills in commercially relevant work. Although he saw a shortage of the 'right kind of people' as a major issue facing the department, the Technical Manager felt that Telexco was on the whole successful in its recruitment. About one in eight new recruits had to be 'eased out' after a year, because they were not considered good bets for adhering to the required norms.

'Easing out' usually took the form of not offering any pay rise, or increase in responsibilty, so that staff left of their own accord.

Even with the 'right' staff, it was however necessary to maintain motivation to work to project targets, and to make sure that people did not become discontented with the work they were doing and begin to look for work elsewhere. One team leader called this 'keeping people happy'. Between them, current team leaders and ex development team leaders now working on Project X identified the following mechanisms for doing so.

(b) Offering training and skills development

Team members had training and skills development objectives set for them following annual appraisals. These were carried out jointly by the team leader and the Technical Manager, according to a format laid down centrally by the Personnel Department of Telecorp, which evaluated performance in terms of technical competence, degree of responsibility assumed, general attitude to work and the company, strengths and weaknesses.

For example, it might be agreed that a programmer who had undertaken low-level design tasks with considerable help was now ready for greater design responsibility. Alternatively, it might be decided that another less experienced programmer should receive formal training in the principles of structured design and be given more low-level design experience before receiving more work with higher level technical responsibility.

(c) Offering pay increases and promotion:

A second outcome of annual staff appraisals was the appraisee being informed of any increase in salary and promotion in terms of job title. Each job title had a broad salary band corresponding to it. Each year central Personnel would lay down

a maximum possible increment for each job title, two thirds of which were awarded as a cost-of-living increase and one third discretionary and performance related. Both team leaders and the Technical manager saw the award of promotion and discretionary increments and the promise of future ones as important motivators.

Formally, increments were controlled by central Personnel, to ensure that pay-compatability was maintained across different Telecorp divisions. In practice, the Technical Manager felt that he and his team leaders had a larger degree of control over the award of increments in that their recommendations to central personnel were generally approved. The main limitation was that the annual divisional business plan allocated a certain fixed sum that could be used for the merit portions of pay-rises within the Technical Department.

Table 8.7 summarises the perceptions of the 16 individuals interviewed as to the criteria used to evaluate their performance for promotion. It indicates the importance for all groups of being able to apply knowledge to solve problems independently of supervision, as well as performing according to set procedures. Management ability and understanding of the application area and of user requirements were also regarded as important by the more senior staff on the customisation teams and on the Project X team. These perceptions are consistent with promotion being used as a mechanism to encourage and reward willingness and ability to take responsibility for decision-making in complex tasks, whilst at the same time adhering to structured standards. The assessment system encouraged individuals to internalise norms of quality and effort.

(d) Setting tight targets.

Team leaders saw setting challenging targets for individuals as a means for encouraging motivation to work, on the assumption that team members preferred to work under a certain level of pressure.

Table 8.7: Perceived importance of various criteria for promotion.

Mean scores for importance on a scale of 1 to 9, broken down according to position in the division of labour and overall.

	System A team leader	Custom-isation Designers	Custom-isation Prog/ Designers	Maint	Proj. X (All Designers)	**Overall**
N	1	4	2	3	6	16
Problem Solving ability:	7.0	7.3	5.5	5.7	7.8	6.9 (s.d.=1.8)
Performing according to procedures:	4.0	5.5	4.0	5.0	6.5	5.5 (s.d.=1.9)
Good technical knowledge:	7.0	7.0	9.0	5.3	8.2	7.4 (s.d.=1.7)
Understanding of application area:	7.0	7.0	1.5	7.3	7.3	6.5 ** (s.d.=2.4)
Management ability:	6.0	6.5	1.5	8.0	6.8	6.3 ** (s.d.=2.2)

Team leaders also saw setting tight targets for the team as a whole as a way of encouraging team members to identify with the team goals. As one put it: "People work better when there's a lot going on." At the same time team leaders recognised that a

balance had to be struck between challenging people and over-driving them so that they became demotivated because demands felt hopelessly unrealistic.

(e) Leading by example.

Team leaders felt that it was necessary to establish productivity norms through making their own levels of effort in direct production activity visible to team members. This was particularly the case if temporary increases in productivity norms were to be obtained on a willing basis.

(f) Allowing team members to 'see a project through'

The Technical manager and team leaders all mentioned the importance of involving a team member in as many stages of a development project as possible, in order to increase his or her identification with the project and motivate their productivity and application to solving problems. Involvement with installation was particularly desirable, because this permitted contact with users on the client site.

(f) Giving team members a variety of tasks simultaneously

The two current team leaders and two ex-development team leaders on the Project X team all stressed the importance of allocating team members with work for as far ahead as possible. This was partly so that they were well informed as to what they would be doing in a few weeks or months time. If they were currently engaged on a portion of development that they did not find rewarding, such as documentation, this could re-assure them that

more interesting work was in store.

More generally, it allowed team members to have control over how they divided up their time between different project commitments, in that they could rotate their attention from one project to another in so far as deadlines permitted. In particular, if they were prevented from working on one project because they were waiting for a portion of design that had not been completed or for the result of a test run, they could begin coding or design work on the next project, rather than suffer enforced idleness. The net result was that team members would work more contentedly and more productively. One team leader felt that programmers worked much less productively when they had only one project to work on and did not know what they would be doing six weeks hence.

8.4 THE NATURE OF TENSIONS AND CONFLICTS

8.4.1 Issues for Managers

(a) Maintaining Motivation

According to the Technical Manager, the mechanisms described in the previous section were generally successful in ensuring that people adhered to norms of quality and productivity implicit in project targets, as well as ensuring that people did not leave. He felt that Telexco was more successful than most systems houses in getting people to identify with the system they were working on and with the company, so that people stayed with the company for 8 years on average, as opposed to the 2 or 3 characteristic of the industry in general. There were however two main problems.

First, the lifespan of a product - at around five years - was considerably longer than the amount of time an individual generally wanted to spend working on one system - two or three

years at most. It was however desirable to keep some people working on a product right through its lifecycle, performing customisations and support, so that the expertise built up in the early stages was still available later. This was perhaps the biggest issue in managing 'the people side'. The Technical Manager felt that the solutions so far adopted were adequate, but had not eradicated discontent. These solutions were to persuade senior people to stay on an obsolescent product for a few months or years more by promising that they would move from it on to the most state-of-the-art system, and, by offering them opportunities to perform installation work in exotic places with generous expenses.

Second, the Technical Manager felt that the current system of job titles and salary bands was too confusing and unrelated to actual levels of job-responsibility to give people a clear identity and perception of their status within the organisation. He wanted a system of job titles which made it clear that there was a definite series of steps in terms of levels of responsibility, so that people could have a clearer sense of achievement on being promoted to more responsible work. He also wanted salary to be linked more clearly to levels of responsibility, rather than the current scheme where the salary-bands for job titles were overlapping.

(b) 'Finding the Right People'

We have already discussed the issues for managers in terms of finding 'the right people' to recruit at graduate level. The technical manager was not averse to recruiting people into more senior positions from outside, but in practice it was extremely difficult to attract the sort of skills and experience required by Telexco's highly specialised existing product range. Since there was also wastage of junior people, finding people to fill slots at all levels was a basic issue for this form of organisation which relied so heavily on the characteristics of its staff.

(c) Keeping in touch with the Product Market

The most major issue named by the Technical Manager did not in fact concern the control of work. It was keeping informed as to what the product market would be demanding in the years ahead, so that plans could be laid. The most pressing issue for current activities was getting 'accurate and reliable information', in

terms of technological advances and user preferences, as to what was going to be required of the Project X product in the office automation market.

It is significant that the levels of productivity of software producers and the standards to which they worked did not feature amongst the Technical Manager's list of issues. This suggests that the use of a 'professional' form of labour control, with adherence to targets ensured by socialisation and encouraging commitment to work through the offer of increasing responsibility with promotion, was associated with an absence of explicit conflict between managers and producers over levels of individual performance.

8.4.2 Issues For Producers

This sub-section presents findings from structured interviews with 16 individual software producers within the Technical Department. In the tables that follow, the sample is broken down into five groups, on the basis of the type of responsibility: a customisation team leader, customisation design staff, customisation low-level design and programming staff, staff supplying support for existing products, and those engaged on the early stages of Project X.

Table 8.8 indicates the considerable extent to which the system of labour control based on giving people responsibility and autonomy for achieving targets meant that software producers within Telexco felt their work goals were being achieved. There was basic harmony between individual and organisational goals, rather than explicit conflict. On average over the whole sample, producers felt that their work was achieving their most important goals and that their job measured up to what they wanted out of a job to a greater than 'satisfactory' extent.

This was not however true of all the 16 as individuals. Four out of the six Project X team members, one designer on the System B customisation team and one designer on the System A team all answered 'not very well' on both counts. We now examine various aspects of software producers' experiences of their jobs, as a basis for identifying the conflicts and tensions underlying these less-than-satisfactory levels of goal achievement.

(a) Utilisation of skills

Table 8.9 indicates that individuals found that their skills were utilised to average levels in terms of time demands. This is further evidence of a lack of tension over productivity levels. The fact that four of the six Project X team members experienced below average time demands even suggests they may have been wanting to do more. At least one felt that he was really marking time, preparing the development methodology whilst design decisions were being made elsewhere.

Table 8.9 also indicates technical demands that were on the whole 'above average'. However, those responsible for system design on the customisation teams felt themselves to be working under significantly different, slightly below average technical demands. Similarly, whilst on the whole the sample felt that they were given the opportunity to 'show what they could do' two thirds of the time, customisation designers reported this state of affairs on average for less than half of their time, which was

Table 8.8: Views on overall goal satisfaction

Mean scores for satisfaction on a scale of 1 to 7 (4 = 'about right'), broken down according to position in the division of labour.

	System A team leader	Custom-isation Designers	Custom-isation Prog/ Designers	Maint	Proj. X (All Designers)	Overall
N	1	4	2	3	6	16
Work achieving most important personal goals:	5.0	4.5	4.5	5.0	3.8	4.4
Job measures up to what you want:	5.0	4.8	5.5	4.0	4.2	4.5

Table 8.9: Experience of utilisation of skills

	System A team leader	Custom-isation Designers	Custom-isation Prog/ Designers	Maint	Proj. X (All Designers)	Overall
N	1	4	2	3	6	16
Time demands: (Scale: 1='ext. light' to 7='ext heavy')	6.0	4.0	5.5	6.3	4.0	4.8 (s.d.=1.5)
Technical demands: (Scale: 1='ext. light' to 7='ext heavy')	4.0	3.5	5.5	6.3	6.0	5.3 (s.d.=1.5)
% time below technical capacity:	60	48	10	57	20	35 ** (s.d.=2.1)
% time trivial or routine:	20	35	25	37	13	25 (s.d.=1.6)
% time show what you can do:	90	43	65	80	70	66

Table 8.10: Experience of autonomy in carrying out tasks:

	System A team leader	Custom-isation Designers	Custom-isation Prog/ Designers	Maint	Proj. X (All Designers)	Overall
N	1	4	2	3	6	16
% time using own approach:	80	53	80	93	70	72 (s.d.=2.3)
% time making all important decisions:	80	45	75	57	40	52 (s.d.=2.9)
% time specs already agreed by others:	40	28	20	17	12	19 (s.d.=1.6)
% time every step has to approved:	0	43	10	30	20	25 (s.d.=2.5)
% time given detailed instructions:	10	23	10	7	10	13 (s.d.=0.9)

Table 8.11: Impact and visibility of contributions

	System A team leader	Custom-isation Designers	Custom-isation Prog/ Designers	Maint	Proj. X (All Designers)	Overall
N	1	4	2	3	6	16
% time involv. in business decisions:	50	18	0	23	12	16 (s.d.=1.2)
% time important contrib:	70	65	75	63	53	62 (s.d.=3.3)
% time contr. to business success:	60	55	30	63	32	45 (s.d.=2.8)
% time work of special interest to management:	20	28	0	30	33	28 (s.d.=2.1)
% time higher mgt doesn't know what ind is doing:	50	45	20	30	47	42 (s.d.=4.0)
freq discuss with higher mgt:						
Daily						0
Weekly		3				3
Monthly	1			1	4	6
Six monthly		1		2		3
Yearly					1	1
Never			2		1	3

Table 8.12: View on Salary

	System A team leader	Custom-isation Designers	Custom-isation Prog/ Designers	Maint	Proj. X (All Designers)	**Overall**
N	1	4	2	3	6	16
View on salary, relative to comparable firms: (Scale: 1='much lower' to 7='much higher')	3.0	4.8	1.0	4.3	4.6	4.2 * (s.d.=1.3)
View on salary plus perks, relative to comparable firms: (Scale: 1='much lower' to 7='much higher')	3.0	4.8	1.0	4.3	4.6	4.2 * (s.d.=1.3)

Table 8.13: Satisfaction with Promotion Prospects

Mean scores for satisfaction on a scale of 1 to 7 (4= 'about right') broken down according to position in the division of labour, and overall.

	System A team leader	Custom-isation Designers	Custom-isation Prog/ Designers	Maint	Proj. X (All Designers)	**Overall**
N	1	4	2	3	6	16
View on probable timing of next promotion:	4.0	4.0	5.5	4.3	4.3	4.4 (s.d.=0.7)
View on probable size of next promotion:	4.0	4.0	3.5	3.7	4.0	3.8 (s.d.=0.4)

Table 8.14: Attitudes to Learning Advanced Software Engineering

	System A team leader	Custom-isation Designers	Custom-isation Prog/ Designers	Maint	Proj. X (All Designers)	**Overall**
N	1	4	2	3	6	16
View on learning new techniques, on a scale of 1 to 7: (1= 'very opposed', 4= 'indifferent', 7= 'very keen')	7.0	5.5	5.5	5.0	6.3	5.8 (s.d.=1.0)

a significantly different level. In fact, none of the customisation team designers interviewed felt they had a chance to show what they could do for more than half of their working time. They also felt that they were working below their technical capacity for a significantly higher percentage of the time than programming staff and the staff of Project X. The staff maintaining System B showed both high skill utilisation and a high proportionn of their time for which they were working below their technical capacity.

These data imply that experienced staff performing design work in customising established products felt that they were being insufficiently challenged technically. A likely explanation for these feelings lies in the extent to which the modification of existing designs to meet customer needs was not very taxing for an experienced designer. This was partly because it demanded rigid adherence to product design and coding standards that had already been established, and partly because the personnel involved had generally already been working on the product for several years. Customisation work also appears to have involved some routine work, which was shared out amongst the team. According to one designer:

> "There's a lot of routine work, making up discs for customers, putting patches on discs. We need technicians, but there aren't any."

Tensions arose, then, for the more senior members of customisation teams, stemming from the need of the company to have experienced and trusted staff working on established products, who could perform modifications that were complex, but fairly routine.

A likely explanation for the combination of high technical demands and feelings of misutilisation reported by the System B maintenance team would appear to lie in the proportion of maintenance work that consisted of fairly routine coding, testing, and 're-starting' work, alongside highly intricate and

intuitive fault diagnosis.

(b) Autonomy and influence over the approach to work

Table 8.10 presents data collected on the extent to which software producers possessed autonomy in the execution of their tasks. On average they were able to 'use their own approach for over 70% of the time, with no significant differences between job titles or levels of seniority. The average amount of time during which they felt they were making all the important decisions regarding work was rather lower at around 50%. All the questions designed to function as indices of the extent to which software producers felt they were directly controlled in matters of technique and approach to their work produced correspondingly low ratings. On average producers felt that specifications were already agreed by others or that every step they took had to be approved by someone else for around 20% of the time, and they were given detailed instructions for around 10% of the time.

However, an analysis of the variation in responses at the individual level indicates that there were in fact some tensions present. Four of the six project X staff felt that, although they on the whole were using their own approach to work, they were 'making the most important decisions' for only 20% of the time. Three of the four designers on the customisation teams were the only other members of the sample who felt that they were 'making the most important decisions' for less than half the time. Given that Chapter 7 shows that software producers in general attached importance to this aspect of work, these results indicate possible tensions.

The explanation for these designers' feelings of lack of influence over their work lies along the same lines as that of their perceptions of low skill utilisation. The redesign work they were engaged on was highly constrained by long-established product standards. Techniques and procedures were to a great extent dictated by the standards to which the basic systems had

been designed and coded. The explanation as to why staff engaged on programming in the same teams felt generally less constrained on the approach adopted was probably that they did not have any expectation of setting standards themselves, but were used to performing low level design and coding according to standards set for them by system designers.

The reason why some staff on Project X also felt that they were not controlling the most important decisions in their area of work lies elsewhere. First, the two members of the team engaged on requirements analysis appear to have been dissatisfied because they were used to more influential roles in discussing system requirements with customers than the early stages of this large project permitted. This was in spite of high utilisation and autonomy in actually carrying out their work. This tension seems to have stemmed from the sheer size of the project, and the lack of influence one individual could feel over the stage they were engaged on. Those involved in devising development methodologies and tools appear to have experienced something very similar. Two members of the team were worried that the important design decisions would be made by someone outside Telexco altogether in one of the other collaborating systems houses. One Telexco Project X team member complained that he was having a great deal of difficulty in getting information out of another division as to what they were actually doing. The root of tensions in this case would then appear to be a new collaborative arrangement in which it was not clear if the organisation to which software producers belonged had sanctions to control the decisions that gave them a feeling of control over their work.

(c) Experiences of Structured Techniques

Data collected on experiences of structured design and programming indicated that neither technique was seen as leading to unacceptable restriction of autonomy at work or a major source of tension. It was however clear that most producers had come to some accomodation between their desire for autonomy and the

benefits they could see associated with these techniques.

All those interviewed, apart from three programmers who had no experience of structured design, considered that the structured design techniques and kdl currently in use on Systems A and B were of great importance for producing better quality code that was easier to read and maintain. This approach to design was also felt to allow mistakes to be detected at the design stage rather than after coding. Further, one individual actually attached considerable value to the extent to which structured design and design walk-throughs increased visibility of progress during the design stages of development. He did not see this as a threat in that it might make producers' output more visible to management, giving the opportunity for more frequent monitoring and more direct pressure. As one put it, 'That just isn't what good management is about'. Rather , he saw visibility of what had and what had not been achieved as an advantage in negotiating budget and timescale increases with management, when unexpected difficulties were encountered. All those interviewed were convinced of the advantages of structured programming from the point of view of maintainability and clarity of the code produced. In this sense, structured techniques were seen not as an infringement of professional autonomy, but as a tool for producing good-quality work.

At the same time, nearly all respondents expressed some reservation about the experience of actually using structured design and programming. 13 out of the 16 felt that one or other of the techniques could be too formalised or restrictive, and if followed too slavishly, would not lead to the best result under all circumstances. Structured programming in particular was seen as leading to unnecessarily tedious and repetitive as opposed to 'clever' programming, which could be more economical on processor time and memory space. These respondents felt that structured techniques did not eliminate the need for skill, in knowing how to use the techniques and when to depart from them. They felt that structured techniques should not be absolutely binding:

"Bad structured design isn't worth having. In the wrong hands it can produce a load of rubbish." (Designer)

"The good thing about structured programming is that it gets all the 'sheep' together, and makes them useful. However, it excludes the 'geniuses'; you should allow the 'geniuses' to go off and do something else, otherwise systems have less flair about them." (Designer)

"If you stick to it too rigidly, structured programming is like a factory conveyor belt. You need the freedom to depart from it. It's a good basis from which to start." (Programmer)

Two programmers also felt that structured programming was a problem if not everyone followed it, which 'created messes'. Two designers felt that, although structured programming could be tedious, it was essential to have some form of standardisation of coding. One designer felt that structured design in fact removed decision-making from the realm of programming, but that it was necessary none-the-less, because it allowed designs to be reviewed before programming was begun.

Overall, then, structured techniques appear to have been valued as a means of producing more reliable, clearer and more easily maintainable code, although there were clearly issues concerning the rigidity with which structured rules were applied, and structured design removing discretion from the realm of programming. However, the current level of 'structuring' within the Technical Department was considered by most respondents considered to be high, and nearly all seemed to have come to an accommodation with it. They balanced their desire to use discretion and judgement on the one hand, and to do good work on the other. This accommodation depended on their feeling that the rules were not absolutely binding, but guidelines. Only one designer, one of the oldest, appeared to be seriously advocating

the position that uncodified skills gained from experience were a superior alternative to the use of structured techniques.

(d) Visibility of Individual Contributions and Impact on Business Aspects

Responses to questions on the extent to which software producers felt their contributions were visible within the organisation and that they exerted influence over business decisions and business success are summarised in Table 8.11. Of those interviewed, the only person who considered that he had a significant involvement in business decisions was the System B team leader. At the same time, there was a fairly uniform consensus amongst all types of producer that they were making an important individual contribution to the Division for around 60% of the time and they were contributing to business success for about half the time. Given the importance for software producers of making a contribution and their indifference to influencing business decisions identified in Chapter 7, these results do not seem to indicate any significant tensions. One designer, for example, was clear that the sphere in which he wished to have influence and make a contribution was that of the task of his own team, rather than that of the Division as a whole:

> "I'm not interested in the financial position of the company, but I do see my team's budget as a challenge. It's things at the level of my team that I identify with."

(e) Views on pay

Table 8.12 shows individuals' views on salaries. They show that pay was almost uniformly judged to be comparable to that for similar jobs within similar organisations in the area, with the exception of junior customisation staff. None of these however mentioned pay as being an issue for them. Junior staff were

apparently prepared to settle for less than the going rate for their skills because of longer term prospects on offer.

(f) Opportunities for Career Advancement and Skill Development

Table 8.13 shows uniform satisfaction with promotion prospects in the short term. There were however some significant issues for one or two individuals in terms of the extent to which, in the longer term, a career within Telexco would permit them to develop skills along purely technical lines. Whilst a clash between 'commercial' and 'technical' orientations did not seem to emerge in terms of how producers experienced current work and the way specifications were set for them, it did appear to be an issue when some of the sample thought about where they wanted to end up in their careers.

We have already seen, in Table 8.2, that those interviewed saw their careers as leading in a number of different directions over the next five years. On the one hand, there were two individuals, one of Project X and another currently a customisation designer, who wanted to go into the technical aspects of marketing and sales. They wanted to be associated with the development and commercial success of particular products, and to have contact with customers. At the other extreme there were two staff, again, one on Project X and one a customisation designer, who were quite clear that their interest was in solving technical problems using the most advanced techniques, developing one system and then moving on to the next. Neither was very interested in dealing with customers or in 'working with people'. Both were also quite uninterested in moving into technical management positions, but wanted to stay as senior designers. The remainder appear to have had views somewhere in between these poles, although the fact that most people saw themselves moving into technical management of software production rather than marketing or sales support suggests that they were more 'technically' than 'product' oriented.

This bias towards a 'technical' orientation does not appear to have been a problem for most of those interviewed. Indeed, one producer commented that the organisation as a whole appeared to be much better at the 'technical' job of producing a system that worked at the outset, than in managing a product over time. For example, instead of up-dating all customers with new releases in order to simplify maintenance, the Division left most customers with the version they had brought, which meant that a whole range of different versions of the same product had to be supported. The organisation clearly catered for people who were more 'technically' than 'product' oriented to a considerable extent, particularly in that it was possible to keep progressing in salary terms as a senior or principal consultant, without going into management. However, the two staff interviewed with very definite 'technical' orientations were both dissatisfied with their prospects for skill development. One was a senior designer, recently transfered from System A to Project X, in his late thirties, who complained:

> "I prefer to work in a technical environment. The company has only trained me for one product, and hasn't kept me up with the state-of-the-art in system design. I've been kept on an old generation of product, and there isn't much technical input on (Project X) now. There's just not enough technology push here."

Both these staff wanted training in more advanced design techniques. The single support team member who felt that his goals were not being achieved 'satisfactorily (see Table 8.8) also attributed this to his view that it was vital to be kept up to date with advanced software engineering techniques, and that he was not being given sufficient training to do so. He too had a definite notion of his longer term interests and doubted the Company was taking care of them. There is then evidence of the tension described by management, of staff resenting being kept with established products, because of the limitations posed for skill development.

8.4.3 Overview: The Nature of Tensions

We have seen that software producers were on the whole fulfilled by the sort of work offered to them, and that managers were satisfied with the standards of individual performance stemming from a form of labour control based on 'professional' responsibility. There do not appear to have been conflicts over levels of productivity or the nature of specifications and targets. The use of structured techniques as a form of quality assurance and coordination appears to have been by-and-large accepted, on the basis that they were guidelines to be departed from when judgement indicated.

The main tensions within this system of 'professional' labour control appear in fact to have stemmed from the need of the Division to keep experienced staff working on existing products, whilst new ones were being developed. This led in some cases to feelings that skills were not being fully utilised in the present. In other cases, producers felt that their skills were not being developed over time sufficiently for them to keep up with the state-of-the-art in software production techniques. This was necessary for preserving their position in the external labour market, as well as their job satisfaction. There was a clash between this sort of strongly 'technical' orientation and managerial goals concerned with maintaining revenue from existing products. It is significant that there were considerably stronger views expressed over fears of not being in a position to learn new structured techniques than over feeling restricted by use of current ones.

A further area of tension arose from the large size and newly-collaborative nature of Project X, which stemmed from the need of Telexco to develop innovative products in markets that were being radically transformed by changes in product technology. The size of Project X and uncertainties about relationships with collaborating divisions appear to have led to feelings of lack of influence on the part of several Telexco staff engaged on

requirements analysis and planning for the project. The issue would appear to be one of the ability of an organisation employing professional technical workers to establish control over its sphere of activity, so that the 'professionals' within it can, in turn, feel in control of their work.

5. INITIATIVES AND POSSIBLE OUTCOMES

The main proposals for change within the Technical Department stemmed from the need perceived by the Manager to devise mechanisms of control and coordination adequate for Project X. The main focus here was the work of the two members of the project team concerned with developing software methodogies, tools and procedures. A more rigorous version of the structured design already in use was proposed, appropriate to the more sophisticated high-level language which it had been decided to use, and the greater size and complexity of the project. This decision stemmed from concerns with the quality, maintainability and compatibility of the different sub-systems. Rigorous 'software engineering' was called for by the increasing competition in these aspects of quality in the field of office and communications software. It was also made necessary by the sheer size of the new product, and the need to ensure integrity. There does not appear to have been any association of the new techniques with disrupting high trust dynamics or introducing more intensive forms of monitoring of team members. Rather these techniques were seen as a way of perpetuating autonomy, whilst ensuring that standards were kept to, and the quality of a larger and more complex system assured.

Table 8.14 summarises how individuals within the Technical Department were approaching the introduction of more advanced software development techniques, such as those proposed for Project X. It suggests that there would not be any resistance to new techniques. Indeed, there was uniform keeness to learn them,

consistent with the basic positive orientation towards structured techniques already in use. We have already discussed the unrest caused when staff had particularly strong and unsatisfied desires to learn the new techniques.

There was however evidence of some unease as to possible consequences of more rigorous techniques on the Project X team, reflecting the ambivalence present towards current techniques. The team member engaged in defining the overall development methodology was extremely keen to learn new techniques, but at the same time concerned that important decisions as to the methodology to be used would be made outside Telexco by one of the other software houses involved, where more advanced software engineering had been in use for some time. He and his colleagues might then find themselves working to a set of rules and standards imposed from outside.

The issue here can be understood as one of a possible centralisation of design decisions being associated with more rigorous design methodologies. Just as the existing techniques were felt to have removed some decisions from the realm of programming to that of design, there would appear to be a possibility that with the new techniques in a large project there could be set up a hierarchy of design stages, with 'lower' ones constrained by the 'upper' ones. The fact that some designers were scared of losing autonomy in the initial development process might suggest that even greater tensions could occur in the design stages of subsequent customisations. Moreover, it is possible that the rigour of the design process could have serious effects in terms of removing discretion and decision-making from lower level design and coding occurring later in the initial development.

It was clear that the thrust for new techniques came on the one hand from a managerial concern with coordination and quality, and on the other from the desires of some designers to learn the latest techniques. There was then no intention that centralisation of design decisions should be associated with a

shift in the nature of labour control away from giving people responsibility and discretion, or to the introduction of 'technician' positions for doing routine coding. One designer felt that advanced structured design made coding very routine, but that this meant that it should be done by computer, using an automatic 'program generator' piece of software. On the whole, though, it would appear likely that, as long as programming remained challenging enough to satisfy the technical appetites of graduate recruits, they would be employed on it, gradually taking on more design tasks, as a way of cultivating designers.

CHAPTER 9

EXCO

SUMMARY: EXCO

1. BUSINESS STRATEGY AND TASK ENVIRONMENT

Market: Developing large-scale telecoms. systems for other Telecorp divsions.
Business Strategy: Differentiation through innovation.
Task Environment: Complex, technically and commercially. Unpredictable technically and commercially. Production smoothing problem due to large size of projects.

2. ORGANISATION, JOB STRUCTURES, CONTROL AND CO-ORDINATION

Organisation: Staff with software background in 'marketing' and 'planning' depts., as well as 'production'. Production dept. made up of several product-based units. Each project unit has design team & several for implementation.
Career Structures: Single career structure with no vertical segmentation, promotion from implementation to design. Mostly computer science graduates.
Overall Control & Co-ordination: Action planning for each project, on basis of targets and estimates drawn up by designers in consultation with implementation team leaders. Targets set with less precision during wind-down phase.
Labour Control: Achieved through 'professional' skills and commitment in the context of targets and specifications, backed up by informal monitoring by team leaders, milestone reviews. Standardisation of skills through structured programming assures quality, as does availability of help from team leaders/colleagues.
Low level co-ordination: Structured standards and direct contact.
Adjustments to plans: Information concerning technical problems collected 'vertically' through informal monitoring: plans adjusted by calling on slack resources.

3. MECHANISMS FOR PRODUCING COMMITMENT TO WORK

Regular contact between producers and team leaders. Offering skill development and training. Offering raises and promotion. Ensuring team is a viable 'social' unit. Keeping producers informed of work in the pipeline.

4. ISSUES AND TENSIONS

For Managers: Keeping producers from leaving a project. Keeping producers informed of future work. Imprecisions in specifications from users. Technical complexity of development process. Readjusting plans to take account of complexities.
For Producers: Lack of challenge and influence during project 'wind-down'. Restricted nature of 'implementation' work. Lack of contact between designers and implementors.

5. INITIATIVES FOR CHANGE AND LIKELY OUTCOMES

Initiatives: More emphasis on training and promotion. Moving into 'external' market. Smaller, shorter projects. Seperate estimates for design and implementation. 'Troubleshooter' senior 'designer' positions. Matrix structure for joint hardware/software projects.
Outcomes: Continuation of 'professional' labour control. Some removal of discretion from junior positions.

9.1 BUSINESS STRATEGY AND TASK ENVIRONMENT

Exco came into being in 1977, when the Telecorp Board decided to establish the large Technical Department of one of its product divisions as a separate organisation. It was already undertaking developments of microelectronic-based telecommunications equipment for other Telecorp divisions and its overheads were considered an increasingly inappropriate burden for the marketing and sales capacity of one division. It was decided to use its capacity in a still wider role developing the central switching elements of communications products in divisions of Telecorp and its US parent organisation, Amercorp, throughout Europe. These elements consisted of mostly standard microprocessor hardware, programmed through software.

Meanwhile, the original Telecorp division retained a smaller software development capacity, and became Telexco, the subject of the previous case study.

Expectations of Resource Providers

Exco became a quasi-independent systems house serving the 'internal' corporate market. It was to undertake contracts developing communications products for product divisions, essentially on a market basis. It had no corporate funding even of overheads, and was treated as a profit centre.

Although Exco was marketing to other product divisions in the same corporation, its overall performance criteria were set through the mechanism normally used for a product division in a multi-divisional firm. Performance was judged by the Board in terms of annual financial targets, rather than more detailed specification of activities to be undertaken and then monitoring the way they were carried out. This can be understood as a means of controlling efficiently operations which involved a great deal of complex information processing, between Exco and 'client' divisions, in the specification and negotiation of development

contracts. The nature of development activity was negotiated through direct lateral contact in a market framework, rather than via a coordination mechanism involving the central corporate hierarchy.

Business Strategy

Exco was a supplier of design and development services to product divisions that were involved in marketing innovative types of telecommunications equipment - public and private exchanges, and telex machines. They were competing in relatively new markets, where both product technology and the nature of demand for particular products were changing rapidly. Such conditions demanded that successful firms adopt basic competitive strategies of the 'Prospector' or 'Innovative Differentiation' type, in order to carve out a market share. There were no established product configurations for which cost leadership could be established. This meant that Exco too was basically developing new kinds of products, rather than cutting costs on existing ones.

Exco's product strategy was built around the distinctive competence of its staff, in designing digital telecommunications switching systems. Initially, it had aimed to win whatever contracts it could from corporate divisions. At first it was totally concerned with one large-scale development, PRIVEX, a private branch telephone exchange (PBX), designed to serve 200 to 4000 lines. By the time of fieldwork, it was developing a smaller PBX (100 to 300 lines) for another Telecorp division, a teletex system for a West German Amercorp division, and a compiler for the Amercorp digital public exchange system. Further it had a contract for a sub-system of the Telecorp contribution to the U.K. digital public telecommunications exchange system, and was involved in the early stages of development of the Telecorp integrated office automation product.

In terms of the size of development project characteristic of this market, typical development projects were scheduled to last for several years, and involved at least fifty person-years of effort. Senior Exco management was in the process of making decisions to focus Exco's product strategy to a greater extent, particularly with respect to the size of developments undertaken. We shall return to this in the last section of this case study.

Task Environment

The tasks involved in undertaking large-scale telecommunications design and development projects for product divisions were as follows. Exco had to tender for contracts, draw up specifications with the client, and develop software to meet them. The type of product it was involved in developing was generally a standard unit, sold repeatedly, involving software running on standard micro-electronic hardware. When a digital private exchange was installed, for example, only a minimal level of software configuration was required, involving loading data on the number of lines to be used and the numbers to be used to identify them. This was much less complex than customisation. As well as developing such products, Exco staff had some direct involvement with users, in loading configuration data in software form on to a product for a particular user once a sale had been negotiated.

Exco projects were not however totally 'one-off.' Exco was responsible for periodic up-date releases of products commissioned by customer divisions, as well as responding to the more complex problems encountered by customer divisions in maintaining systems once they had been installed. Initial fault finding and simple maintenance were responsibilities of the customer division.

Other than in these areas, Exco's sphere of activity did not include contact with user organisations. Marketing, selling and installation of products were all carried out by its customer

divisions. Whilst precise data were not made available, it was clear that the vast majority of activity within Exco was concerned with development work on new products or enhancements of existing ones, as opposed to configuraton work or maintenance. Of this development activity, the majority appears to have concerned software.

The overall business strategy of attracting and fulfilling large design and development contracts for innovative telecommunications switching systems of this type meant that Exco had a task environment that was highly complex technically. The development of switching systems poses some of the most intricate problems in software production, in designing the sequencing of operations to be performed by the system. There was also considerable 'commercial' complexity in understanding what potential clients wanted.

The task environment involved great unpredictability, first of all in terms of the technical problems encountered, since each system developed was essentially new and innovative. There was also great commercial unpredictability. The context of rapid development in telecommunications product technology also meant that the nature and volume of client demand for systems developments within Telecorp and Amercorp were subject to rapid changes. The flow of work could be guaranteed for months or years ahead, however, once a large project had been won. The large size of projects, unpredictability of demand, and the need to have staff available for new projects meant in effect that Exco had a 'production smoothing' problem, in balancing its resources to meet demand.

9.2 ORGANISATION, CAREER STRUCTURES, CONTROL AND COORDINATION

9.2.1 Organisation

Figure 9.1 shows the structure of Exco. The bulk of Exco personnel was concentrated in two main functional departments where actual system development occurred, the Engineering Development Centre (EDC) and Software Development Centre (SDC), with the latter by far the larger.

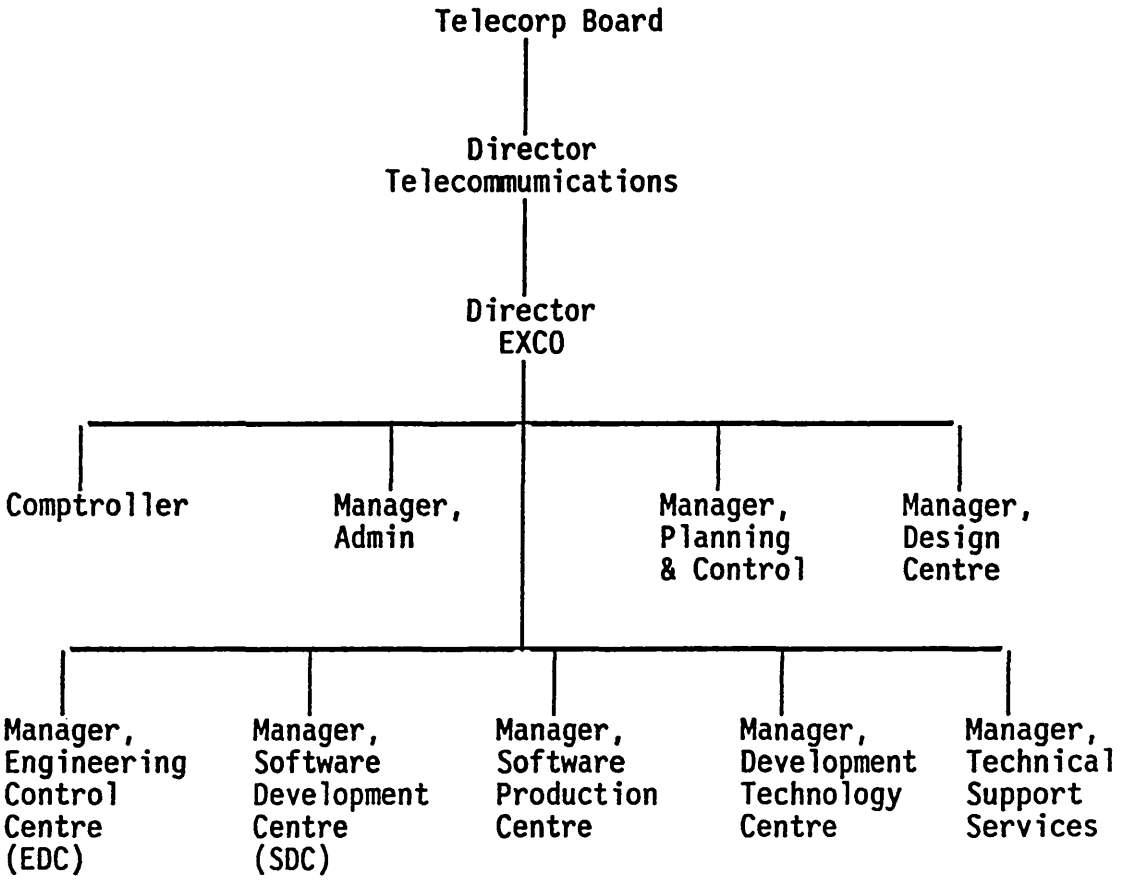
The EDC was in fact concerned with developing only some of the microelectronic hardware elements for Exco products. Most of the hardware, such as terminals, casings and circuit-board mountings were designed, acquired or manufactured and assembled by customer divisions. The EDC largely dealt with the design of printed circuit boards to carry standard microelectronic hardware for the actual switching systems at the heart of telecommunications products. Software for these systems was written within the SDC. Once a system was developed, on the market and in use, the SDC also retained a responsibility for providing specialist backup to the customer division's field service engineers. The SDC would deal with maintenance calls involving complex software faults.

As will be described in the final section of this case study, this functional separation of hardware and software design was becoming increasingly problematic.

The remaining seven departments were also functionally-based and provided specialist services to the SDC and EDC. The Comptroller's department handled accounting, whilst Administration was responsible for organising recruitment and personnel reviews as well as general administration.

The Design Centre consisted of a small group of experienced software and hardware engineers who liaised with potential customer divisions, investigating possibilities of contracts for development work on new products. Such 'marketing' was also a

Figure 9.1: Organisation of Exco



major sphere of activity for the Director of Exco. The Design Centre produced outline designs for new systems for the purposes of submitting tenders and negotiating contracts. The separation of this sales and marketing function from the SDC can be seen as a response to the different types of complexity involved in finding out about possible contracts, and keeping up-to-date with the state of telecommunications products and the sort of features that might be required in them. In addition, once detailed specifications for new systems or enhancements to existing ones had been drawn up by the SDC or EDC the Design Centre was responsible for 'auditing' them. It oversaw the quality and integrity of designs.

Planning and Control also contained people with a software background. Its task was to keep formal records of development project schedules, to record progress month by month and to supply customer divisions with progress reports. This department was supposed to manage the interface with the client during the course of development projects. The Development Technology Centre consisted of a handful of specialist software engineers who developed software tools, such as program editors, documentation aids, and database systems to be used by software engineers within the SDC.

The two remaining departments - Technical Support Services and the Software Production Centre - provided technical services to Exco customers complementary to the work of the SDC and EDC. They also contained staff with software backgrounds. Technical Support Services provided back-up support to customer division field engineers on maintenance problems with Exco-designed hardware. The Software Production Centre held current releases of the systems that had been developed by the SDC. As sales were made by Exco's customer divisions, the SPC would produce a disk containing software instructions as to the details of how the system was to be configured for the particular client. This was sent to the customer division and loaded into the system prior to shipment to the user.

We turn now to a consideration of the organisation and divisions of labour within the SDC, as a basis for understanding the nature of the career structures for software producers.

9.2.2 Divisions of Labour Within the Software Development Centre

In response to the complexity and unpredictability of its task environment, the SDC had a form of organisation with many elements of Mintzberg's 'adhocracy'. The basic element of organisation was the project sub-unit. This can be seen as a response to the technical complexity and unpredictability of the work within each project, making a self-contained unit a more effective form of organisation than coordinating the separate functional areas involved. Project units were not only changeable in the sense that they came and went as projects did. They also changed in composition during the life-cycle of a project.

Development projects comprised these stages:

- requirements analysis
- high-level design of the overall system
- low-level design
- coding
- code-reading
- testing
- documentation

The basic division of labour within a project unit was between more experienced staff involved in requirements analysis and high-level design - 'designers'- and less experienced staff who performed the remaining five stages - 'implementors'. The rationale for this division appears to have stemmed partly from the complexity of designing a telecommunications switching system, which meant that it was advantageous to concentrate design work in as few heads as possible, using a small team of designers who could sort out interface problems by direct lateral contact. The separation of 'design' from 'implementation' also

appears to have been a way of utilising the most skilled staff most fully. Because design work was the first stage of a project, and thus also the first part to be finished, using design staff intensively at the earlier stages and then leaving later stages to others meant that design staff became available as quickly as possible to start a new project.

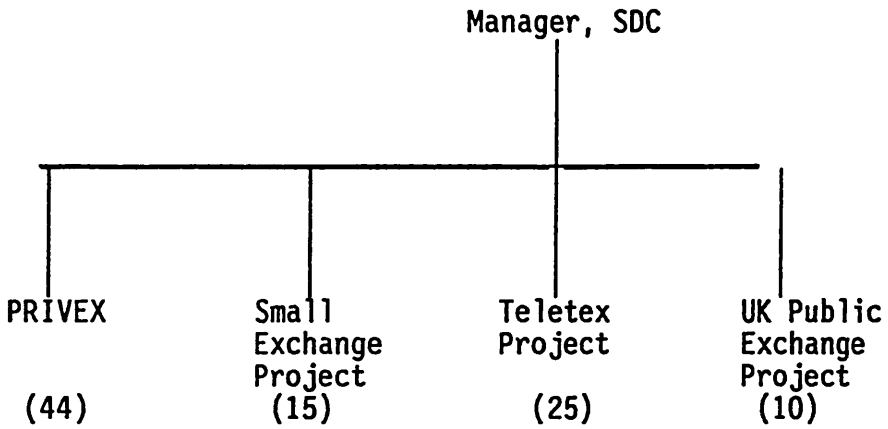
The typical cycle of development ofaproject over time was then as follows. As the project progressed, the initial team of designers would reduce in size but continue to perform remaining areas of high level design, whilst other teams - perhaps as many as ten - would be added, to carry out implementation. Each team would be responsible for a particular area of the overall high-level design. Complexities within each area could then be dealt with within a team. Generally, a team's work was split into a large number of system elements, and low-level (program-level) design, coding, testing and documentation for each one carried out by one team member, with a colleague performing code reading. On production of the program level design for each element, the team member concerned would produce a detailed plan for testing the final code, prior to integration with the complete system. Towards the end of the development, the integration of the various elements would be the responsibility of a separate integration team.

Ideally, the number and size of implementation teams would reach a maximum a few months before the delivery date, and then would begin to decline once the bulk of coding and testing were over. Gradually, the remaining implementation teams would make a transition to carrying out first documentation and minor modifications and then those maintenance and modification tasks on installed systems that could not be handled by local field engineers.

The orderly flow of staff from one project to another was however not always possible. In order to retain the capacity to staff a project that Exco senior management knew it was likely to get, or one that was just emerging from the design stage, it was often

Figure 9.2: Organisation of the Software Development Centre

(Numbers of personnel indicated in brackets)



necessary to retain more staff than were necessary on the wind-down stage of another project. In such cases, implementors were retained to perform detailed documentation or to investigate faults reported in installed systems. Labour hoarding of this form was a fairly standard feature of the shifting configuration of project units within the SDC, and was a response to the unpredictability of demand as well as the large size of projects.

The organisation of the Centre at the time of field work is shown in Figure 9.2.

9.2.3 Career Structures

The educational background of software engineers at Exco was typically either a degree in computer science, or else some other science or engineering degree involving a considerable degree of computing experience, such as mathematics or physics. There were unfortunately only opportunities to interview 'implementors' and staff engaged on maintenance within the PRIVEX project. Table 9.1 shows that, of 9 PRIVEX staff interviewed all but one had this sort of background. The exception was a recently recruited team leader with responsibility for the team providing maintenance and support services. Although without academic qualifications, he had over 10 years industrial experience with various types of real-time software.

Project staff had job-titles according to the standard Telecorp system set out in Figure 9.3. As a rule, design team members were Senior Programmers or above, and had at least five years experience, whilst implementation team members had job-titles between Programmer 3 (the grade to which graduates were recruited) and Senior Programmer. According to managers, the progression from implementation work to design work was a normal one to make with time, although many implementors would leave before reaching the design level. Table 9.2 shows the expectations of eight junior members of the PRIVEX project in four different implementation teams and of one team leader, as to

TABLE 9.1: THE NATURE OF THE WORKFORCE WITHIN EXCO

Sex	Job Title	Age	Education	Nature of First Job	First Employment at Exco	Length of time at Exco	Professional Affiliation	TU Member
F	Programmer 3	20-24	B.Sc. Comp. Sci.	Programming	Yes	1 year	None	No
M	Programmer 3	20-24	M.Sc. Comp. Sci.	Programming	Yes	1 year	None	No
M	Programmer 2	20-24	B.Sc. Maths	Operations Research	No	1 year	None	No
M	Programmer 2	20-24	B.Sc. Physics	Hardware Support Engineer	No	2 years	None	No
M	Programmer 2	20-24	B.A. Maths	Programming	Yes	2 years	None	No
M	Programmer 2	25-29	B.Sc. Comp. Sci.	Programming	No	1 year	None	No
M	Programmer 2	20-24	B.Sc. Physics	Programming	Yes	2 years	None	No
M	Senior Systems Engineer	30-34	CSEs	Programming	No	1 year	None	No

Table 9.2: Job individuals expect to be in in five years time

		Team A	Team B	Team C	Team D	Support Team Leader	Overall
N		2	2	3	1	1	9
		EDDDDDDDDEDDDDDDDDEDDDDDDDDEDDDDDDDDEDDDDDDDDE					
	1	3 1	3 2	3 2	3	3	5
SYS DES ,TECH CO		3	3	3	3	3	55.6
		EDDDDDDDDEDDDDDDDDEDDDDDDDDEDDDDDDDDEDDDDDDDDE					
	4	3 1	3	3 1	3 1	3	3
SW DEV MGT		3	3	3	3	3	33.3
		EDDDDDDDDEDDDDDDDDEDDDDDDDDEDDDDDDDDEDDDDDDDDE					
	7	3	3	3	3	3 1	3 1
SW DEPT MGT		3	3	3	3	3	11.1
		EDDDDDDDDEDDDDDDDDEDDDDDDDDEDDDDDDDDEDDDDDDDDE					
	Column	2	2	3	1	1	9

Table 9.3: Career goals of software producers

Mean scores for importance on a scale of 1 to 9, broken down according to position in the division of labour, and overall.

	Team A	Team B	Team C	Team D	Support Team Leader	Overall
N	2	2	3	1	1	9
						(s.d.=0.8)
Developing a Career in one Company:	4.0	4.0	5.6	7.0	6.0	5.1 (s.d.=1.4)
						(s.d.=0.9)
View on transferability of skills: (Scale: 1= 'not at all' to 7='completely')	4.0	5.0	6.0	6.0	6.0	5.3 (s.d.=1.5)

Figure 9.3 Job Titles within Exco

Programmer 3 (Graduate Trainee)
Programmer 2
Programmer 1
Senior Programmer
Team Leader

the sort of work they would be doing in five years' time. The implementors all anticipated moving into design work or into software management. Table 9.3 shows that they attached only moderate importance to staying with one company, and saw their skills as having a considerable degree of transferability.

The overall picture of the workforce at Exco is then one of a single career structure or pool of university-educated computer scientists, assigned to project sub-units usually for a few months or years at a time, with divisions of labour between design and implementation based on experience. There was no vertical segementation between 'professional' and 'technician' jobs. Whilst staff gained experience and expertise in particular projects they were working on, there do not seem to have been any horizontal specialisations in career structures. Indeed we shall see that one of the main tensions was the desire for software producers to move regularly from one project to the next so that they could gain breadth of experience.

A caveat must be added to this analysis. It is possible that less educated software producers with lower career opportunities were employed in some of the services departments. In particular, there was probably a great deal of routine work in the Software Production Centre, which may have given scope for a 'technician' grade. Unfortunately, it was not possible to negotiate access to these areas.

9.2.4 Overall Control and Coordination

Data was collected on the mechanisms of control and coordination being used in the PRIVEX project, as well as the way these mechanisms had developed over the last two years of the project's life. PRIVEX was in a wind-down stage. Over the previous two years, it had gone through a design and implementation cycle for its last major enhancement. This enhancement had been achieved using three successive releases of new features, produced at roughly nine-monthly intervals. The last of the implementation work on the third release was being completed, although a special version of the PBX was in the middle of being developed for a European Amercorp division. Project organisation at this juncture is shown in Figure 9.4. The project had fewer and smaller implementation teams than at the height of implementation, grouped under two implementation managers who reported to an overall project manager.

It was widely recognised within Exco that the approach to setting targets, monitoring and controlling work had had to change drastically with the move from corporate resourcing to being an internally-trading profit centre. One project manager described the old regime in the following, no doubt rather exaggerated, terms:

> "There used to be absolutely no limits ... . You could put as many people as possible on a project and test as much as you wanted to ... ."

Clearly, the need to tender competitively was seen as leading to much tighter targets and controls. On the whole, managers at all levels appeared still to be adjusting to the more rigorous requirements and to regard current control arrangements as provisional, to be improved.

The basic mechanism used for overall control and coordination of the project throughout its life-cycle was action-planning. Action plans were drawn up first of all for designers and then for implementation teams on the basis of specifications for portions of the system and time targets for developing them.

In keeping with the general picture of evolution in organisation and systems of control, the methods for arriving at specifications and targets and for drawing up work plans underwent considerable change over the product's lifecycle. This was a response to the need to improve the method of handling renegotiating prices and deadlines when difficulties were encountered in this very complex system. The emphasis in the evolution of these procedures was on taking account of interface issues in dividing up a large and complex overall design for implementation in several different teams, and also in leaving room for renegotiation when problems were encountered.

(a) Specifications and Estimates

Planning for the last major enhancement of PRIVEX began with a call from Exco to the customer division for 'Change Requests'. These specified modifications in what the product could do that were felt to be desirable, regardless of cost. The change requests were at first considered by a design panel consisting of a Design Consistency Manager - an experienced designer reporting directly to the project manager, two implementation managers and three team leaders who were finishing off and maintaining the previous enhancement. The purpose of involving implementation team leaders at this early stage was to tap their knowledge of the existing system, so that ramifications of a change in one area of the system on all other parts of it could be identified as early as possible and taken into account in the decision to proceed. Each member of the panel examined all the change requests in one of 20 functional areas of the system where they had expertise. They produced a 'high-level' or functional specification for an addition or modification to the system in

that area.

Specifications for each new feature were circulated to all other members of the panel for them to add their opinions on possible ramifications on the areas where they had particular knowledge. The Project Manager then made decisions as to which features should be included for further consideration, and arrived at a total of 160. He then negotiated a rough specification for what was to be included in the enhancement as a whole with the customer division, its development cost and delivery date. At this stage, the 160 new features were spread over three annual releases, to spread the load of testing new features in the context of the whole system.

(b) Action Plans

Once the three packages of features had been decided, implementation team leaders for the new enhancement were appointed, and groups of features allocated to each one. The implementation team leaders then performed the next stage of design work, the high level design of the logical structure of the software to perform the functions specified for each feature, along with a refined estimate of the development effort required. There was no formal language or standard format for notating designs. Estimates of effort were also made without formal aids, entirely on the basis of the experience of implementation team leaders.

These logical designs were submitted to the implementation manager and the project manager for approval. Between them, they conducted final negotiations with the customer divisions as to the specifications of each version of the enhancement, its overall cost and delivery date. The result was a target for each of the two implementation managers in terms of specifications for code to be produced over a period of almost two years, divided into three sub-targets, each with a delivery date and an overall cost. At this point, five implementation teams, concerned with

five principle functional areas of the system, were staffed up. Headcounts allowed some extra capacity to take account of maintenance work on installed systems.

The three delivery dates and budget allocations set for successive releases of the three releases set overall targets for the two implementation managers. Within these targets they were given a great deal of discretion as to how they set detailed targets for each of their teams. Groups of logical specifications and overall targets for them were allocated to each of the team leaders, who had a similar level of discretion in allocating work and setting targets for each team member.

In general all the logical specifications dealing with a particular area of the system were allocated by a team leader to one individual, for two reasons. First, it allowed team members to design and then code a number of specifications concurrently, allowing them to improve on the total estimated development time which had been calculated on the basis of each specification giving rise to a separate piece of development. This usually gave rise to some slack within an individual's workload to deal with unforeseen complexities. Second it reduced the need for communication between team members as to design interfaces, design and coding standards.

To summarise, the process by which project targets were set at the outset of the development was one where complex negotiations were carried out between two sorts of software specialists, designers and implementors, on the one hand and marketing experts in the customer division on the other. The result was a workplan for the project development workforce as a whole defined with considerable certainty for a period of almost two years. This approach attempted to confine complexities of estimating and target setting in a few experienced heads at the beginning of the project, leaving the control of performance to meet these targets as the main uncertainty to be tackled in the management of subseqent activities.

(c) Modifications to Planning Mechanisms

This approach was judged as having worked well in terms of identifying the likely impacts of proposed modifications. In the opinion of one Implementation manager, these would not have emerged had design been performed solely by a specialist design team. One of the basic problems with the approach was however that estimates of effort were often not realistic. One of the reasons for this was the difficulty of estimating low-level design and coding costs on the basis of the still high-level logical designs produced by team leaders. These often proved inadequate when lower level design was begun, so that further specifications had to be added for about half of the original feature specifications.

For the specification of the 'special' versions of PRIVEX underway at the time of fieldwork, the method of drawing up specifications was altered somewhat. The basic difference was that each feature was contracted separately, and in two stages, design and implementation. This made estimates more accurate, because each feature was effectively recontracted at the half-way stage.

Change requests from the customer division were first of all considered by a team of two specialist designers. Requests were turned into estimates for the design work involved in meeting them, which were submitted to the Project Manager and then the newly-created Design Centre for approval. If accepted by the customer, each one of these then became a contract for a piece of design work. As well as each contract being for design work only, it was also for one feature only. The design team then carried out the design of the logical system for each feature, in rather more detail than had previously been the case, so that fewer problems would have to be resolved by implementors. They consulted implementation team leaders if the feature in question was seen as having complex interactions with code in a number of areas. This logical design work then served as a basis for a

further contract for implementing the feature, once it had been approved by the project manager, the Design Centre and, finally, the customer.

The net result of this move to contracting for design work separately, as a response to technical complexity and unpredictability, was then a removal of some design work from the sphere of implementation teams.

As activity on PRIVEX was wound down, the system of work allocation for the implementation teams, other than the one engaged on the new 'specials', also changed in nature. Allocations became much less rigorously linked to project targets, as teams were occupied with completing only the last few features, called for when problems had been encountered in earlier specifications. More and more, implementors were engaged on maintenance and documentation of code. According to one implementation manager, it was often by no means clear who actually needed the documentation being produced. Furthermore, most of the programmers who had actually worked on the implementation had now either left the company or moved on to other projects, and the three implementation teams were largely staffed by young computer scientists who had been with Exco less than a year. A major aim of the exercise was to keep them occupied whilst another project got to the stage where junior implementation staff were required.

It would appear that during this 'wind-down' phase in the project, at least some of the work allocated was no longer for the purpose of meeting project targets for code that had to be delivered. Documentation work was being allocated to further departmental rather than project goals. It served the departmental priority of having implementors available to undertake work on new projects, once design work had been completed on them.

Four implementation staff completed diary sheets to record

assignments over a ten week period. They reported assignments running from one to eight weeks. This very partial data gives a picture of assignments several weeks long.

9.2.5 Control of Individual Performance

(a) Professional Skills and Commitment to Targets

The basic form of control of individual performance to meet the targets was that software producers were expected to have skills, ability to exercise technical judgement, and motivation needed to do so. The project manager and implementation managers were all clear that non-codified skills of experienced development staff had played an important role in the control of work throughout the development of the last enhancement. They had been the basis on which staff could be expected to meet their targets, for both design and implementation phases. Section 3 describes the things managers saw as important in maintaining motivation to apply these skills to meet norms of productivity and quality.

(b) Structured Techniques and Standards

Non-codified skills were supplemented by codified techniques for the lowest level of design and actual coding, which performed according to strict structured programming standards. Managers considered this specific form of the standardisation of skills as vital for assuring quality. However, the implementation managers considered that it was still important to have skilled programmers for good code to be introduced. It was also important for their expertise to develop in order to produce staff suitable for more responsible design work.

At the beginning of development of the last PRIVEX enhancement, no definite design technique or notation had been used, apart from the broad method of top-down refinement. Function specifications and logical designs were depicted using flow

charts and annotations in English, in a style evolved informally amongst project staff. During the course of implementation, however, the project and implementation managers made attempts to introduce Keyword Design Language (KDL) as a notation. This met resistance from many implementors who found it difficult and inconvenient to learn at first. Its use gradually increased over the course of the three versions of the enhancement, and had become fairly standard by the time of fieldwork. Managers however considered it of more relevance to settling interface ambiguities

than for assuring the quality of code. It was not considered an absolute priority.

(c) The Availability of Help

Table 9.4 illustrates the importance of the availability of help and advice with technical problems to team members in performing tasks allocated. It suggests a generally cooperative way of working within teams, where help might be sought several times a day, and was as likely to come from colleagues as from team leaders.

(d) Project Reviews and Monitoring

Reviews at 'milestone' points were occasions on which individuals were called to account for their work. They served above all to examine quality of work, but also had a time-control function. They set milestone dates by which major phases of development were supposed to be completed, and so provided a coarse form of progress monitoring. There were four distinct stages of review during the course of a development:

- High level logical designs were checked by the implementation and project managers.

- Program-level design produced by implementation team members was checked by the team leader.

- Code was read through by implementation team colleagues.

- Module-level and integration testing were carried out by, respectively, implementation team members and the 'integration team'.

All of these reviews were fairly informal, without any set format and without a set review panel. Their importance for identifying issues of quality and appropriateness for user needs can be judged from one implementation manager's assessment that modifications were made to most features at each of the above four stages. The more major problems were usually detected at the first and second review stages. The last two served to detect low-level design and coding errors, which were usually not serious.

Table 9.5 shows that 7 implementation staff completing diary sheets reported on progress informally on a roughly daily basis. It illustrates a basically low intensity of informal monitoring practiced by supervisors in-between review points.

9.2.6 Low-Level Coordination

Apart from the basic mechanism of action planning using specifications, coordination of the work of software producers within teams was achieved by the use of structured programming and the KDL design representation as forms of **standardisation of skills**, and through **direct lateral contact** to resolve ambiguities. Data from diaries completed by 7 team members, shown in Table 9.6 give some indication of the pattern of contacts over interface ambiguities. They show the continued importance of almost daily direct lateral contacts between implementors, and

Table 9.4: Time Spent receiving help

		From Supervisors		From Colleagues	
	N	No. of contacts per day	Total time (min) per day	No. of contacts per day	Total time (min) per day
Team A	2	2.0	9.7	2.6	7.6
Team B	2	0.6	8.0	1.6	20
Team C	2	3.8	2.3	1.9	21
Team D	1	0	0	0	0

Table 9.5: Time Spent Reporting Informally To Supervisors

	N	No. of contacts per day	Total time (min) per day
Team A	2	0.4	1.3
Team B	2	0.5	7.2
Team C	2	2.1	5.0
Team D	1	0.3	3.3

Table 9.6: Time Spent discussing interfaces

		With Supervisors			With Colleagues	
		N	No. of contacts per day	Total time (min) per day	No. of contacts per day	Total time (min) per day
Team	A	2	0.5	4.3	1.0	6.1
Team	B	2	0	0	0.8	27
Team	C	3	1.0	3.8	2.0	17
Team	D	1	0	0	0	0

suggest that the use of KDL had not eradicated the need for contact and mutual understanding of each other's work.

9.2.7 Adjustments to Plans

The above mechanisms could not ensure that work was performed according to specification and targets of time and budget. Unpredictable technical problems were encountered frequently, often when high-level specifications proved to be ambiguous once design to the next level of detail was undertaken. In such cases, slippage from time targets might result, as extra time was taken to resolve the complexity. It was necessary to collect information on slippage as it occurred and necessary to make adjustments to development schedules.

(a) Vertical Information Collection

A computerised system for monitoring the amount of time each team member spent on each stage of the features allocated to them was scrapped by mutual agreement of the implementation manager and team leaders at an early stage. Team leaders found entering information on the computer time-consuming, and were also suspicious of the running estimates of work left. Once a task had been begun, they preferred to discuss with team members and re-estimate the amount of effort still required, taking into account additional information that might have emerged as to the complexity of the work involved.

This computer-based system was replaced by one in which team members gave a brief weekly written report to team leaders, setting out the amount of time they had spent on allocated tasks and estimates for the amount of work remaining. The report compared the resulting total estimate for the tasks in the individual's work plan with the orginal estimate. In providing re-estimates that differed greatly from original allocations, team members would be aware that a justification in terms of un-

anticipated complications had to be given. Such re-estimates would first be discussed informally with the team leader.

Whilst the written reports from team members provided 'hard' information for project control, the regular informal contacts between team leaders and team members indicated in Tables 9.4, 9.5 and 9.6 were important for providing 'softer' information as to the nature of problems and progress with them. A system of weekly meetings between each implementation manager and his team leaders passed this information up the hierarchy. Team leaders' monthly reports were supposed to 'contain no surprises'. On the strength of these meetings, as well as 'hard' data on progress, implementation managers' reports to the project manager would contain information needed for project control. This included some speculation as to problems that might arise, or corrective action that might be necessary in a few months.

The Planning and Control function required implementation managers to complete a separate monthly report on progress relative to targets, but this information was not in fact used for internal project management. It was used for giving monthly progress reports to the client consortium. In practice, the information given to Planning and Control was for the last completed month of work, whereas Implementation Managers reported at the end of each month on the current month's work to the Project Manager. This meant that the information given to the client had already been digested a month previously by the project control system. The principle was that any corrective action for problem areas should already be underway by the time the client got to hear of them.

(b) Adjustment Mechanisms

We have already seen that one important issue with respect to the quality of systems under development was ambiguity of specifications. The weekly meetings between implementation managers and team leaders were one mechanism by which such design

ambiguities and other unanticipated technical complexities were dealt with. Simple problems in interfacing modules being developed by different teams could be dealt with by altering design specifications. Such alterations were often made at the meetings, although they had usually been discussed between the team leaders involved in the course of normal daily contacts. Very simple interface issues would not need to be brought to a meeting. For serious design ambiguities, it might be decided at a weekly meeting to raise a new feature specification. If approved by the project manager and client it would then be added to the project schedule.

The basic mechanism for rescheduling work allocations when new features were added, or when slippage on existing ones occurred, was the deployment of **slack resources** built into work plans in a number of forms.

First, the Project Manager could negotiate to 'borrow' individuals from another project which did not have a full work load, either because design work had been held up, or because development work was over, and team members were waiting to be transferred to a new project. In both of these cases, slack resources made available through allowances built into project plans or the nature of project life-cycles in other parts of the organisation were used to cope with slippage. Second, smaller amounts of extra resources could allow existing team members to work overtime for a period in order to meet a delivery date. Finally, implementation managers could make a case to the project manager for more resources (i.e. an increased headcount) for a particular team on a more permanent basis. On the 'specials' development this would be a request to apply to the customer division for more funds, through contracting for a further feature.

9.3 MECHANISMS FOR PRODUCING COMMITMENT TO WORK

Team leaders and implementation managers identified the following mechanisms for maintaining the motivation of software producers to meet the norms of quality and productivity implicit in the way targets and specifications were drawn up.

(a) **Maintaining regular contact:** Tables 9.4, 9.5 and 9.6 together illustrate the frequency with which 7 team members in late 1982 had contact with their team leaders over a range of task-related issues. Implicit in this sort of management activity was also discussion of wider concerns, general support and reinforcement of norms of productivity and standards of work.

(b) **Skill development:** ensuring where possible that team members were engaged in areas of work where their skills were being developed, and arranging for individuals to change project at least every two years to increase their experience.

(c) **Team atmosphere:** ensuring that they were working in a team where they 'fitted in' socially;

(d) **Providing training and development:** regular opportunities to go on training courses, even when not strictly called for by the demands of current work;

(e) **Providing information:** keeping team members informed as to work that was coming up, so they knew that new experience was on the horizon;

(f) **Providing promotion, annual pay rises and constructive appraisals:** Table 10.7 shows that, for 9 PRIVEX staff interviewed, both using skills independently (i.e. 'problem solving ability' and 'good technical knowledge') and adherence to procedures, were perceived as important criteria by which individuals were assessed for promotions.

Table 9.7: Perceived importance of various criteria for promotion.

Mean scores for importance on a scale of 1 to 9, broken down according to position in the division of labour and overall.

	Team A	Team B	Team C	Team D	Support Team Leader	Overall
N	2	2	3	1	1	9
Problem Solving ability:	7.5	6.5	7.0	7.0	-	7.0 (s.d.=0.8)
Performing according to procedures:	6.5	5.0	7.0	8.0	7.0	6.6 (s.d.=1.1)
Good technical knowledge:	6.0	7.0	6.7	8.0	8.0	6.9 (s.d.=1.1)
Understanding of application area:	5.5	5.5	6.0	6.0	9.0	6.1 (s.d.=1.8)
Management ability:	5.0	3.0	6.0	8.0	8.0	5.3 (s.d.=2.1)

Perceived assessment criteria clearly reinforced orientations towards working autonomousy in the application of complex technical skills, although they also encouraged the following of set procedures. The latter undoubtedly reflected the importance individuals felt management attached to structured standards.

9.4 ISSUES AND TENSIONS

9.4.1 Issues for Management on the PRIVEX Project

PRIVEX managers did not refer to either the productivity or quality of work of direct producers as being an issue, indicating a lack of explicit conflict over standards of individual performance. Whatever problems they experienced, it was not with standards of individual performance associated with a 'professional' emphasis in labour control. The issues they were contending with were as follows.

(a) Sustaining Staff Motivation And Keeping People

Managers identified a number of the things which were difficult to arrange in practice although important for motivating staff.

First, they felt it was difficult to motivate people at the tail-end of the PRIVEX project, with many having left the project and most of those remaining engaged on final coding, documentation and maintenance. Work was simply not very challenging, and people were wanting to know when they were going to be back on 'real' development work.

> "Once a system has been delivered. there's generally a period of unrest. People are just beginning to realise there's a lot of maintenance to be done afterwards."

The PRIVEX project manager felt it was difficult to re-assure people in this count, because he himself didn't know what was in the pipeline:

> "Lines of communication are very long at Exco".

Managers felt that the 'taking an interest in people', their needs and motivation, was generally underdeveloped at Exco. They tended to concentrate on the technical issues.

Second, there were problems of keeping staff on a long project like PRIVEX, lasting several years. Implementation managers ideally wanted the same staff in the implementation teams for successive enhancements, because they knew not only the design standards, but also the system itself, and became skilled at judging the effects of a change in one area of code on other areas of the systems. The problem was that implementors tended to see staying on one project for more than two years as a disadvantage in terms of their career development.

In a situation where software technology was changing almost from year to year, managers felt that implementors saw their position in the labour market as dependent on continuous skill development and exposure to new technologies. This meant gaining in-depth experience on one sort of system and then moving on to something else, preferably more 'state of the art', in the general area of real-time communications software. Staff had generally wanted to leave the PRIVEX project after about two years. They had either asked to be transfered to other projects, or else left the company.

(b) Imprecisions In Specifications

The PRIVEX project manager felt that it was often very difficult to communicate with the customer division's senior marketing people to get a clear picture of the functions to be performed in each enhancement, and of the relative importance of different features. He was sure that Exco personnel could transform a 'change request' into a high-level specification, and then develop software that accurately performed the functions represented. However, there was often doubt as to how precisely the change request represented what the client in fact wanted. This lack of clarity in individual change requests often made it difficult to get a clear opinion out of the client as to which features should be included in each release. Further, when the result of the design stage of a new feature was presented to the client for approval, client representatives were likely to claim it was something other than what they had requested.

(c) Uncertainties Within The Development Process

Implementation managers felt that the sheer technical complexity of the software they were developing was inevitably a major problem in managing its development. One claimed that, in developing new features for a complex system such as PRIVEX, it was sometimes possible for an individual to spend an hour coding a new feature or part of one, to discover that it had some unanticipated and disastrous consequences for other parts of the system. It was then sometimes necessary to spend a week tracking down the interactions that were the cause of the problem. This sort of complexity made estimating targets and progress very difficult, and 'very subjective'.

This led to a further problem. Estimating of this sort of complexity required considerable experience, and team leaders often found it difficult to know how to break up the estimates into blocks for assignments to individual team members. Many team leaders lacked the experience to do this finer level of

estimating well.

(d) Uncertainties In Planning

Technical uncertainties were seen as leading to uncertainties in planning and scheduling. This area was a major worry for both implementation managers. Unforeseen technical problems and 'changes' in user requirements made keeping to schedules difficult, and often led to specifications for new features to supplement existing ones. Keeping track of these could be difficult, let alone scheduling them.

9.4.2 Issues For Software Producers

Tables 9.8 to 9.14 present data on the nature of job experiences for the 9 PRIVEX staff interviewed, with the sample broken down according to which team staff were on. Team A was engaged in recoding low-level designs produced during the main PRIVEX enhancement, so that the PRIVEX system could be run on a more powerful set of microprocessors. It was engaged entirely on coding and testing. Team B was engaged on completing the implementation of the final batch of logical designs for 'late' features for their area of the system, as well as maintaining and documenting code already developed. Team C had completed all implementation work and were engaged on maintenance work and documentation. Team D was however engaged on actual development work, producing diagnostic software for testing PRIVEX hardware. The Support team leader was responsible for a newly-formed team that was to attend to serious problems with the new enhancement that were reported from the field.

Table 9.8 shows that, on average, those interviewed felt that their work-related goals were being achieved rather less than satisfactorily. There was in fact considerable variation in both measures of goal achievement, with significant differences between the team-based groupings for the second measure. On the

Table 9.8 Views on overall goal achievement

Mean scores for goal achievement on a scale of 1 to 7 (4='satisfactory'), broken down according to position in the division of labour, and overall.

	Team A	Team B	Team C	Team D	Support Team Leader	Overall
N	2	2	3	1	1	9
Work achieving most important personal goals:	2.5	4.0	3.0	5.0	4.0	3.4 (s.d.=1.1)
Job measures up to what you want:	2.0	4.0	3.0	6.0	6.0	3.7 * (s.d.=1.6)

Table 9.9: Experience of utilisation of skills

	Team A	Team B	Team C	Team D	Support Team Leader	Overall
N	2	2	3	1	1	9
Time demands: (Scale: 1='ext. light' to 7='ext heavy')	5.5	4.0	5.0	5.0	6.0	5.0 (s.d.=0.9)
Technical demands: (Scale: 1='ext. light' to 7='ext heavy')	2.5	4.5	4.3	4.0	6.0	4.1 (s.d.=1.5)
% time below technical capacity:	65	55	33	40	10	43 (s.d.=2.3)
% time trivial or routine:	75	55	27	40	10	43 (s.d.=2.5)
% time show what you can do:	5	50	43	70	80	43 * (s.d.=3.1)

Table 9.10: Experience of autonomy in carrying out tasks:

	Team A	Team B	Team C	Team D	Support Team Leader	Overall
N	2	2	3	1	1	9
% time using own approach:	05	40	83	90	80	57 * (s.d.=3.6)
% time making all important decisions:	05	15	40	90	40	32 (s.d.=3.2)
% time specs already agreed by others:	100	75	40	30	20	58 (s.d.=4.2)
% time every step has to be approved:	45	85	23	20	80	48 (s.d.=3.4)
% time given detailed instructions:	20	45	20	20	60	30 (s.d.=2.2)

Table 9.11: Impact and visibility of contributions

	Team A	Team B	Team C	Team D	Support Team Leader	Overall
N	2	2	3	1	1	9
% time involv. in business decisions:	00	05	03	00	20	04 (s.d.=0.7)
% time important contrib:	20	35	40	80	70	42 (s.d.=2.4)
% time contr. to business success:	15	20	30	50	80	32 (s.d.=2.6)
% time work of special interest to management:	15	15	26	10	100	28 (s.d.=3.1)
% time higher mgt doesn't know what ind. is doing:	50	40	33	30	10	36 (s.d.=3.1)
Freq discuss with higher mgt:						
Daily						0
Weekly				1	1	2
Monthly		1	2			3
Six monthly			1			1
Yearly		1				1
Never	2					2

Table 9.12: View on Salary

	Team A	Team B	Team C	Team D	Support Team Leader	Overall
N	2	2	3	1	1	9
View on salary, relative to comparable firms: (Scale: 1='much lower' to 7='much higher')	4.5	4.5	4.7	6.0	5.0	4.8 (s.d.=0.7)
View on salary plus perks, relative to comparable firms: (Scale: 1='much lower' to 7='much higher')	4.5	4.5	4.7	6.0	5.0	4.8 (s.d.=0.7)

Table 9.13: Views on probable timing of next promotion

	Team A	Team B	Team C	Team D	Support Team Leader	Overall
N	2	2	3	1	1	9
Less than 6 months:	1	1	1			3
6 months - 1 year:	1			1	1	3
1 - 2 years:		1	2			3

Table 9.14: Satisfaction with Promotion Prospects

Mean scores for satisfaction on a scale of 1 to 7 (4= 'about right') broken down according to position in the division of labour, and overall.

	Team A	Team B	Team C	Team D	Support Team Leader	Overall
N	2	2	3	1	1	9
View on probable timing of next promotion:	5.5	4.0	4.7	6.0	4.0	4.8 (s.d.=0.8)
View on probable size of next promotion:	4.0	4.0	4.0	4.0	4.0	4.0 (s.d.=0.0)

Table 9.15: Attitudes to Learning Advanced Software Engineering

	Team A	Team B	Team C	Team D	Support Team Leader	Overall
N	2	2	3	1	1	9
View on learning new techniques, on a scale of 1 to 7: (1= 'very opposed', 4= 'indifferent', 7= 'very keen')	5.5	6.0	6.3	5.0	4.0	5.7 (s.d.=1.1)

one hand, members of Team A development team members were fairly fulfilled, and on the other, the Team D member and the Support Team leader were quite dissatisfied. In all, 5 of the 9 staff interviewed felt that their most important personal work goals were being achieved less than 'satisfactorily'.

This suggests that at least for some in the sample interviewed there were considerable tensions between individuals' priorities and what was offered them by system of work organisation, and control. In order to arrive at a more detailed understanding of areas of harmony and conflict between the priorities of producers and of management control, we now examine data as to how producers experienced those aspects of their jobs relevant to the work goals explored in Chapter 7.

(a) Skill Utilisation

Table 9.9 shows the extent to which individuals experienced their skills as being utilised. Most of those interviewed experienced both time and technical demands of their job as moderate, although the Support Team leader felt much more extended technically, whilst the junior programmers engaged in coding activity on Team A felt very underextended. The uniform experience of moderate time-demands suggest that none of those interviewed would be likely to feel underutilised, on the one hand, and, on the other, that there were no major issues in terms of the level of productivity at which these individuals were expected to work.

Whilst experiences of technical demands were also on average moderate, the low levels registered by some however raise the possibility that specific individuals in this sample experienced misutilisation. Turning now to the extent to which producers felt they were being given 'the chance to show what they could do', there were some significant differences between groups, although the small numbers mean that interpretation has to be tentative. Those engaged primarily on post-development maintenance work, on Teams B and C seem to have felt their skills

were being moderately utilised. They felt they were being given the chance to 'show what they could do' for about half the time. This fell to less than 10% for both the Team A members interviewed, who were engaged on coding typical of the wind-down phase of a large development project. Team A members seem not to be having their goals of skill utilisation met, at least in their current work.

In contrast both the Support Team leader and the Team D member engaged on developing software for testing newly-installed hardware felt challenged for over 70% of their time. The former was in the midst of coming to terms with a demanding task for which he had specifically been hired, whilst the latter was responsible for the entire span of developing an intricate and self-contained piece of software.

(b) Control over the Approach to Work

Table 9.10 shows how those interviewed saw the extent to which they had autonomy in performing their tasks. In terms of the degree to which they controlled the approach to their work, there was considerable variation over the sample, and also significant variation between the groups. At one extreme the Team A members felt they had almost no control over the way they did their work and that they were working to somebody else's specification all the time. At the other extreme the Team C members and the Support Team leader all felt they were using their own approach over 80% of the time and working to somebody else's specification for 20-40% of the time. Whilst on average, the sample felt that they were 'given detailed instructions as to how to proceed' for about one third of the time, the majority experienced themselves as 'making the most important decisions' for less than half the time. This suggests that staff engaged on the 'wind-down' implementation tasks typical of the sample as a whole experienced tensions with respect to their desires for controlling technical decisions relevant to their work.

This view is supported by data obtain from two implementors, who felt that that the arrangement by which high-level design was done prior to lower level implementation by different people meant that it was often difficult for them to have any contact with the designer whose work they were bringing to fruition. Designers often moved on to a new project soon after implementors arrived on the scene. This led to feelings of lack of involvement in and influence over design decisions.

(c) Experiences of Structured Techniques

All those interviewed felt that for developing the sort of applications software they were engaged with there was really no alternative to using structured programming. Some form of structured low-level design was necessary for making sure that designs were clear and comprehensible, so that different parts of the system fitted together and maintenance work could be handled easily. Structured programming was important not only because it led to clearer code, but also because it meant there was consistency of style between programmers. There was consensus that the sort of structured standards applied to development work was appropriate and that the level of adherence to them was high. Feelings of lack of control over technical decisions do not then appear to be linked to the use of structured standards, but rather to the lack of input implementors at the tail-end of a project had into design decisions.

There were at the same time some reservations as to the relevance of structured techniques to maintenance work, particularly when this involved diagnosing faults and writing patches in low-level assembly language. The Team D member engaged on developing diagnostic software also felt that structured techniques were not relevant to his work because he was writing code in a low-level language, with very restricted memory space. Structured design standards were of no use when trying to maximise effective use of memory.

(d) Impact and Visibility

Table 9.11 summarises perceptions of individuals as to the impact of their work amd visibility of their contributions. None felt that they were involved in business decisions to any appreciable extent. Feelings of the extent to which they were contributing to business success or making an individual contribution were more positive. There was however considerable variation, but no significant differences between the groups. However, only the Team D member and the Support team leader felt they were both making an important contribution and contributing to business success for more than 50% of the time. There is here a clear suggestion that those engaged on 'wind-down' implementation activities experienced not only a lack of influence over technical decisions and low skill utilisation, but also a more general lack of influence and contribution.

(e) Pay and Promotion

Table 9.12 shows that salaries were generally thought to be a little higher than what would be on offer in comparable firms. There did not appear to be tensions over pay.

Tables 9.13 and 9.14 show that all respondents expected to be promoted within the next two years and were satisfied with the size of step-up anticipated. The picture is one of individuals expectant of promotions and/or pay rises on an annual basis, and satisfied with respect to these goals.

9.4.3 Overview: The Nature of Tensions

In Section 9.4.1 we saw that managers attributed slippage to unanticipated technical complexity rather then low productivity or poor quality work. This suggests that there were not major conflicts over productivity or quality. Managers were happy with

the levels of productivity and quality of work they were obtaining through a form of labour control that emphasised 'professional' commitment. The high proportion of the, mostly junior, staff interviewed who experienced low goal fulfilment, suggests however that there were considerable conflicts between their preferences and what the organisation was currently offering them. We have seen that these conflicts were on the whole not associated with issues of individual performance levels. They stemmed from other management concerns. They stemmed from a form of project organisation designed to coordinate work in a very complex and unpredictable task environment, which emphasised the concentration of design work in a few experienced heads.

The system of action-planning divided work up into design and implementation tasks in order that it could be coordinated and led to the creation of tasks of limited scope for junior producers. The need to make the limited number of experienced designers available for the next project as soon as possible meant that these designers were not available to implementors, contributing to feelings of lack of technical influence in the latter.

The need to have a stock of implementors to staff the next large project for a subsequent implementation phase meant some were kept misutilised, working on routine documentation for the completed PRIVEX enhancement. Managers experienced problems in maintaining staff motivation during 'wind-down', because it was not possible to maintain some or all of the conditions they identified as encouraging commitment to work. It appears that project targets were set quite laxly at such times. It is possible that producers in fact wanted to work harder, given the low time demands they experienced and the weak positive relationship between time demands and goal satisfaction, established in Chapter 7.

Finally, the need to take advantage of rapid developments in hardware technology, through converting the PRIVEX software to a

new format for a new type of microprocessor, meant a team of implementors A were engaged on merely recoding existing designs. All of these practices undermined work goal achievement in junior staff.

It is not clear from the data collected how this low goal achievement manifested itself in behavioural terms. It does not seem to have resulted in any organised militancy or rejection of project targets, presumably because junior producers had hopes of better and more responsible work to come. It is most likely that it resulted in unrest in the form of requests for transfers and threats of leaving.

Given that producers wanted to be more fully utilised, their demand was most likely to be to know that more stimulating development work, in which they could be seen to be achieving results, was on the way. This is supported by the complaint of a project manager that he himself was not kept sufficiently in the picture to re-assure his staff on this score.

9.5 MANAGEMENT INTENTIONS AND POSSIBLE OUTCOMES

This section discusses management intentions for change in the organisation of Exco, reflecting the management concerns we have identified within the PRIVEX project.

(a) Intentions Stemming from Concerns with 'Motivation'

One group of intentions concerned ways of improving the extent to which it was possible to motivate people by providing them with satisfying work. This was more concerned with keeping skilled people, than with mobilising commitment to project targets, which was largely assumed. Management intentions here were very much in line with the continuation of a 'professional' emphasis in labour

control.

(i) 'Giving People More Attention'

Both implementation managers and the project manager on PRIVEX thought they needed to give more attention to the development needs of staff, in order to improve their motivation generally, and to encourage them to stay with the organisation once they had gained skills in particular products, come to understand Exco standards, and were known to work according to the desired norms of productivity and quality. They thought that more effort could be put into providing training. The project manager also favoured more regular promotion opportunities, even if the promotion was in terms of job title only. He felt that promoting people increased their sense of being valued by the organisation.

(ii) Moving Away From Large Development Contracts

The PRIVEX project manager reported that, as a consequence of experiences on PRIVEX and other large projects, Exco senior management had decided to move away from large contracts.

One reason was that such projects were complex to monitor, control and reschedule, particularly when, as with PRIVEX frequent adjustments to designs and schedules had to be made as technical problems were encountered. When the total personnel on a project reached 30 or more, the interactions between technical problems could be very complex to sort out, involving individuals from several different teams. Once a problem had been identified, moreover, working out how this affected the development schedules of the

different teams concerned could be "horrendous".

A second reason for moving to smaller projects was however also to do with encouraging motivation. Two years was reckoned to be the maximum amount of time a producer would want to stay on one project before they began to feel that they needed to be broadening their experience on some other type of development. It was considered that the surest way to avoid losing people

was to put them on projects lasting no more than two years.

Together with the consideration of avoiding very large teams, this led to a policy of taking on projects that would involve less than 25 people for less than two years. Concerns with providing the sort of conditions that would motivate people were thus such that they had an impact on business strategy.

(b) Moving Out of The Purely Corporate Market

One important consequence of the strategic autonomy of Exco from the corporate centre was that by the end of 1982 there were intentions at senior management level of moving out of the strictly corporate market into the more general market for communications software development. Reasons were not made available as to precisely what was behind this move - possibly because it was still a bone of contention with the Telecorp Board. However it is probable that the motives were that advantages were seen in the wider market for ensuring supply of work for the future, and also for selecting projects of the preferred smaller size. This strategic intention appeared to be entirely consistent with perpetuating the existing paradigm of work organisation.

(c) Product Diversification and Moves Towards Matrix Organisation in Combined Hardware-Software Projects

Over the two years before fieldwork took place, Exco's Engineering Development Centre had begun to tender for development of some 'hardware' contracts for industrial process control systems. Whilst in the 70s these had involved designing specialised integrated electronic circuits, by the early eighties, there was a demand for programmable systems, based on the use of programmable standard micro-processors. As a consequence, assembly language programming of microprocessor systems was being carried out by electronic engineers within the EDC. Moreover, the development of industrial process control systems based on this sort of technology had been identified by senior management as a viable 'external' market the company could move into.

At least one manager within the SDC referred to resentment within his department that tenders for such systems with a large software element could originate from within the EDC without reference to the SDC. The solution canvassed with senior management was a move away from a functional split between software and hardware in development project organisation, to a matrix approach with EDC and SDC personnel on project teams for developments of this sort. By the end of the fieldwork period, there were already some EDC personnel reporting to SDC-managed development teams.

This sort of development appears to be consistent with the continuation of the current paradigm of work organisation. It does however suggest that over time, there would be a redrawing of boundaries between hardware and software areas of expertise. There is also some evidence of conflict between hardware and software developers over who should be responsible for the area of assembly-language programming of microprocessors.

(d) The Creation of 'Troubleshooting' Technical Consultant Positions

According to the PRIVEX project manager, one change it was expected that the recently-appointed new head of the SDC would make was the creation of a number of 'technical consultant' positions, staffed by experienced developers with considerable design experience, reporting directly to him and without fixed attachments to any one project. These people were to act as 'troubleshooters', getting involved in projects for a few weeks at a time, when design difficulties emerged during implementation. They were to act as an expert resource that could be called upon to deal with unforeseen technical complexities, minimising slippage and disruption of schedules.

The implications for this initiative for the existing paradigm of work organisation are not easy to assess. At one level, the creation of rewarding and responsible senior 'technical' posts would appear to be consistent with motivating junior people to use their skills on development projects. We have established that promotion opportunities were seen as a way of encouraging motivation and commitment to project targets. These posts would open up an avenue of promotion to positions of technical responsibility, as an alternative to those of team leader, implementation manager and project manager, where technical responsibility and influence were mixed with managerial elements.

At the same time there were possible problems in terms of the impact of these positions on the motivation of development team members. Since team members were motivated by having the opportunity to use skills and take responsibility for an area of work, they could experience their responsibility and control over the approach to work as undermined if an external 'expert' were called in every time a major problem was encountered. The actual impact would probably depend on the extent to which the external consultant took on a 'coach' role, working with the team members concerned to solve the problem, as opposed to taking the problem away and solving it for them. Whilst working with team members

and educating them might take longer, and so might be less effective in dealing with slippage on the particular project concerned, it might have better results in terms of general staff motivation and development.

(e) Intentions Regarding Structured Techniques

Managers interviewed within the PRIVEX project did not mention any intentions regarding the introduction of new structured programming or design techniques. However, the issue was raised in the form of the recently-created corporate Software Engineering Manager, who intended to introduce greater corporate control over software engineering techniques. This amounted to the Telecorp centre seeking to introduce an element of standardisation of skills into the way it controlled its operating divisions, in addition to control of outputs in financial terms.

It is difficult to assess the likely impact of this initiative on the work organisation and forms of labour control used within Exco. There is first of all the issue of how much impact corporate pressure would have on Exco procedures. Exco management saw ensuring norms of individual productivity and quality of work were maintained in terms of motivating people, rather than introducing more rigorous structured techniques or other aspects of advanced software engineering. Managers appeared quite satisfied with the existing structured programming standards and the use of KDL.

Table 9.15 shows that those direct producers interviewed were fairly keen to learn more advanced techniques than they were currently using, except for the Support Team leader, who was indifferent, because he did not see development techniques as having much relevance to his maintenance specialism. These data suggest that producers did not see more advanced techniques as a threat to fulfilment of their work goals, rather the reverse. At the same time, two junior producers felt that, whilst it was

necessary, structured design techniques took discretion away from low-level designers and made their work 'less original'. There is thus the possibility that more rigorous forms of structured design would have further routinised the work of junior implementors. However, it is also likely that the promise of more responsible work would have maintained the interest of these staff.

CHAPTER 10

SYSCO, MANUFACTURING SECTION

SUMMARY: SYSCO, MANUFACTURING SECTION

1. BUSINESS STRATEGY AND TASK ENVIRONMENT

Market: Selling and customising dataprocessing packages for manufacturing companies

Businesss Strategy: Focussed differentiation through quality, in established market with established products.

Task environment: Complex technically and commercially. Stable commercially and technically, except for unpredictable problems in asssessing user requirements, and unpredictable demand for maintenance.

2. ORGANISATION, JOB STRUCTURES, CONTROL AND CO-ORDINATION

Organisation: Staff with software background in marketing department as well as main production department. Within production dept, seperate pools of systems analysts, responsible for requirements analysis, and systems designers/programmers.

Career Structure: Promotion from programming to systems design to systems analysis to management, in production or marketing. i.e single career structure with no vertical segmentation, but no closed internal labour market. Some horizontal specialisation in different hardware systems & application areas. Mixture of graduate and non-graduate staff.

Overall Control & Co-ordination: Activities on many small projects controlled by dept. manager through centralised action planning, based on specifications and estimates refined at several points during customisation process, by analysts and design/prog team leaders in collaboration. Co-ordination between designers & programmers achieved using company specific standards, including structured programming.

Labour Control: Achieved through 'professional' skills and commitment in the context of targets and specifications, backed up by informal monitoring by team leaders, & milestone reviews. Quality assured through standardisation of skills with structured programming, & availability of help from team leaders/colleagues.

Low-level Co-ordination: structured programming standards & direct contact.

Adjustments to plans: Information concerning occasional technical problems collected 'vertically' through informal monitoring; plans adjusted to take account of slippage or demand for maintenance work by renegotiating deadlines.

3. MECHANISMS FOR PRODUCING COMMITMENT TO WORK

Selection. Offering skill development & training. Offering pay raises & promotion. Allowing producers to 'see a project through'. Direct communication of norms to junior producers.

4. ISSUES AND TENSIONS

For Managers: Finding the 'right people'. Developing staff with 'application area' skills. Marketing & Product strategy. Getting accurate req. specs.

For Producers: Lack of skill utilisation & contribution for junior staff. Lack of adherence to standards by management. No senior practitioner posts.

5. INITIATIVES FOR CHANGE AND LIKELY OUTCOMES

Initiatives: New products, and redrawing of company boundaries. Interest in new structured techniques from first line management only.

Possible outcomes: Continuation of 'professional' control of labour.

10.1 ENVIRONMENT AND BUSINESS STRATEGY

The Manufacturing Section of Sysco employed about 50 people in 1982 and was one of two profit centres in Sysco, which was in turn one of three companies making up Sysgroup Ltd. Sysgroup provided a broad range of services to industrial and commercial users of data processing systems. The Manufacturing Section of Sysco was primarily responsible for selling and customising a modular set of software packages, Packset, designed to provide any or all of the systems required by manufacturing companies. In general it sold and installed complete hardware and software systems, buying in standard hardware - mainly minicomputers. It marketed to the UK and internationally, except for the U.S.A, which was covered by Sysgroup's parent organisation.

Expectations of Resource Providers

Sysgroup developed by diversification from a small data processing consultancy specialising in the needs of manufacturing firms set up in 1964. The sixties saw the emergence of a basic competitive strategy for the group as a whole that in 1982 was still important for shaping performance expectations placed upon Sysco and its Manufacturing Section in particular.

The basic form of competitive advantage sought by Sysgroup directors was to differentiate their software services in carefully selected markets within the general area of commercial data processing, on the basis of quality and comprehensiveness of service, whilst offering competitive prices. These goals were seen as achievable through three basic aspects of the way Sysgroup companies were expected to operate.

The first was the use of a comprehensive methodology for supplying software services to a client, spanning the organisational, commercial and technical sides of identifying requirements, developing the system and installing it. In the

mid 60's the company obtained financial support from the (then) National and Regional Development Council (NRDC) to develop the Sysgroup System Design Methodology (SDM).

The second was the use of customisable applications software packages which decreased the cost of developing and supplying systems through re-using code that had already been developed.

The third aspect of the way Sysgroup companies were expected to operate concerned focussing on a limited number of market 'niches'. During the 70's, other firms, particularly software divisions of mainframe and mini-computer hardware manufacturers had begun to market software packages as part of a total systems service for manufacturing users. Since hardware manufacturers had much larger sales forces, which they could utilise for marketing software as well as hardware, Sysgroup began to concentrate on particular niches in the market for manufacturing systems which the hardware manufactures had overlooked or were weak in. It also diversified into a range of new application areas for service industries such as hotels, banking and insurance, and into new computer-related services, such as providing bureau facilities and training in systems development skills.

The Manufacturing Section of Sysco was not expected to undertake fundamental product innovation, but to market an established set of products and associated services, so as to build up corporate reserves for financing new ventures. It was expected to undertake new product development only in the form of periodically upgrading packages. Most such enhancements were undertaken, however, by a product development division of the US parent company.

Business Strategy

The business strategy of the Sysco Manufacturing Section was shaped by the role outlined for it above within the overall group strategy. The basic competitive strategy was thus one of

focussed differentiation on the basis of quality.

The Section sought to sell and customise data processing packages, combined with consultancy services, to manufacturing companies, competing primarily on the basis of the quality and comprehensiveness of its services rather than price. Its senior management emphasised in particular the importance of Sysco experience in implementing systems as well as designing and coding them. The head of the Manufacturing Section software production department (see the organisational chart in Fig. 10.1) estimated that in a medium sized project of £600,000 charged to the client, 60-70% of that could be for implementation and general consultancy work. The remainder would roughly be equally divided between hardware and software, with hardware tending to account for more.

The need to identify product market niches had led to a focus on two particular areas. The first was in small, basic systems required by small manufacturing concerns of up to £20 million annual turnover, which had not previously used computerised data processing. This was a growth market, which hardware manufacturers were not exploiting because their packages were less modular than those marketed by Sysco. It was difficult for most competitors to strip their systems down to the essentials and get the price below £100,000. In contrast, Packset consisted of six basic modules, Financial Control, Order Control, Cost Control, Manufacturing Control, Inventory Control and Business Planning. Not only could each of these modules be bought in isolation, but each was made up of three sub-systems or 'elements' which were also 'free standing'. Sysco could often sell a basic version of Packset for £60,000, made up of £50,000 for a minicomputer and £10,000 for software and consultancy. In such cases, very little customisation was viable, and clients had to be prepared to adapt their procedures to the requirements of the package.

The second niche was at the other end of the market. When large multi-site manufacturing concerns were looking for large integrated data processing systems, Sysco management felt that customisation expertise and general consultancy support, as well as integrated software were important selling points, and ones on which they could compete sucessfully.

Environmental Strategy

Allied with this business strategy was a degree of environmental management, through cooperative relations with potential competitors. In both of its market niches, Sysco had established cooperative relations with minicomputer manufacturer 'competitors'. These were aware that Sysco had identified niches in the systems market in which they were not in direct competition. They saw Sysco software products as a way of selling hardware to clients whose needs they would otherwise not be able to market to. One minicomputer manufacturer had arranged discounted sales of hardware via Sysco, and had invited sales representatives from the Manufacturing Section to give material to its own sales force.

Task Environment

Its business strategy meant that the Manufacturing Section was responsible for maintaining documentation on Packset releases and undertaking a small amount of product enhancement work; marketing its services, identifying client requirements, and customising software to meet them; installing systems and providing consultancy to clients as to how to use them; maintaining systems after installation, and providing modifications as required by clients. The emphasis in this set of tasks was very much on selling and customising established products. In terms of the total volume of activities, in 1982 the Manufacturing Section was expected to sell about 2 large integrated customised Packset systems, and 6 to 8 basic systems at the low end of the market.

The resulting task environment involved complexity, in both 'commercial' and 'technical' areas. Considerable knowledge of application areas and of Sysco products was needed to deal with sales leads. Technical complexity took the form of knowledge needed to understand user requirements and customise intricate pieces of software. The task environment was at the same time basically stable. Both market niches were established and growing at a fairly predictable rate. The commercial aspects of the task environment were predictable. Since the Section was working in applications areas that were well established and using packages that were mostly several years old, the types of technical problems encountered in programming systems were also fairly well known. The main form of unpredictability lay in the area of assessing user requirements and specifying systems. The variety of organisational forms and influences in user organisations meant that novel situations were often encountered.

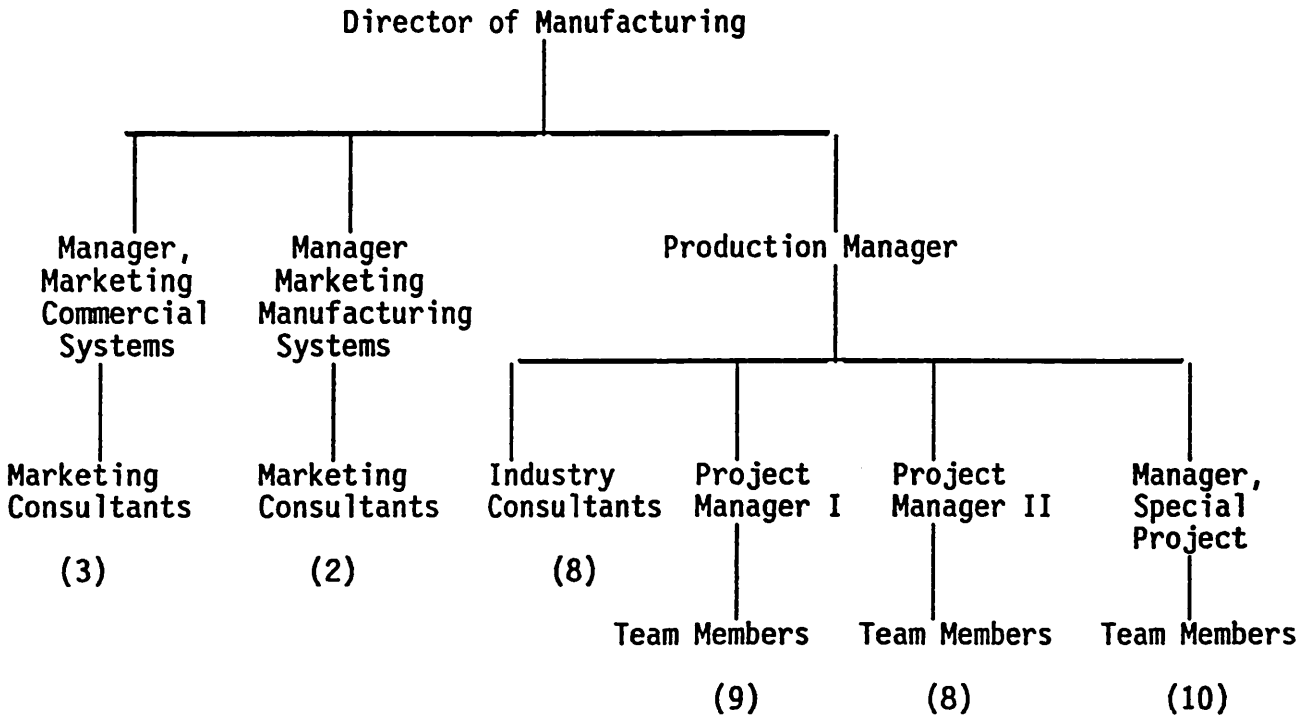
10.2 ORGANISATION, CAREER STRUCTURES, CONTROL AND COORDINATION

10.2.1 Organisational Structure

The organisation of the Manufacturing Section is shown in Fig. 10.1. It had two basic functional departments: Marketing and Production. This functional split can be seen as a response to the different types of complexity involved in marketing and selling systems on the one hand, and customising, installing and maintaining them on the other. The bulk of software staff were employed in the Production Department, but the small Marketing function was also staffed by people with a background in product development or customisation work.

The Marketing area was divided into two wings, each with its own manager. One was concerned with the 'commercial' Packset modules i.e. Cost Control, Financial Control and Order Control, and the

Figure 10.1: Organisation of the Production Department



(Departmental numbers shown in brackets)

other with the more specifically 'manufacturing' modules. The rationale for this was that there was a considerable market in firms that were interested in the commercial modules that did not want inventory control or production control. Both Marketing Managers saw themselves as more concerned with selling than marketing per se. Most of their time was spent reacting to enquiries, because Sysgroup was well established by reputation and there was also a larger Sysgroup Marketing section. They assessed user needs and whether Sysco could meet them with a Packset customisation. When the feasibility of a sale had been established and an agreement to proceed in principle had been reached with the client, the project was passed to the Production Department, where further feasibility studies were performed and estimates produced before proceeding to a contract for customisation and installation.

The Marketing Departments also took the initiative in the development of product enhancements within the Manufacturing Section. On the basis of their knowledge of user requirements, they identified weaknesses in Sysco's range, and annually submitted proposals and estimates for product enhancements for consideration at Sysgroup level.

10.2.2 Divisions of Labour Within the Production Department

The Production Department was also divided into two basic wings, although both were under the control of one manager. Broadly their responsibilities were systems analysis and installation consultancy on the one hand, and computer system design and programming on the other.

In the first wing there were eight 'industry consultants', the most experienced members of the department, who performed the more detailed feasibility studies making up the **Definition** stage of a project once it had been handed over from Marketing, leading to the drawing up of a detailed contract for a customisation. They also performed the first phase of **Planning** the

customisation, the Functional System Design (FSD).

FSDs were handed over to one of two development project managers in the second wing of the department. Each specialised in one of the two main versions of Packset, designed to run on different minicomputers. Each managed a pool of staff experienced in taking specifications and customising the system design and programming of Packset modules for that particular hardware. According to the Production Manager it only took a few weeks for a member of staff to convert their expertise from one version to the other, and so staff were reasonably transferable between the two pools. Project managers allocated staff first of all to complete the **Planning Stage** by performing a logical design of the software, the Computer System Design (CSD). Usually, the CSD was accomplished by a single experienced developer. Project managers then allocated up to four less experienced staff, depending on the degree of customisation, to carry out the **Build Stage**, consisting of program-level design and coding, followed by testing of each program and the system as a whole, and the documentation.

In simple cases requiring very little customisation, the CSD, programming and testing might all be done by one programmer. In cases where it was agreed that practically no customisation was required, the small amount of customisation of code might be performed at the user site by the industry consultant, without setting up a separate programming team.

In the more general case where a programming team was necessary, whilst it was producing the customised code, the industry consultant continued to work at the client site, preparing the organisation for the **Installation Stage** of the project. This involved preparing manuals, non-computerised data handling procedures and forms to deal with the inputs and outputs to the computer system, as well as training user staff. If there was an existing computer system which the new system was replacing, it was also necessary to design conversion software for converting data on the old system to the new. The system was then installed

by the industry consultant, working together with one or two members of the programming team. Installation involved conducting a user acceptance test and making revisions to the code and user manuals in the light of problems emerging. Medium sized projects of a total of £700,000, utilising 4 programmers, would typically run for 6 to 8 months from feasibility to installation.

As well as having allocation and control functions, project managers were 'direct producers', performing functional system design and installation consultancy on some projects exactly as industry consultants did. They also performed a large proportion of the sales support work required by Marketing, providing advice and initial estimates on the basis of their knowledge of their particular version of Packset.

In addition to the eight industry consultants, the two 'general' project managers and their respective pools of programming staff, there was also in 1982 a 'special project' team, with its own project manager. This team was concerned with customising a single very large manufucturing and inventory control system for a particular client. The project manager was an industry consultant, and the team included other industry consultants as well as programmers. Such an arrangement was common for large projects, and was in this case due to last for over 18 months.

The Production Department was also responsible for responding to maintenance calls from clients with Packset systems installed, and for development work on internally funded product enhancements. Maintenance work was in general not charged to the client, but charged to an internal maintenance budget made up from a proportion of all sales. A typical maintenance problem was when a part of a recently installed system 'crashed' and required emergency attention. Sysco generally agreed to take on such small 'emergency projects' as part of the total service. Maintenance, modification and product enhancement were all the responsibility of the programming wing, and were organised by one of the two project 'general' managers according to which version

of Packset was concerned.

10.2.3 The Nature of the Workforce and Job Structures

The Sysgroup Personnel function produced a set of very general job descriptions corresponding to a series of job titles, shown in Fig 10.2. Within the Production Department, junior programmers and programmers were responsible mainly for program-level design, coding and testing, plus some installation work. Senior programmers and technical consultants undertook more responsible CSD work. Consultants, senior consultants and principle consultants dealt with feasibility studies FSD work. There was a general relation between job title and level of work, but no definite correspondence.

According to the Production Manager and the Sysgroup Personnel Director, as staff on the programming side gained more experience they were given more responsible design work and also promoted. More experienced CSD-level staff would also be introduced progressively to aspects of FSD work and user-oriented consultancy, opening up the path to further promotion to a project manager or industry consultant work.

Figure 10.2: Job Titles and Responsibilities

Junior Programmer	6 months	
Programmer	18 months	Program Design, Coding,
Senior Programmer	2 years	Installation Computer System Design
Technical Consultant	2 years	
Consultant		Functional System Design, Installation Consultancy, Project Management, Sales Support
Senior Consultant		
Principal Consultant		

According to the Production Manager, Sysco recruited only staff with experience. At the programming level this meant recruiting people usually with a higher education background, who had trained elsewhere as COBOL data processing programmers. Sysco also took some two or three people each year from the Sysgroup graduate trainee programme, but only after they had been trained in COBOL programming and had had a few months experience in another Sysgroup company, usually on maintenance programming. It was not considered important that recruits by either route had a university background in computer science, because this was of little relevance to commercial data processing programming.

Table 10.1 shows the qualifications and job histories of 19 staff interviewed within the production department. It shows that there was no difference in educational qualifications in terms of

subject or level between 'junior' programming staff and 'senior' consultancy staff. At all levels, there was a mix of staff who had degrees in applied computing, science and engineering, and others who had begun their working lives in some form of clerical or craft engineering work, and then found an opportunity to train as programmers when computer systems were introduced into their area of work. There was one senior programmer with an arts degree. No staff at all were members of trade unions, and only two consultant-level staff had current membership of the British Computer Society. Neither regarded their membership as at all significant for their standing in the company or the external labour market. One found it occasionally of use as a channel for finding out about new technical developments.

Table 10.1 provides support for the picture that experience of data processing software work rather than any specific academic training was the basis for access to more senior and responsible positions. There was considerable variation as to where senior staff had gained that experience. Whilst some had achieved their status by promotions up from programming jobs within Sysco, at least as many had been recruited in as consultants already experienced in systems analysis. Amongst programmers too, only two had started their working lives at Sysco. Whilst the Manufacturing Section offered a promotion ladder, it did not operate a closed internal labour market.

Table 10.2 shows that of the 19 staff interviewed all but two programmers saw their career as leading into the realms of systems analysis, or management over the next five years, in keeping with the nature of promotions available within Sysco. Table 10.3 however shows that staff did not see their horizon as restricted to Sysco. They considered their skills as highly transferable, and did not see developing a career within one company as particularly important.

Overall, then, the Manufacturing Section was characterised by a lack of vertical segmentation in its career structures. There were, however, some elements of horizontal specialisation. As

Sex	Job Title	Age	Education	Nature of First Job	First Employment at Sysco	Length of time at Sysco	Professional Affiliation	TU Member
M	Programmer	25-29	B.A. Comp.Science	Systems Analyst	No	6 months	None	No
M	Programmer	20-25	B.Sc. Maths & Computing	Programming	No	6 months	None	No
F	Programmer	20-25	B.Sc. Physics	Programming	Yes	6 months	None	No
M	Junior Programmer	20-25	B.Sc. Chemical Engineering	Programming	Yes	6 months	None	No
M	Programmer	20-25	'O' Levels	Clerical Work	No	1 year	None	No
M	Programmer	20-25	HNC Computer Studies	Programming	No	4 months	None	No
M	Senior Programmer	25-29	M.A. History	Programming	No	4 months	None	No
M	Senior Programmer	20-25	'A' Levels	Programming	No	3 years	None	No
F	Technical Consultant	20-25	B.Sc.Biochemistry	Programming	Yes	2 years	None	No
M	Technical Consultant	25-29	City & Guilds Electrical Eng.	Engineering Craft Work	No	3 years	None	No
M	Consultant	36-40	'A' Levels	Accountancy	No	9 years	None	No
M	Consultant	36-40	B.Sc. Naval Architecture	Naval Architecture	No	6 months	None	No
M	Senior Consultant	36-40	B.Sc.Mech.Eng.	Programming	No	2 years	None	No
M	Senior Consultant	40-45	B.Sc. Physics	Operational Research	No	11 years	None	No
M	Senior Consultant	40-45	B.Sc. Applied Computing (Mature student)	Programming	No	2 years	None	No
M	Senior Consultant	26-30	HNC Comp. Studies	Programming	Yes	7 years	None	No
M	Senior Consultant	26-30	'A' Levels	Programming	Yes	10 years	None	No
M	Principal Consultant	26-30	HND Maths & Computing	Programmer	No	2 years	None	No
M	Principal Consultant	36-40	'O' Levels	Company Secretarial Work	No	14 years	None	No

Table 10.2: Job individuals expect to be in in five years time

	Prog.	System Designer	Industry Con-sultant	Project Manager	Overall
N	7	3	6	3	19
Responding	6	3	6	2	17
SYS DES, TECH CONS	2				2
FUNCT SYS ANAL	3	1	1		5
SW DEV MGT.		2		2	4
SW SALES, MARKETING			2		2
PROJECT, CLIENT MGT.	1		3		4

Table 10.3: Career goals of software producers

Mean scores for importance on a scale of 1 to 9, broken down according to position in the division of labour, and overall.

	Prog.	System Designer	Industry Con-sultant	Project Manager	Overall
N	7	3	6	3	19
Developing a Career in one Company:	2.7	5.7	3.8	3.7	(s.d.=1.5) 3.8 (s.d.=2.3)
View on transfer-ability of skills: (Scale: 1= 'not at all' to 7='completely')	6.3	6.0	5.5	6.0	(s.d.=1.3) 6.0 (s.d.=0.9)

they progressed through the grades, as well as skills in designing and programming with COBOL, programmers acquired expertise on a particular minicomputer system, in the working of particular Packset modules and in a particular broad set of and inventory control. This horizontal specialisation can be seen as a response to the complexity present in each distinct application area, and in understanding the systems software of different mini-computer systems.

10.2.3 Overall Control and Coordination

The Director of Manufacturing maintained overall control over the Marketing and Production functions by setting each six-monthly performance targets in terms of revenue and expenditure. Each was effectively treated as a profit centre. For the Marketing function, revenue came from licence fees charged for Packset modules or elements, regardless of whether they were customised or not. Marketing obtained no commission for the volume of customisation work or installation consultancy in the Production Department that resulted in the Production Department from sales. Its out-goings were the costs of its own staff and overheads, as well as internal payments to the Production department for sales support. For the Production department revenue came from staff time charged to clients for development work and installation consultancy, as well as cross-charges from Marketing. Costs were staff time, overheads and machine time used in development work.

This separation for overall control for Production and Marketing can be understood as stemming from the basic commercial predictability of the Sysco's markets. The flow of enquiries to Marketing was reliable. Although complex judgements had to be made as to how to deal with them, a steady flow of work to the Production Department could be depended upon.

Within the Production Department, its manager maintained overall control in seeking to achieve financial targets through a system of action planning, which allocated tasks in the various stages of customising and installing systems to his staff.

In essence the system was this: The Production manager drew up six-monthly work plans for the industry consultants and for the two pools of 'general' development staff, based on time and cost targets and specifications drawn up for successive stages of individual customisation contracts, negotiated with clients on a fixed cost basis. A separate work plan was drawn up for each successive stage of the special project, based on activities and costs negotiated with the client for each stage. Here, we examine briefly the important features of the way specifications and estimates were drawn up, and the way they were compiled into action plans.

(a) Specifications and Estimates

In line with procedures set out in the Sysgroup Systems Development Methodology (SDM), overall estimates for customisation projects were produced first of all at the conclusion of the initial survey, then refined once or twice during the course of the Definition stage, and refined again during the Planning stage, when the functional system design and computer system design were produced. The conclusion of each stage also saw the production of a detailed specification and estimate for the next stage, which were reviewed and agreed with the client. The functions of these elaborate estimating and specifying arrangements can be understood as follows. First, they were a way of ensuring that users were involved at several points during the course of a customisation in reviewing system specifications. This was a way of assuring the quality of systems. Second, the 'iterative and interactive' approach to estimating advocated in the SDM was a way of ensuring that estimates given to clients were accurate and could be met.

In achieving this accuracy of estimation, direct lateral contact between staff with an appreciation of the users requirements and staff with up-to-date experience of development was important. In the earlier stages of a project, this combination of perspectives was between marketing consultants and industry consultants. In the later more exact phases of the Definition Stage, industry consultants and senior programming staff worked together on estimates. At all stage, estimates were made initially by staff who were actually going to be involved in the next stage or in controlling it, although they had to be approved by the Manufacturing Manager.

Estimating was performed through the application of uncodified experience rather than through the application of any formal procedures. According to both the SDM manual and the Production Manager, those providing estimates would be aware of certain factors to be taken into account e.g. size and type of client company, size and complexity of programs to be developed, but there were no rigorous or explict techniques for measuring these parameters or turning them into estimates.

The Production Manager stressed that the whole point of the estimating procedure was to avoid giving a definitive figure for the overall cost until after the FSD had been performed and requirements had been both ascertained and agreed. However, he felt it important to avoid raising the overall estimate significantly from an initial rough quote supplied by marketing to the figure in the contract after the FSD, because clients would suspect sharp practice in luring them into a more expensive contract than they had initially wished. He felt it was vital to be as accurate as possible at the initial quote, and if possible, given that they were not competing primarily on price, to err on the high side. He was also concerned that the Production Department should not get saddled with unrealistically tight schedules because he felt that his people could not work well if subject to more than a certain level of pressure.

For these reasons he encouraged industry consultants providing

initial estimates in a sales support role to 'protect' the Production Department by providing generous estimates. There was little danger of losing contracts because the marketing consultant concerned would 'carry' the more purely commercial sense of what the client would stand. The direct lateral relations between Marketing and Production thus served not only to facilitate trade-offs between functionality and cost, but also to balance 'production' and 'commercial' criteria in estimating. On occasions the Production Manager would assent to a project being estimated in a very tight way, because it had strategic value. However, his general philosphy was that projects where the client was not prepared to pay the cost of development when 'estimated reasonably' were not worth having. He saw the price of Sysco systems as related to the cost of development rather than what the market would bear.

(b) Action Plans

The predictablity of demand meant that work could be allocated to producers with certainty for one month ahead at a time, and with a reasonable certainty for up to six months. The basically predictable nature of demand seems to have made compiling action plans to allocate work a fairly straightforward exercise. Projects were simply 'queued up' in the workplan, and finish dates offered to clients depending on the amount of work already in the order book. On rare occasions an important project was brought to the front of the queue, although this usually meant rescheduling finish dates for some others. The six month time frame allowed projects to be planned ahead and commitments to a finish date made, although monthly revision of allocations was necessary to some extent to take account of new projects arriving, as well as occasional slippage.

The Production Manager was responsible for assigning both the industry consultants and the two 'general' pools of programming staff to projects, as well as allocating the resources the special project required. With respect to the latter, he agreed

a work plan every six months with the special project manager, based on the estimates and workplans drawn up for the current phase with the client.
For the main group of staff, he drew up plans allocating people to projects on the basis of (a) the amounts of industry-consultant time estimated for feasibility studies, depth studies and FSD's in the early stages of a project and (b) amounts of time of staff of various levels of experience estimated for the CSD, Build and Installation phases. Individuals were allocated to projects in multiples of half-week blocks.

The work plans drawn up by the Production Manager were used to allocate tasks to individuals in two ways. First, the Production Manager allocated whole phases of project Definition, FSD, or Installation, without breaking them down into smaller units, to industry consultants. Second, the two general project managers broke down the CSD, program design, coding and testing phases of projects into smaller assignments for individuals within their 'pools'. Such assignments were at the level of designing or coding a portion of a Packset module.

Data collected from diaries filled in by 14 staff showed the following range of assignment sizes over a ten-week period:

Program-level staff:	3 to 10 days
CSD level staff:	1 to 12 days
Industry Consultants:	1 to 24 days

This data confirms the general impression given by the Production Manager that more senior staff were entrusted with larger single assignments.

The Production Manager aimed also to build a level of extra capacity into the department so that there was staff time available in each six monthly workplan beyond what was needed for contracted customisation work. In the case of industry consultants, 'spare time' was needed to provide sales support consultancy to marketing. It was not possible to predict exactly

when such calls would be made, and so 'movable holes' had to be left in plans.

In the case of programming staff and the two project managers, spare time was built in to provide capacity for responding to equally unpredictable calls for maintenance and 'emergency' modifications. These generally involved a few days or at most weeks work, and were estimated by one of the two project managers, although the Production Manager had to certify the estimate before a contract was signed. In practice this time was made available by allocating three or four programmers at any one time to tasks on 'internal work' enhancing existing Packset product code. When maintenance calls were received, the project manager concerned with the system in question allocated someone to it, generally from 'internal work'. Project managers had discretion in this sphere, and did not as a rule refer to the Production Manager.

10.2.5 Control of Individual Performance

This section describes the mechanisms the Production Manager and project managers saw as important for controlling the quality of work and productivity of direct producers within the Production department.

(a) Skills and Commitment in Meeting Targets

The control of work was seen by managers as underpinned by an assumption of a basic commitment of project staff to work to norms of productivity and quality in meeting targets for blocks of work allocated to them, without intensive surveillance or direction. There was a strong emphasis on 'professional' labour control. They were expected to apply expertise with discretion and responsibility. The next section discusses factors seen as important in reproducing the necessary commitment to project targets.

(b) Standardisation of Skills

A number of levels of standardisation of skills were also seen as important for assuring quality of work, for making staff predictable in what they produced in their exercise of discretion.

First, the Systems Development Methodology was a set of guidelines for the organisation and management of development and customisation projects. It was used for training people as they reached the consultant level in skills of organising a project into different phases of activity, of identifying who should be involved in them, of when the various estimates and specifications described in the previous section should be drawn up, and how progress should be monitored. It was basically a manual of project management. It served to make sure that industry consultants could be relied to conduct their projects and their relations with clients in a reasonsably uniform way.

In terms of techniques used for each stage of a project, the SDM contained checklists as to the main activities that should be included in a phase e.g. for the CSD, 'design basic data-structure/database, design inter-program flow, design terminal conversations ... define network communications requirements', but did not specify techniques. It did not go down to the level of specifying particular analysis, design or programming techniques or detailed procedures that should be followed.

Functional System Design and installation work were carried out primarily on the basis of the non-codified expertise and powers of discretion and judgement, gained through experience, of staff at consultant level or above. There was no special formalised standardisation of techniques or ways of representing system designs, beyond flow charts annotated in English. However, a second level of standardisation of skills was important for

controlling this area of work. Staff had to be assessed as having certain levels of general experience and knowledge before being assigned to these tasks, and so their skills were

standardised at a general level. They were not however expected to use any definite detailed methodolgies, nor to have any definite professional certification or training.

Programmers were also assigned work on the basis of their general skill and knowledge level. However, their work was also controlled through a more detailed, third level of standardisation of skills. They were expected to apply the Jackson method of structured programming, from the level of program design downwards. This was seen by management as a technique for producing well designed programs that were easily read and modified, but which required discretion and judgement derived from non-codified expertise to be used to best effect.

(c) Availablity of Help

Within the programming pool, contact with both supervisors and colleagues had an important role to play in assuring the quality of work through providing assistance with problems encountered. In the case of staff engaged on program-level design, coding and testing, this advice was typically given through contact with project managers, who were responsible for the technical supervision of their work, or through talking to other programmers. Of the four staff at this level who completed diary sheets, on average they contacted their supervisors 0.7 times a day for help, spending 3.4 minutes a day in doing so. They contacted a colleague for help 1.1 times a day, for a total of 8.1 minutes a day.

Technical consultants appear to have been much less dependent on others for technical help. On average, the three who completed diary sheets asked their immediate supervisor - their project manager - for help only 0.1 times a day, for a total of 0.7

minutes. They asked a colleague for help on average 0.3 times a day, and spent 5.6 minutes getting it. None of the industry consultants completing diary sheets indicated that they asked for help on any of the days concerned. Clearly, senior staff were much more independent in the use of their skills.

(d) Milestone Reviews and Monitoring

The basic points at which individuals had their work evaluated were milestone reviews conducted at the end of project stages, where individuals knew they could be called to account for the quality of their work.

In addition to the reviews conducted with the client at the conclusion of the feasibility stage, and after the production of the FSD, a series of formal internal reviews of progress were conducted during the course of a customisation project. Internal reviews took the form of a formal meeting chaired by the Production manager in which all those involved in a project examined the quality of what had been done so far. The agenda was compiled on the basis of checklists in the SDM.

Between review points, individuals were monitored formally at weekly intervals only. Industry consultants reported direct to the Production manager. Otherwise, he largely trusted them to their own devices. He would chat with them about progress if they were working in the office.

For the work undertaken by various levels of 'programming' staff, weekly reports had to be made to one of the project managers. Project managers also gathered data informally from their staff on a continuous basis, but this was primarily concerned with identifying technical problems rather than enforcing performance norms. Such informal monitoring was bound up with a supportive 'professional' style of supervision, in which help was also offered in dealing with technical complexities. Table 10.4 suggests that even informal reporting for 'programming' staff

occurred less than daily.

Table 10.4: Informal Reporting for Programming Staff.

	N	Av.no of times per day	Av. total time per day (min)
CSD-level	3	0.5	14.9
Programming-level	4	0.7	10.6

10.2.6 Low-Level Coordination

The system of action planning itself was a basic mechanism for coordinating the work of direct producers engaged on different stages of a customisation project. Action plans were however supplemented by further mechanisms in achieving lower-level coordination between producers.

(a) Standardisation of Skills

First, the SDM set out the sort of outputs to be expected from each successive stage of a project. It functioned as a form of standardisation of skills, by which, for example, a Technical Consultant knew what form functional specifications would be handed to him or her by an industry consultant. Second, the Jackson structured programming approach made it easier for one programmer to understand the work of another. Finally, programming staff were expected to work within the requirements of 'standards' or conventions needed to ensure the consistent

functioning of a Packset system even when it had been customised. These took the form of conventions for program structure, and for naming variables set down in manuals for each Packset module. These standards can be seen as a further aspect of the standardisation of skills.

(b) Direct Contact

The level of novelty or unpredictability of technical problems encountered appears to have been such that standardisation of skills was not adequate for eliminating all need for direct contact between software producers to clarify ambiguities. It was common for staff engaged on related modules of a large system to work in direct contact to make sure that the interface between the two modules was programmed consistently. Table 10.5 shows data from diary sheets indicating the average number of times a day and the average total time a day spent by staff at various levels discussing interfaces with colleagues and with their project managers.

Programmers and staff engaged on CSD-level work spent considerably more time discussing interfaces with their colleagues than industry consultants. CSD-level staff were much more likely to sort issues out by direct contact with colleagues, whilst program-level staff appeared to go through their project manager as much as directly.

Table 10.5: Time Spent discussing interfaces

		With Supervisors		With Colleagues	
	N	No. of contacts per day	Total time (min) per day	No. of contacts per day	Total time (min) per day
Programming Staff	4	0.4	3.3	0.7	5.1
CSD-level Staff	3	0.05	2.9	0.1	8.0
Industry Consultants	3	0.06	0.6	0.1	2.2
Project Managers	1	0.3	1.7	3.7	124

10.2.7 Adjustments to Plans

The Production Manager felt that meeting targets on scheduled projects was not on the whole a problem. In three years in his position, he estimated that as many projects had underrun their estimates as overrun. When a particular activity within a project overran, it was more than likely compensated by another that took less resources than estimated. However the fact that neither the complexity to be encountered in a task, nor the ability of a particular member of staff to deal with it could be predicted with precision, meant that slippage was always possible and the progress of projects had to be monitored continuously so that some form of rescheduling could be undertaken as early as possible.

Slippage was on the whole a more serious issue for CSD work and the subsequent 'Build Stage'. Given that the feasibility studies and FSD work were each estimated individually, and were also small compared to the total cost of a software licence, hardware, software customisation and installation consultancy, if industry consultants did overrun estimates there was generally little problem in absorbing the extra spend in the slack built into the overall project estimate. Alternatively the consultant could persuade the user management to pay more on the grounds that unanticipated complexities had been discovered.

However, industry consultants were expected not to overrun, partly because they might be required on another project and because overrunning at an early stage was not good for encouraging the client's trust. The Production manager reported that overruns were 'very rare' for industry consultants, because they estimated their own work, and because they usually worked with application areas and industries in which they had a lot of experience. One factor in the way Sysco successfully controlled installation work was undoubtedly the use of its own consultants on-site on a fairly continuous basis to make sure that the work of preparing new data handling procedures was accomplished.

A second reason why monitoring and rescheduling of activities was seen as necessary was the Production Department's responsibility for dealing with calls from clients for systems maintenance and modification. According to the Production Manager, this was a much greater source of uncertainty in managing the Department than slippage within scheduled projects. Large 'emergency projects' required a re-allocation of personnel between projects, or drafting in of extra resources. For this, the Production Manager considered that he needed up-to-date information on progress of all current projects, so that he knew which were slightly behind or slightly ahead of schedule. He could then decide if resources could be freed up from scheduled projects or whether extra resources would have to be brought in from elsewhere in the Group.

Unpredictability of the demand for work, combined with a certain degree of technical unpredictability thus produced the need for vertical information gathering and periodic replanning.

(a) Vertical Information Systems

The weekly progress reports from programming staff to the project managers formed a basis for running re-estimates to be agreed between these managers and each member of their staff as to how much time each block of work allocated still required. The Production Manager felt that programmers were often inclined to suggest re-estimates that were over-optimistic, when unforeseen technical complexity was encountered. They were aware that they would be evaluated for promotion in terms of their productivity as well as the quality of their work, and were often reluctant to accept that they could not meet a first estimate. In such situations, the project manager reminded the programmer of all the aspects still remaining of the activity, and made sure they were taken account of in a more realistic estimate.

Project managers used data on progress gathered in their informal contacts with their staff to report informally to the Production manager. They passed on 'soft' information on technical problems and slippage as they became evident. They also held formal weekly meetings with the Production manager to review progress on each project.

(b) Adjustments to Plans

If it was clear that within a project one staff member was running over and another was running well within the estimate for his or her activity, the project manager would switch some blocks of activity between the two. If the project as a whole was running behind, it would be reported to the Production Manager. If overall slippage occurred in the CSD or Build Stage of a project, or if extra staff resources needed to be made available

for an 'emergency' maintenance or modification project, the Production Manager had to adjust action plans. He identified the following options:

- The start date for the next project for the staff concerned could be postponed by negotiating with the client. The extra spend on the current project would be met from the general surplus generated on all projects. Rescheduling delivery of one project and the start of another was however something the Production Manager was very reluctant to do, because of ramifications in terms of Sysco's credibility.

- Team members could be added to a project with slippage or put on an 'emergency project' from a project which was proceeding within its estimate, or from 'internal work. This adjustment mechanism was thus based on the slack resources built into action plans.

- The Production Manager also had the authority to bring in extra team members from other parts of Sysgroup, for a particular project, provided their costs could be met from revenue for that project (for an emergency project, for exaxmple), or from the reserves of the department as a whole. In general this occurred when an 'emergency' project required expertise in a particular Packset module and the staff possessing it within Sysco were occupied on another vital project.

10.3 COMMITMENT TO PROJECT TARGETS

The Production manager and project managers were together able to identify the following mechanisms by which the typical professional technical work goals of direct producers were mobilised towards a commitment to project goals.

(a) Selection and Socialisation

It was vital to select staff with the right technical potential and attitude. According to one industry consultant, in a larger and less specialised Sysgroup company, 'bad programmers' were 'lost'. Project managers shed them from development projects as early as possible and put them on maintenance work. They were not given promotion and generally encouraged to leave the organisation. He felt that only the better ones got to Sysco. It would appear that staff at Sysco were screened for their commitment to work as well as technical competence.

(b) Responsibility

Managers considered it important that producers should be given extended tasks which utilised or extended their expertise, and which permitted them discretion and initiative. We have seen that tasks of larger spans were allocated to more senior staff, in recognition of their greater capacity to take responsibility.

(c) 'Seeing a Project Through'

It was also seen as important to give staff the chance to see a project through, in particular in the sense of involvement in installation work. For industry consultants this was part of the normal organisation of work, that they were responsible for both requirements analysis and installation. For programming staff, it meant arranging for junior project staff to have the opportunity to assist industry consultants on installation work.

(d) Skill Development, Pay Rises and Promotion

Managers saw keeping people satisified with pay and promotion progression as vital for securing their motivation. The system of staff performance reviews was also important for encouraging

commitment in that it provided an opportunity for staff to have their development needs assessed and planned for. Further, the type of development encouraged was in terms of the development of skill and the ability to work responsibly and with motivation. According to managers, promotion and pay rises were awarded on criteria of staff possessing expertise and the ability to use it without close supervision.

Up to the level of consultant, performance reviews were held every six months, with annual reviews for more senior people. Performance was assessed in advance by the relevant project manager and set out on an appraisal form under headings of 'Job knowledge', 'Quality of work', 'Quantity of work', 'Adaptability', 'Reliability', 'Initiative' and 'Cooperation'. At the review itself, the staff member, project manager and production manager would discuss the assessment and identify areas of development, in terms of training and new areas of work the employee was ready for.

Consistent with this picture, Table 10.6 shows that, across the whole sample, those interviewed saw 'problem solving ability' and 'understanding of the application area' as the most significant criteria by which they were judged for promotion. It also shows how the promotion system influenced people at all levels to gain expertise in the 'client-oriented' areas of functional analysis and installation, rather than concentrating on the more abstract technical aspects of system design and programming. In addition, industry consultants and project managers were aware of being evaluated for their management ability. Further, this table confirms the importance attached to procedures and standards within Sysco. Across the whole sample, 'performing according to procedures' was seen as being as important a criteria in promotion assessments as 'good technical knowledge'.

Interviews with direct producers provided evidence that a considerable amount of formal training was actually received. Of seven program-level staff interviewed, two had received training

Table 10.6: Perceived Importance of Various Criteria for Promotion.

Mean scores for importance on a scale of 1 to 9, broken down according to position in the division of labour and overall.

	Prog.	System Designer	Industry Con-sultant	Project Manager	Overall
N	7	3	6	3	19
Problem Solving ability:	7.2	8.0	8.3	8.7	7.9 (s.d.=1.2)
Performing according to procedures:	6.7	6.0	5.6	5.0	6.0 (s.d.=1.7)
Good technical knowledge:	6.9	6.7	5.3	5.3	6.1 (s.d.=1.6)
Understanding of application area:	7.1	7.7	8.2	7.7	7.6 (s.d.=1.3)
Management ability:	5.1	7.0	8.3	8.7	7.0 ** (s.d.=1.8)

at Sysco in basic programming techniques, and another two in aspects of particular application areas. Two of the three CSD-level staff had been trained in programming techniques, two in 'effective communications', one in systems design, one in data base design, one in sales and one in project management. Four of six industry consultants had received training in sales, three in project management, and two in database design.

10.4 ISSUES AND TENSIONS

10.4.1 Issues Perceived by Managers

This section considers data obtained from managers at various levels concerning what they saw as issues in the functioning of the company.

(a) Senior Management: Product Strategy

According to the Marketing and Production Managers, the strategic issues which preoccupied their superiors i.e. the Director of Manufacturing and the Managing Director of Sysco were those of marketing strategy, product enhancement and development. Senior management saw maintaining competitive advantage in terms of chosing 'deep' product niches where Sysco could achieve differentiation of its products from those of its competitors.

This meant that senior management were not primarily concerned with performance in terms of simple productivity of software producers, or with changes in techniques, work organisation or the system of controlling work in order to intensify it. Their attention was primarily on the rapidly changing market place, on sales figures, and on what sort of product developments might be called for. The SDM was regarded as a tried and tested approach to managing all types of development projects, and as long as

project managers and industry consultants were trained in it, and high quality staff continued to be recruited or developed, the execution of customisation projects could be left to the Production Department and its existing ways of working.

(b) The Production Manager: Finding People With Skills

The Production Manager perceived his main problems as twofold.

First, he identified the complexity of balancing the forward workplan to take account of unpredictable client needs for maintenance and modification work. He felt there was no obvious solution to this, and also appreciated it as a challenge.

Second, the main constraint he saw in achieving this was the availability of industry consultants skilled in particular application areas, Packset packages, and industries. This raised the priority of developing staff with application area and product expertise as well as design and programming skills. Here the organisation faced a dilemma, also voiced by the Group Personnel Manager, that it was difficult to spare the time of people with these skills to train others.

(c) Project Managers and Industry Consultants: Getting System Requirements Right

Project managers and industry consultants typically saw defining the functions to be performed by a system as the most difficult part of its development and the most crucial determinant of the quality of the product. They considered the autonomous use of their non-codified expertise as the best approach, and tended to see the future viability of the organisation in terms of the presence of these skills.

Several project managers and industry consultants were also concerned with the quality of work done by programming staff.

They referred to a national shortage of experienced COBOL programmers, to difficulties in keeping good programmers, and to a recent fall in overall standards in these grades. They focussed on issues of experience and quality of work, rather than those of productivity per se.

10.4.2 Issues For Software Producers

This section presents data from interviews with 19 staff within the Production department, concerning their experiences of work, the tensions they experienced and senses in which they felt conflicts with organisational goals. In the tables that follow, the sample is broken down into four groups, according to position in the division of labour: project managers, industry consultants, more experienced programming staff responsible for CSD-level work, and less experienced programmers engaged on program-level design and coding.

Tables 10.7 and 10.8 show that on average achievement of work-related goals was moderate. Almost all industry consultants and project managers gave higher than 'satisfactory' ratings for the extents to which their jobs measured up to what they wanted and their most important goals were being met. One industry consultant, however, found both less than satisactory. Three out of seven program-level staff and one out of three CSD-level staff experienced less than 'satisfactory' goal achievement. The fact that the majority found themselves achieving their work-related goals to a fair extent suggests an absence of serious conflict between individual and organisational goals. The experience of the minority however indicates tensions, in particular between what more junior staff aspired to and what the organisation could offer. In order to understand these tensions, as well as what made senior staff more fulfilled, we now examine data on various aspects of experiences of work.

Views on overall goal satisfaction

Table 10.7: Achieving Most Important Work Related Goals

		Prog.	System Designer	Industry Con-sultant	Project Manager	Overall
N		7	3	6	3	19
Work achieving most important personal goals:						
Not at all	1					
	2	2	1	1		
	3	1				
Satisfactorily	4	1	1	2	1	
	5	2		2		
	6		1	1	2	
Perfectly	7	1				
Average		4.0	4.0	4.3	5.3	4.3 (s.d.=1.6)

Table 10.8: Extent to which job measures up to what is desired

		Prog.	System Designer	Industry Con-sultant	Project Manager	Overall
N		7	3	6	3	19
Job measures up to what you want:						
Not at all	1					
	2	1	1			
	3	2				
Satisfactorily	4	4		4		
	5	1	1	3	1	
	6			1	1	
Perfectly	7	1	1		1	
Average		4.0	4.7	4.8	6.0	4.7 (s.d.=1.6)

Table 10.9: Experience of utilisation of skills

	Prog.	System Designer	Industry Con-sultant	Project Manager	Overall
N	7	3	6	3	19
Time demands: (Scale: 1='ext. light' to 7='ext heavy')	4.7	5.3	5.5	6.7	5.4 (s.d.=1.3)
Technical demands: (Scale: 1='ext. light' to 7='ext heavy')	4.1	4.3	4.7	5.3	4.5 (s.d.=1.4)
% time below technical capacity:	30	27	30	20	28 (s.d.=1.8)
% time trivial or routine:	20	10	18	7	16 (s.d.=1.6)
% time show what you can do:	37	43	65	80	54 (s.d.=2.9)

Table 10.10: Experience of autonomy in carrying out tasks:

	Prog.	System Designer	Industry Con-sultant	Project Manager	Overall
N	7	3	6	3	19
% time using own approach:	58	70	70	80	68 (s.d.=2.4)
% time making all important decisions:	37	70	78	70	61 * (s.d.=2.7)
% time specs already agreed by others:	71	17	40	13	42 * (s.d.=3.2)
% time every step has to be approved:	34	20	13	13	22 * (s.d.=2.2)
% time given detailed instructions:	35	10	20	13	22 (s.d.=1.4)

* = significant at 0.02 level

Table 10.11: Impact and Visibility Of Contributions

	Prog.	System Designer	Industry Con-sultant	Project Manager	Overall
N	7	3	6	3	19
% time involv. in business decisions:	0	3	15	10	7 * (s.d.=1.0)
% time important contrib:	37	53	52	50	47 (s.d.=2.2)
% time contr. to business success:	33	47	45	87	48 (s.d.=3.1)
% time work of special interest to management:	32	47	31	31	34 (s.d.=2.4)
% time higher mgt doesn't know what ind. is doing:	32	10	36	0	25 (s.d.=3.3)

Table 10.12: View on Salary

	Prog.	System Designer	Industry Con-sultant	Project Manager	Overall
N	7	3	6	3	19
View on salary, relative to comparable firms: (Scale: 1='much lower' to 7='much higher')	3.0	4.0	3.2	4.0	3.3 (s.d.=0.9)
View on salary plus perks, relative to comparable firms: (Scale: 1='much lower' to 7='much higher')	3.0	3.0	3.4	3.5	3.2 (s.d.=1.0)

Table 10.13: Satisfaction with Promotion Prospects

Mean scores for satisfaction on a scale of 1 to 7 (4= 'about right') broken down according to position in the division of labour, and overall.

	Prog.	System Designer	Industry Con-sultant	Project Manager	Overall
N	7	3	6	3	19
View on probable timing of next promotion: (1=much too soon, 7=much too long)	4.8	4.3	4.2	4.3	4.5 (s.d.=0.6)
View on probable size of next promotion: (1=much too small, 7=much too big)	4.4	4.0	3.8	4.3	4.1 (s.d.=0.7)

Table 10.14: Attitudes to Learning Advanced Software Engineering

	Prog.	System Designer	Industry Con-sultant	Project Manager	Overall
N	7	3	6	3	19
View on learning new techniques, on a scale of 1 to 7: (1= 'very opposed', 4= 'indifferent', 7= 'very keen')	6.2	6.5	5.0	7.0	5.9 (s.d.=1.9)

(a) Skill Utilisation

Table 10.9 shows that time demands were perceived as moderately heavy on average, with no significant differences between groups. A number of more junior programmers referred to discussions with and questions from their project managers as to why assignments had taken as long as they did. This implies that junior staff were not completely trusted to work to norms of quality and productivity, but had to be reminded of the standards were in these respects. Apart from this, however, there was no other evidence in the interviews conducted of tensions over time demands or levels of effort.

Both technical demands and the proportion of time producers felt they were 'given the chance to show what they could do' were more moderate, again with no significant differences between groups. One industry consultant spoke of the pleasure it gave him when he met a tight deadline by designing a system creatively. Like the majority of those interviewed, he clearly felt well-utilised in terms of both time and technical demands, and was committed to working at a certain level of productivity provided that it went together with a feeling that he was achieving something through the use of his skills. However, six out of nine programming staff interviewed felt that they were given the opportunity to show what they could do for less than half their time. Given the general relationship established in Chapter 7 between goal achievement and skill utilisation, this suggests one source of tension experienced by some programmers, working at both CSD-level and program-level.

(b) Control Over the Approach to Work

Table 10.10 suggests that most staff felt they were 'using their own approach' and 'making all the important decisions' in their work for most of the time. Staff performing program-level design and coding felt they were making the most important decisions for a significantly lower proportion of time than other groups.

Three out of seven felt they were making the most important decisions in their own area for less than half the time. This group also felt that the specifications to which they worked had been agreed by others for a higher proportion of the time - over two-thirds - and that they were given detailed instructions more than the others. All those interviewed, however, felt they were given detailed instructions for less than half the time. The overall picture is one of staff enjoying autonomy in how they worked, although more junior staff felt themselves more circumscribed by specifications and detailed instructions. Not even the most junior staff, however experienced themselves as having to seek approval for everything they did. As one programmer put it:

> "For major steps, I have to refer to my supervisor before proceeding. Provided he feels I understand what I'm doing, how I go about it is up to me".

(c) Perceptions of Structured Techniques

Qualitative data obtained during interviews concerning perceptions of structured design and programming standards further illuminate the extent to which staff at different levels controlled their approach to their work.

Program-design level staff were unanimous in valuing use of structured programming standards. They found standards made it easier to communicate with each other. None of these staff expressed any problems with the requirements of the company standards, which suggests that they were not a threat to the autonomy programmers sought. They accepted these standards as a framework within which they worked. The only problem programmers named with respect to standards was that they were often not applied evenly, largely because some programmers had not been trained in them. Several referred to a resulting 'double standard' within the organisation.

CSD-level staff were also enthusiastic about the use of Jackson structured standards in design and programming. Again, there did not appear to be any fundamental tension in practice between the use of standards and control over work. However, more than programmers, they tended to stress that standards should be seen as guides for how to proceed and not as inviolable. If applied too rigorously, they could "cramp a designer's style." The current degree of use and management attitude to standards was, however, "about right".

For both industry consultants and project managers, basic structured standards were seen as important for designing and coding packages that were easily modifiable. The type of structuring currently practised was seen as generally appropriate, although there was some ambiguity about the desirability of enforcing these structured standards strictly. On the one hand there were concerns that some programmers did not know how to use the standards since a need to keep them on fee-earning work had meant they had not been able to have the necessary training. One industry consultant felt that project managers did not enforce the standards adequately. They tended to want to be informal and relied perhaps too heavily on the expertise of their staff to produce good quality code. He called this the 'small company syndrome'. On the other hand, industry consultants and project managers were alike emphatic that standards should not be used to restrict the highly competent programmer who knew when to depart from them beneficially.

Industry consultants' and project managers' views on the use of formal methods and standards for requirements analysis and high-level design work were also generally favourable to the status quo. They saw the SDM as a guideline or framework within which to proceed, which could be used to identify the basic steps to be taken in designing and contracting for a system. There was considerable resistance to the idea of a detailed methodology which would specify how to go about each step. As long as the result was a design that was structured according to the Jackson rules, the feeling was that consultants and analysts should be

permitted to chose their own way of getting there. This data corroborates the data from Table 10.10, that this level of staff worked with a very high degree of control over techniques.

(d) Visibility and Impact Within The Organisation

Table 10.11 shows the basically moderate extent to which staff in the Production Department felt they were making an individual contribution and having an impact on business success. The averages for the whole sample in terms of the proportions of time for which they felt they were making an 'important contribution' and 'contributing to business success' were both around 50%, with no significant differences between groups. These moderate feelings of contributing suggest moderate satisfaction of these particular work goals. Seven staff, across all groups, felt, however, that they were contributing to business success for less than half the time. Some staff, then, experienced tension with respect to the impact of their work on achieving organisational goals.

(e) Pay and Promotion

Table 10.12 shows that those interviewed were broadly satisfied with their salaries and fringe benefits in comparison to what they could be earning elsewhere, although they considered their rewards on the low side. Sysco appears to have been a 'competitive' rather than a 'top payer'.

Table 10.13 shows that staff at all levels were on average satisfied with their promotion prospects. There was however evidence of tensions in this respect.

Industry consultants at senior and principle consultant level anticipated being promoted into sales or product management, but recognised they might have to wait a few years for a vacancy to become available. Some resented not being able to continue to

advance within the organisation as a programming manager or an industry consultant, but having to seek promotion as a manager. They felt that they were forced into management if they wanted to rise above a certain level of earnings, when they would really be happier staying as a "high powered" consultant.

One also felt that he had been forced on to the "user analysis, installation consultancy" side from being a programming team manager on a large 'special project'. Although he would have been happier to continue to as a high status systems designer/programmer, responsible for managing the work of others, he realised there were no further opportunities in this direction. As a programming manager, he felt his pay was limited to twice that of a trainee programmer, whilst he estimated his productivity to be five times greater. The lack of a "fully professional" programmer grade meant that his contribution on the programming side could not be recognised.

This view contrasted with that of a technical consultant who was happy to move into managment or industry consultancy in a few years, but saw it as important in the shorter term to gain "a sound technical basis", which she felt many managers in data processing lacked.

10.4.3 The Nature of Tensions and Conflicts

The moderately positive levels of goal achievement for producers and the basic satisfaction of managers with standards of individual performance of all grades except junior programmers, indicate a lack of explicit conflict between the goals of individuals and the system of organisation and control. Desires for work that utilised skills, and for control over decision-making in work were met to a large extent.

Processes of socialisation and careful estimating in product markets where prices could be related to costs seem to have mostly eliminated overt conflict over labour productivity.

Targets could be set so as to present a challenge and bring out the commitment to work, but not so tightly as to damage job satisfaction and undermine internalised commitment. The system of rewarding the development of skill and commitment by promotion to positions of greater responsibility appears to have been significant in reproducing commitment to norms of quality and productivity in meeting project targets.

Norms seem to have been explicitly stated only by supervisors of junior programmers. The concerns of some project managers about falling standards of experience and quality of work amongst programmers do not suggest that there was even here any great conflict over levels of effort. Managers were merely concerned that staff were not developed to the right level.

The use of specifications and standards seems also to have been largely accepted by direct producers, since on the whole they still left scope for challenge and use of discretion. There is no evidence that producers saw specifications as too 'commercial' in nature, or as compromised from a technical standpoint. Rather, they saw constraints of time and or cost as boundaries within which they had to use ingenuity.

There were two main sites of tension, however. First, some junior programming staff saw the program-level design, coding and testing tasks allocated to them as limited in challenge, scope, and the level of decision-making involved. They were not given tasks of any great responsibility. It appears that the system of developing experienced and responsible staff from a pool of younger staff given less responsible work led to some resentment at lower levels. However, it seems likely that junior staff were willing to undergo such an apprenticeship because of the promise of more responsible and challenging work in the future, either within Sysco or elsewhere.

Whilst their tasks were circumscribed by specifications and structured standards, these do not appear in themselves to have been a focus of dissatisfaction. Rather, frustrations concerning

uneven documentation and adherence to standards were named. Whilst some junior staff sometimes found their limited involvement in projects less than satisfying, they saw structured standards as a help in their work, rather than an unwelcome level of prescription. There is a suggestion that the preference for autonomous operation by project leaders and industry consultants, their recognition of standards on the one hand and the belief that they could be flouted on the other, actually undermined the degree to which their junior colleagues could take responsibility for substantial tasks. Junior staff found learning about package products hampered by lack of proper documentation. They tended to favour more rigid enforcement of standards because they would improve their understanding and control of their work.

Second, we have seen that the system of recruiting managerial staff from the ranks of industry consultants and industry consultants from system designers (technical consultants) also led to frustrations amongst senior staff. Some felt that greater seniority and responsibility was tied to moving away from the more technical end of software production.

The issue here would appear to be the equation of certain stages in performing customisation projects with seniority. Industry consultants or technical consultants who wished to stay as such clearly considered that their current work offered sufficient challenge and responsibility for them to build their career within it. They resented the assumption that, as they became recognised as more competent and trusted by their organisation, they were expected to move on to other work.

10.5 STRATEGIES FOR CHANGE AND POSSIBLE OUTCOMES

This section discusses management initiatives contemplated in the context of the issues described in the previous section, and their possible impacts on the organisation of work and the nature

of career structures.

(a) New Products

Within Sysgroup generally, the main senior management initiatives were at the level of frequent re-drawing of the boundaries of the three Group companies. The rapidity with which new goods and services appeared on the market meant that the spheres of operation of the different companies and profit centres within them were rarely completely distinct or stable.

Whilst the companies were allowed considerable autonomy in deciding how they should sell their products to meet their profit targets, Group Directors intervened on an almost annual basis to redraw the boundaries between companies. The rationale for this was the perception that new products needed to be developed and marketed in new combinations.

The result was a considerable overlap in areas of expertise in the different companies, as well as movement of staff between them. This meant that software producers were likely to find themselves transferred from one company to another from time to time, but this did not appear to cause any concern amongst them. The main concern was with advancement, no matter where it was within Sysgroup, or indeed elsewhere.

(b) Developing Industry Consultant Skills

The main area of strategy for the future identified by the production manager has already been referred to. He was concerned with developing more staff with the range of skills and experience necessary for 'industry consultant' work, so that this class of direct producer could continue to work autonomously. This was clearly designed to perpetuate the existing model of work organisation, and was not at odds with the goals or ideas of work organisation of most staff.

The only exceptions would appear to be those programmers who wished to remain doing mostly the more technical computer system design and programming management work, rather than moving into the more application area-oriented 'industry-consultant' realm. The emphasis on 'industry consultant ' and project manager-type skills placed by higher management would, however, suggest that the emergence of a parallel 'technical' job ladder within Sysco is unlikely.

(c) More Rigorous Structured Techniques

The main advocates for the introduction of more rigorous structured techiques appear to have been those most directly involved in controlling the course of customisation projects. One interpretation of this is that the control of work to meet estimates was delegated to Project Managers and industry consultants to such an extent that only they saw it as a problem in need of new approaches.

Many industry consultants were not convinced that, given the difficulties of recruiting experienced programmers, quality standards could be maintained through the non-codified expertise of staff alone. They advocated training in rigorous structured programming and design, and stricter enforcement of standards at design reviews. At the same time, they felt that such techniques would restrict the invention and diminish the job satisfaction of more competent programmers. A similar advocacy of more rigorous structured techniques by project managers is also an interesting contrast to the perceptions of industry consultants and programming staff that project managers often did not enforce the application of existing structured techniques and standards. One interpretation is that project managers preferred not to use techniques too strictly, but felt they should do, because structure was virtue. Another possibility is that they advocated more rigorous structured techniques only as a second best option, given that staff of the right experience were not available.

The views of programming staff working at CSD-level and program - level concerning new development methodologies were as follows. On the one hand, several felt that management were backward in introducing new automated development tools, such as program generators, which could produce code in COBOL from precisely-notated program designs. This perception that the Company was not interested in the latest development techniques tallies with the preoccupation with the product market as opposed to the software production process noted on the part of senior management. At the same time, programmers were concerned that the introduction of more rigorous development methodologies would restrict their autonomy and make their work more boring. Table 10.14 shows that they were nonetheless keen to learn more advanced software engineering methodologies, so as to raise their standing in the labour market.

Whether or not project managers and industry consultants continued to press for more rigorous programming methodologies would appear to depend on the sort of pressures they felt under to complete projects to high standards and tight time targets, and the sort of recruitment strategy adopted at Group and company level. A decision to go for lower rather than higher entry salaries would strengthen arguments for more rigorous techniques. A problem that this sort of approach might encounter at some future point could be a difficulty in training competent industry consultants, able to analyse complex situations and exercise discretion, from programmers schooled in the application of rigid techniques to tasks with low levels of responsibility. The problem would be how to get people to make the transition from a system of labour control emphasising 'bureaucratic control' to one emphasising 'professional control'. This would then open up a possibility of deciding to recruit a different class of junior software producer, who might be groomed from the beginning towards taking more responsibility. There was, however, no immediate sign of such a segmentation of job structures at the time of fieldwork.

With respect to their own 'client-oriented' responsibilities i.e. feasibility studies, FSDs and installation work, industry consultants saw no such role for structured methodologies, beyond the guidelines of the SDM. They shared the perspective of their superiors that the best way to assure quality in the specification and installation of systems was through the expertise of the staff engaged to do this.

CHAPTER 11

INFOSERVICES DIVISION

SUMMARY: INFOSERVICES

1. BUSINESS STRATEGY AND TASK ENVIRONMENT

Market: Selling dataprocessing software systems to divisions of a corporation, in competition with outside suppliers.
Businesss Strategy: Differentiation on basis of quality and innovation.
Environmental Strategy: Developing co-operative relations with 'customers'.
Task environment: Diverse; complex commercially and technically; unpredictable commercially in all areas, technically in some.

2. ORGANISATION, JOB STRUCTURES, CONTROL AND CO-ORDINATION

Organisation: Main software development department dept., supported by service groups containing software staff responsible for planning, & specialist areas of development. Main software dept. divided into groups responsible for different application areas. Each group has seperate analyst and programmer teams, although one also has analyst-programmer teams.
Career Structures: Separate career structures in systems analysis, general programming, & specialist programming. Higher educational attainment required for entrants to specialist work. Within each career ladder, promotion with experience to more responsible work, with no vertical segmentation. No internal labour market. Managers & 'staff' recruited from amongst producers.
Overall Control and co-ordination: Central action plans for projects laid by senior management, to tailor activities to combined priorities of client divisions. Detailed low-level action plans drawn up on basis of specifications and estimates compiled by team leaders, refined at several points during development process. Action plans also specify tasks for staff in client divisions.
Labour Control: Achieved through 'professional' skills and commitment in the context of targets and specifications, backed up by informal monitoring & milestone reviews & availability of help from team leaders & colleagues. For programmers only, quality assured through structured programming,
Low-level co-ordination: Standardisation of skills through structured programming used to co-ordinate development & maintenance staff. Direct lateral relations used for detailed co-ordination between producers.
Slippage and Adjustments: Information concerning technical problems & lack of progress by user staff collected 'vertically' through informal monitoring: plans adjusted by renegotiating projects or deadlines, or utilising slack resources.

3. MECHANISMS FOR PRODUCING COMMITMENT TO WORK

Selection. Socialisation through supportive supervision, but with some direct assertion of norms to junior staff. Providing job responsibility, skill development & training, pay raises & promotion. Letting producers 'see a project through'.

4. ISSUES AND TENSIONS

For Managers: Finding the 'right people'. Developing staff with project management skills. Nature of future services. Priorities & schedules. Obtaining accurate user specs. Maintaining user co-operation during development. Conflict between middle management autonomy and standardisation.
For Producers: Moderate skill utilisation. Those not yet using it fear structured programming will remove discretion. User departments employing own analysts remove discretion from Infoservices staff.

5. INITIATIVES FOR CHANGE AND LIKELY OUTCOMES

Initiatives: Diversification in application areas. Introduction of more formalised programming techniques and procedures, so as to separate maintenance work from development. Introduction of structured analysis to improve communication with user. Increased use of packages, bought in and customised.
Possible outcomes: Continuation of 'professional' control of labour, with separation out of maintenance work to training positions. Use of packages associated with shift away from specialist programmers and fewer development staff overall. Increasing importance of 'staff' roles for software staff.

11.1 BUSINESS STRATEGY AND TASK ENVIRONMENT

Infoservices Division was part of the Financial Directorate of Telecorp corporation, and fulfilled a corporate data processing function, serving 14 product divisions on 11 sites across England, Scotland and Wales, involved in the design and manufacture of a variety of electronic and telecommunications products. The Division handled three main classes of data processing application: accounting and general administration systems; manufacturing systems, such as materials requirements planning or production scheduling; and engineering systems for product design and development. In this case study we are concerned with the parts of Infoservices concerned with software services, i.e. developing or acquiring, installing and maintaining applications software systems. Other parts were concerned with running systems once installed, and with maintaining hardware and systems software.

Resource Providers and Their Expectations

Infoservices' business strategy was shaped first of all by the way it was funded by the corporate centre and operating divisions. Its annual budget was made up of two components. First, the bulk of the operating budget was obtained through fees from user divisions, charged for staff time spent on development of applications software systems, installing and maintaining them, for time spent on operating installed systems, and also for mainframe processing time used. Second, direct funding from the corporate budget was provided for purchase of central hardware, writing and maintenance of systems software needed to run applications on it, activities involved in planning for the development of information processing within the corporation as a whole, and the conduct of feasibility studies for new applications. The Division was treated as a cost centre rather than a profit centre. In practice, the basic criterion used by resource providers for judging Infoservices' performance was that

what it provided should be more cost effective than what was available through buying in data processing services from outside providers.

Business Strategy

Business strategy was also shaped by the perceived characteristics of Infoservices' business environment, in the sense of the nature of demand for its services, and the technology available for supplying them.

During the 1970's, like many corporate data processing departments, Infoservices had pursued economies of scale in providing mainly batch data processing services. The significance of hardware costs in this period led to an emphasis on centralised hardware, shared between many applications and divisions, and utilised to the maximum. Along with the centralisation of mainframe hardware went almost complete centralisation of Infoservices's 200 staff engaged in development and operations, to a North London site.

The economies of scale offered by centralised technology led to a natural monopoly and a stable environment for Infoservices in terms of the demand for its services. In terms of its technological environment, it had to deal only with incremental improvements in processing, storage and peripheral hardware, and in systems and application software, within the paradigm of centralised hardware and applications software developed in-house.

In the late 70's, mini-computers became available with relatively cheap standard software packages to run common administrative, accounting and manufacturing applications that were also easy to operate. This removed the established returns to scale and raised the possibility of user divisions or even departments within them having their own computing facilities for some applications. Perceptions that such a mode of provision was often

more cost-effective than the central mainframe approach, and that it would be possible for them to have their own self-contained data processing installation led many user divisions to demand their own mini-computer systems. Whilst most mini-systems were, in fact, ordered, installed and maintained by Infoservices staff, dissatisfaction with Infoservices' levels of charging led some user divisions to demand greater autonomy in the way they obtained data processing services. One or two decided to 'go it alone' and set up their own small data processing departments, selecting, running and maintaining mini-computer systems independently.

Facing what was becoming an essentially competitive environment, Infoservices management responded with a basic competitive strategy of differentiating their services on the basis of quality. They recognised that overall Telecorp was being run according to a cost-cutting strategy, which meant that both the corporate Board and General Managers of divisions were keen to impose limits on data processing budgets. They chose to argue the cost-effectiveness of their software services within present levels of expenditure, so as to retain their position as sole suppliers of data processing software services within the corporation. They were however prepared to allow the operation of mini-systems to be decentralised to user divisions. The goal of retaining their monopoly did not stem merely from a desire to protect the size of their organisation, their prestige and career prospects. They were also concerned that users 'going it alone' might install incompatible systems which would cause difficulties when future attempts might be made to unify and link information processing systems in different parts of the corporation.

Infoservices managers saw a number of ways in which it was important to differentiate their services from those of outside suppliers.

First, each year choices had to be made as to which new systems should be supplied and which existing systems enhanced. These choices had to reflect the current priorities of the corporation

as a whole as to where computerisation was needed most urgently, according to powerful figures within the overall organisation, in particular members of the main Board and general managers of large divisions. Infoservices management also considered it important to meet some information-processing needs within the Company which could emerge or change almost from month to month. An example of this was when changes in legislation affecting sickness benefit schemes or pensions meant that administrative systems had to undergo major changes at a few months notice. This meant re-assessing priorities fairly continuously over the course of the year.

Second, systems had to be specified and developed to cost, so as to be cost-effective in what they offered users, as well as competitive in absolute price terms.

Third, Infoservices management recognised the importance of making the most advanced forms of data processing software technology available to users in selected areas where it could enhance quality without increasing costs. There was a strategy of innovative differentiation in some application areas. In application areas where software packages were commercially available, Infoservices management encouraged its staff to offer user divisions support in acquiring their own mini-computers and packaged applications software, customising them if necessary. This was seen as often providing the best quality solution within the limits of user budgets. It was also a way of meeting the desires of users to 'go it alone' half way. By helping and advising users who wanted their own facilities, Infoservices could maintain some control over the computer systems they ended up with.

A further area of innovative differentiation was the development of 'user friendly' systems based on 'fourth generation' software languages. This allowed users in different divisions without programming skills to interrogate a database and request reports according to their own specifications, using a simple query language. It also included word processing and graphics

facilities.

Environmental Strategy

A major issue perceived by Infoservices management in pursuing differentiation of its services on the basis of their quality was propagating its own interpretation of what the provision of high quality data processing services amounted to. There was a potential clash between a particular division's short-term conception of its needs, and the view stemming from a long-term and corporate-wide perspective. User divisions needed convincing that it was worth spending more on the development of generic systems that would meet the needs of other divisions as well as their own, or on package customisations so as to incorporate features that would make them more compatible with other systems that might be required in the future, for example to provide corporate-wide strategic information processing. This was a problem given that user divisions were being encouraged to cut costs. Telecorp was run, according to the General Manager of Infoservices, "like a one year company."

Linked to this was a decision to influence the basic approach to information processing used at corporate and divisional level, in the direction of simplification. Rather than encouraging a division to make as much use as possible of computerised data processing, Infoservies was concerned to maximise the effectiveness of what could be provided by ensuring that computer systems were used as part of a simple and economical overall information processing system. Its general manager spoke of "a need to halve the amount of information being processed in the Company."

The mechanism adopted for environmental management was establishing co-operative relations with powerful corporate actors, through a system of Information Systems Review Boards involving representatives from user divisions. These served as channels for user demand to reach Infoservices, as well as for

Infoservices to communicate with users what it saw as being technically possible and preferable, and for some form of agreement as to schedules and priorities to be negotiated.

Each division of Telecorp had a Functional Information Systems Review Board (FISRB), consisting of senior managers concerned with information processing. These Boards reviewed requirements on an annual basis, and proposed new developments. There was also a Company Information Systems Review Board (CISRB), on which the Infoservices general manager and the Finance Director - to whom he reported - sat, together with directors with line responsibilities for user divisions and their general managers. Endorsement from this Board was required for the development of systems costing more than a certain amount. This Board was seen by the Infoservices General Manager as an important forum for raising awareness as to the sort of generic systems that should be developed, and for getting sanction to draw up plans for them.

Task Environment

The task environment facing the parts of Infoservices concerned with supplying software services can be summarised as follows. It was diverse in terms of the sort of demand that had to be met. Infoservices had to cater for a wide range of application areas. It was complex commercially, in terms of knowledge needed to assess competing priorities, and draw up plans, and to analyse user requirements once projects had been scheduled. It was also complex technically, although the degree and type of complexity varied from area to area. Infoservices took on projects with a great deal of variety in scope and technical complexity, from the mere installation of a standard package to a large-scale bespoke development. It had to have not only expertise in developing data processing software, but also knowledge of packages available commercially across a wide spectrum of application areas.

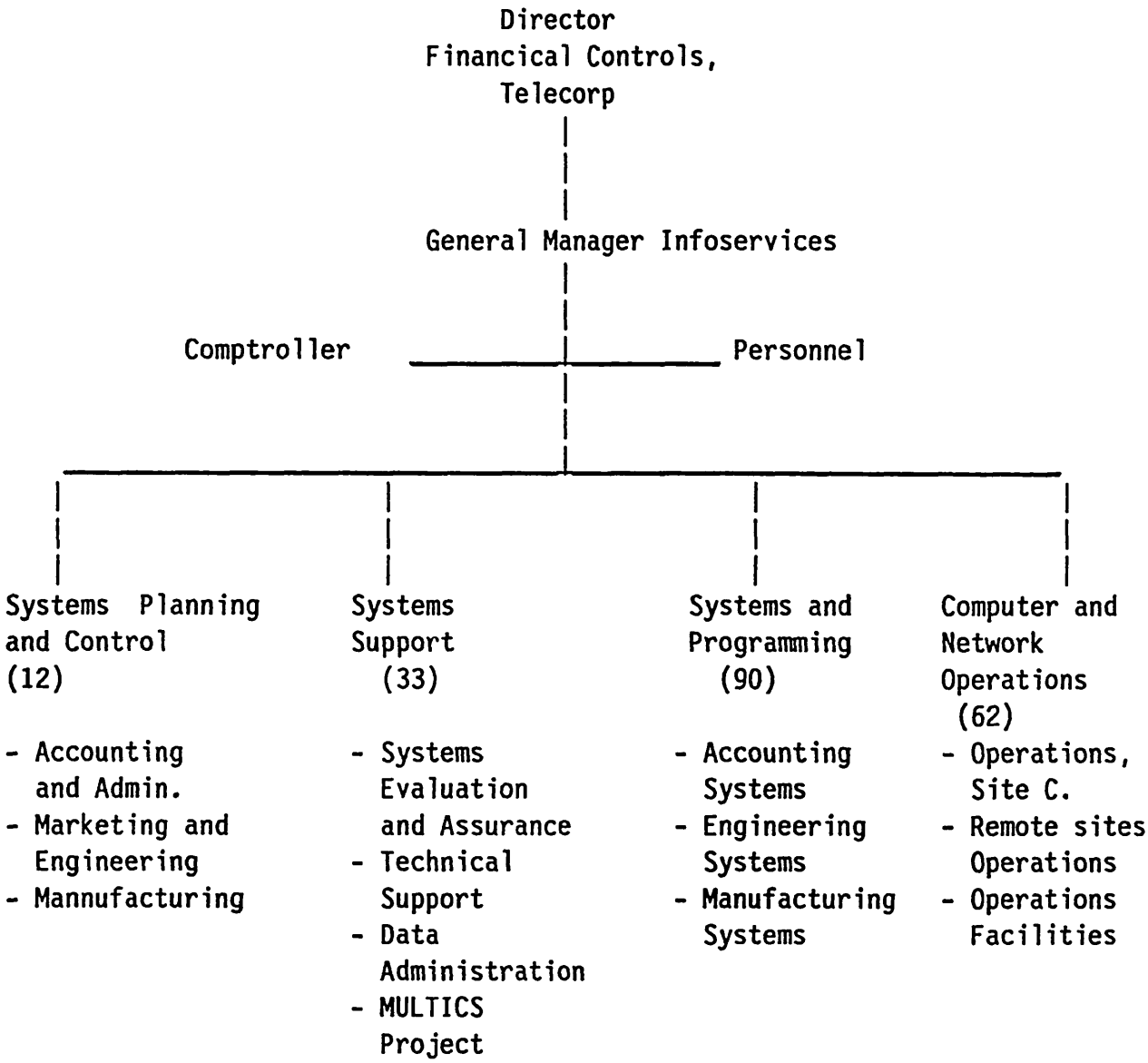
In terms of unpredictability, the main source seems to have been 'commercial'. Infoservices management saw responding to unpredictable user requests as important for maintaining faith with user divisions. There was also technical unpredictability in some areas. Keeping up with user preferences meant keeping up to date with new developments in software and hardware technology, which users might hear about and otherwise decide to buy in from outside. In addition to applications software packages running on mini computers, users were also coming to expect applications which ran off integrated company databases, spanning several applications areas, and systems developed using '4th generation' languages. There were however considerable areas of activity in which technical aspects were much more stable. Infoservices continued to develop new applications systems and enhancements to existing ones using the very traditional technology of code written in COBOL, running on a mini or mainframe computer.

11.2 ORGANISATION, CAREER STRUCTURES, CONTROL AND COORDINATION

11.2.1 Organisational Structure and Departmental Tasks

Figure 11.1 shows the four principle departments and the basic functional groups within them. Systems and Programming (S & P) was responsible for the bulk of software production and maintenance. Computer and Network Operations was responsible for operating these systems once installed. Software staff were also employed in Systems Planning and Control, which provided supporting services to S & P, and in Systems Support which

Figure 11.1: Organisational Structure of Infoservices



provided a variety of support services to both S & P and Operations. The separation out of these supporting services can be seen as a response to the increasing complexity and unpredictability of the task environment.

Systems Support had been built up around a **Technical Support** group, concerned with maintaining hardware and its systems, timesharing and communications software. This group was staffed mainly by computer science graduates, whose jobs formed a cluster that was distinct from those in production of applications software. Two of the other groups in the department were concerned with specialised and complex areas of software production which it was considered worthwhile separating out from S & P because of particular skills required. One group had used a fourth generation language to develop a set of 'user friendly' applications which non-specialist user staff could use at a terminal. The second group maintained databases used by applications systems, and designed and programmed new sections of databases required by new systems. This **Data Administration** group was responsible for ensuring that databases could be used by several different applications systems without becoming corrupted, and for ensuring that databases used in different parts of the company were consistent with one another.

A further group of eight staff recently transferred from other sites made up a **Systems Evaluation and Assurance** group. They were seen as providing specialist support for achieving quality within S & P. They were intended to carry out feasibility studies as the first stage of developing new systems, and to design test methodologies for assuring the quality of systems. The idea of taking feasibility studies and overall system assurance out of S & P was, however, contested by software producers within S & P.

Finally, a group of two systems analysts were entrusted with an **Infoservices Framework Assignnment**, to investigate long-term data processing requirements within the Company as well as technological trends, and to produce information for Infoservices management to draw up policies for the next five years. The group

also had a brief to communicate these policies to key users throughout the Company. This group was thus a staff support to those senior Infoservices managers concerned with managing the nature of user demand.

Systems Planning and Control was divided into three sections assessing the requirements for the next year in each of the basic application areas around which Systems and Programming was organised. They provided a staff function supporting the annual planning of development work to be undertaken. Staff in this section were also responsible for carrying out surveys of packages available in their respective application areas, to make this information available to other parts of Infoservices. Further, they provided a general expert service to senior Infoservices management in deciding in which areas an innovative approach to differentiation was needed. They briefed senior managers on issues such as the sorts of application areas Infoservices should be moving into, and the types of hardware and software solutions it should be providing and training its staff for.

11.2.2 Divisions of Labour in Software Production

We now turn to an examination of the specialisms involved in producing software within S & P and Data Administration. This provides a basis for developing an understanding of the nature of software careers, in the final part of this section.

Systems and Programming

Systems and Programming was organised in three basic groups which were concerned with supplying software systems for the three basic application areas, accounting, manufacturing and engineering. This application-based form of organisation can be seen as a response to the different types of complexities involved in dealing with these application areas. The tasks

undertaken in supplying software systems within each one were as follows.

Feasibility Study:
This entailed an investigation of the broad area of user requirements and of various technical options for meeting them, in sufficient depth to permit a decision to proceed with the development of a computer system along the lines of one of them. Of particular significance was the decision whether to go for an in-house development or enhancement or for a solution based on a bought-in package.

Functional Requirements Analysis:
This entailed detailed analysis of current user procedures and a specification of the functions to be performed by the computer system. The amount of work involved depended a great deal on whether the system was a new development or an enhancement, whether it was being developed for one site or as a generic system to be used on many sites, and whether or not it had been decided at the feasibility study stage to use a bought-in package. For packages, the emphasis might be more on seeing how user procedures could be adapted to fit the package.

Computer System Design and Programming:
This comprised production of a design for a software system to perform the functions specified, and its conversion into code, usually through an intervening stage of detailed or 'program-level' design. This applied to in-house new developments and enhancements, and to customisations of procedural packages. Code was generally produced in the standard data processing language, COBOL.

Testing:
Once a development or customisation had reached the stage of code it was tested according to standards agreed with the user division at the Functional Requirements Analysis stage.

Installation:

This phase amounted to the transition from a piece of software to a 'live' application. It involved any or all of: installation of hardware (for a new 'stand-alone' system), installation of software, initial software maintenance to remove residual 'bugs', system enhancement work to take account of unforeseen user requirements, and training of users. The amount of work was much greater for new systems than for enhancements. A new system might involve about three weeks' staff time at each site. If the system was to be installed at several different sites, the total installation effort could be of the same order as that for developing the system.

Maintenance:

This involved attending to minor operating problems as well as more drastic failures once systems were installed and operating.

Within each 'application area' group, there were systems analysts, responsible for feasibility studies, requirements analysis and installation work, and programmers, responsible for systems design, programming, and some maintenance. The combination of the two specialisms in a 'product-based' unit appears to have been a result first of management's perceptions of different types of complexity in understanding each application area, requiring different types of systems analysis expertise. Managers also perceived the amount of information processing that needed to occur between analysts and programmers, as, for example, when unanticipated or exceptional circumstances encountered by programmers had consequences for analysts at the implementation stage. For this reason, an earlier arrangement with three groups of analysts and a general programming pool had been abandoned.

Data were collected on divisions of labour in two groups, Accounting and Manufacturing. This revealed considerable differences. Both organised analysts in product-based teams, with

a team leader, so that one team could deal with all the systems in a particular area within the broad remit of the group. This amounted to grouping together tasks with complex interdependencies. Beyond this similarity, the two groups differed in the way programming was organised.

In the Manufacturing Group, a seven-person **Programming** team was responsible for system design and coding specified by the analyst teams, as well as some maintenance work on recently developed systems. In the Accounting Group, more than half of the computer system design and coding was performed within the analyst teams. According to the Accounting Group manager this was because most of the work in his area was on enhancements and maintenance or on customising packages. The size of projects was generally small, a few person-weeks, and coding work was not very specialised. He preferred one team to have control over the whole process of developing and delivering an enhancement, and thus encouraged his systems analysts to do their own design and coding, working as 'analyst programmers'. He also felt that with the arrival of newer parameter-driven packages progressively less coding would be required. Such packages allowed analysts or even users to select functions by supplying values to variables presented in menus. There was little need for programming as such. He was not keen to encourage the development of specialist programmers and saw his group moving toward an entirely 'product-based' structure, with teams consisting of analyst programmers specialising in particular application areas.

For the moment, though, his group retained a programming capacity in the form of two teams of Infoservices staff retained at a remote site. Both remote teams were primarily responsible for carrying out maintenance and minor enhancements of all systems specific to their respective sites.

At the time of the centralisation of Infoservices staff in the late 1970s, it was recognised that there were issues of maintaining and developing programming techniques and standards. This was particularly because the staff trained at the main

Infoservices site had generally been using Jackson structured programming standards for some years, whereas staff brought in from other sites with the general centralisation process often had not. A fourth group was set up within S & P to deal with the problem, **Technical Practice.** It contained five of the most experienced development staff in S & P. Two were responsible for setting standards in terms of development techniques and practices in the control of development projects. Technical Practice thus offered a staff function to managers in the three development groups, defining development techniques and standards. By the time of fieldwork, their tasks of assessing current standards and recommending new ones was at an early stage and they had had little direct impact on development projects.

The other three senior members of Technical Practice were responsible for the supervision of 14 trainee programmers (recently-recruited graduates from university courses or the Manpower Services Commission Training Opportunities Scheme [TOPS]) who shared maintenance work on 'live' applications systems with the three 'development' groups, before taking up positions there. In this respect, Technical Practice was recognised as having an important socialisation function as well as a technical training one.

Data Administration

Development projects within Data Administration generally encompassed three phases:

> **Developing the 'data model' or 'data structure'**, the three-dimensional abstract shape representing the 'information space' in which the data elements could be ordered.
>
> **High level design** of a software system for creating this database, and allowing communications with applications software.

Program-level design and coding.

The group was organised in two teams of five, with team leaders. There was no functional specialisation within teams, except that more experienced members did more of the work in the first two phases. However, the Manager was keen that even his most skilled data structure designers should do some coding, so that they kept in touch with the problems of programmers and could be more 'considerate' in design work. All team members also did maintenance work.

11.2.3 The Nature of the Workforce and its Career Structures

There were a number of separate and parallel career structures for staff involved in applications software development. Each of the three applications groups within S & P were seen as separate areas of expertise, with their own promotion ladders. According to Group Managers, there was in practice no mobility in systems analysts' jobs between the Accounting and Manufacturing Groups. Programmers too tended to specialise in one area or another and to stick to it. As a result, there was a recognisable career structure for analysts or analyst/programmers in Accounting, and one for analysts and another for programmers in Manufacturing. Jobs in Data Administration made up a specialised promotion structure, largely distinct from job structures in S & P.

There was no vertical segmentation in any of these career structures. In all teams, junior team members generally did less responsible or less complex tasks than senior members, but could expect to gain experience and promotion together. Trainee analysts were put to work with an experienced team member on a larger project until they were judged to have sufficient experience to handle less complex projects of their own. In the programming teams, system design was performed mainly by one more senior team member, and low level design and coding by the rest, but the division between the two types of work was again one of

experience. In each team, an experienced programmer was being introduced to progressively more complex design tasks. For very small or simple projects, the functional requirements specification was given straight to a programmer for low-level design and coding. In Data Administration, less experienced staff were given the opportunity to perform progressively more complicated design work, initially under close supervision. In all teams, team leaders were 'productive' team members, in addition to their tasks of monitoring and controlling. They also had responsibility for lateral communications, for example with Operations to make sure that there was the capacity to run a proposed new system.

Educational Backgrounds and Promotion Paths in S & P.

A system of eight job titles was used for software staff. Different titles roughly corresponded to different levels of responsibility, as shown in Figure 11.2 but there was no exact set of duties corresponding to each one. Table 11.1 shows the academic backgrounds and job histories of 13 staff interviewed from within the Accounting and Manufacturing Groups. It shows that staff had a very broad range of educational bac kgrounds in terms of both subject and level, and also that there was no difference in terms of background between those in senior positions and junior positions, or between programmers and analysts. Only two staff had degrees in 'applied computing' i.e. data processing. The rest were a mixture of those who had gone into data processing following a degree course in an arts subject and those without degree level education who had started as trainee programmers after working in some form of clerical work or as a computer operator. Overall, only 7 of the 13 interviewed had degrees.

The basis on which staff attained senior analyst or programming positions was experience, rather than any definite educational standard. People with degrees in applied computing still had to start as trainee programmers, although the rapid progress of the

Figure 11.2: Job Titles in Infoservices

Trainee Analyst	Junior	Trainee Programmer
Analyst 2	Monthly	Programmer 1
Analyst 1		Programmer 2
Systems Analyst 2	Senior	Program Designer
Systems Analyst 1	Monthly	Systems Designer
Senior Systems Analyst		Senior Systems Designer

Project Leader

Senior Project Leader

Sex	Job Title	Age	Education	Nature of First Job	First Employment at Infosen	Length of time at Infosen	Professional Affiliation	TU Member
M	Program Designer	25-29	'O' levels	Computer Operations	No	7 years	None	No
F	Program Designer	25-29	'O' levels TOPS programming	Typist	No	3 years	None	No
M	Systems Designer	30-34	HNC Business Studies	Clerical Work	No	7 years	None	No
M	Systems Designer	25-29	B.Sc. Applied Computing	Programming	Yes	4 years	None	No
F	Trainee Systems Analyst	20-25	Secretarial qualification	Purchasing	No	3 months	None	No
M	Trainee Systems Analyst	30-34	BA Theology, TOPS Systems Analysis	Programming	No	3 months	None	No
M	Analyst II	20-24	MA Psychology	Systems Analysis	Yes	1 year	None	No
M	Analyst I	20-24	BA History	Systems Analysis	Yes	2 years	None	No
M	Systems Analyst II	25-29	B.Sc. Economics	Systems Analysis	No	1 year	None	No
M	Systems Analyst I	30-34	PhD. Social Anthropology TOPS Programming	Lecturing	No	2 years	None	No
M	Systems Analyst I	25-29	'O' levels	Computer Operations	No	3 years	None	No
M	Senior Systems Analyst	35-39	Accountancy qualification	Accountancy	No	3 years	None	No
M	Project Leader	30-34	B.Sc. Computing & Business Studies	Programming	No	2 years	BCS member	No

systems designer with this qualification, compared to his colleague without it, is an indication that people with degrees in applied computing could expect to progress more rapidly. Table 11.1 also shows that whilst a minority of people in senior positions attained them through being promoted from the most junior grades within Infoservices, most joined in middle positions, with some experience. Whilst a continuous promotion ladder was on offer in both programming and analysis, the Division did not operate a closed internal labour market, but recruited at all levels. Indeed the short lengths of time for which staff had been with the Division suggest the high labour turnover typical of the data processing software industry. This is supported by data in Table 11.2, which show that those interviewed attached little importance to developing a career within one company, and saw their skills as transferable to other companies.

According to managers, management positions in S & P and Data Adminstration, and the various staff positions in Technical Practice, Systems Support and Systems Planning Control were all filled by people who had been experienced software producers. There was no educational or professional qualification associated with management or staff positions. They were effectively extensions to the experience-based job ladders operating within development groups. Table 11.3 indicates that the majority of staff interviewed saw themselves moving into management positions in five years time, either within Infoservices or elsewhere.

Nature of the Workforce in Data Administration

Unfortunately, no interviews were possible with staff within Data Administration. According to the Manager of Data Administration, however, his staff were all graduates, from a number of disciplines, mostly scientific. He saw them as being of higher educational achievement and more skilled than people in S & P. He was particularly keen on mechanical engineering graduates, because of their ability to conceptualise data structures in

Table 11.2: Job individuals expect to be in in five years time

	Team Leader	Systems Anal.	Junior Analyst	Systems Des.	Program Des.	Overall
N	2	2	4	2	2	12
Responding	2	1	3	1	2	9
FUNCT SYS ANAL		1				1
SW DEV MGT					2	2
PROJ, CLIENT MGT	2		3	1		6

Table 11.3: Career goals of software producers

Mean scores for importance on a scale of 1 to 9, broken down according to position in the division of labour, and overall.

	Team Leader	Systems Anal.	Analyst-Prog.	Junior Analyst	Systems Des.	Program Des.	Overall
N	2	2	1	4	2	2	13
Developing a Career in one Company:	4.5	2.5	3.0	2.8	3.5	4.0	3.3 (s.d.=1.4) (s.d.=1.2)
View on transferability of skills: (Scale: 1= 'not at all' to 7='completely')	4.0	6.5	7.0	4.3	4.5	6.5	5.2 (s.d.=1.5)

three dimensions. Most of the group had come through the STC graduate recruitment programme. There was little scope for recruiting experienced people from outside because knowledge of the specific database management system used was required, and it was only in use at a handful of UK installations. The Manager was also keen on maintaining a definite culture around commitment to work in his group. He saw commitment as being built up as people were trained. The few instances of introducing experienced people from outside had not been successful. It would appear then that something much more like an internal labour market operated within the Data Administration job structure.

11.2.4 Overall Control and Coordination

The basis for overall management control and coordination of Infoservices software activities was a system of centralised action planning. An annual plan for software development projects was drawn up by the General Manager and four departmental heads. This was a mechanism for directing Infoservices activities to meet user priorities, and for addressing the unpredictability in the demand for major development projects which could emerge from the highly political system of the various ISRBs.

The overall annual workplan was outlined in April of the year preceding, by Planning and Control, drawing on three sources for ideas as to new developments needed. Requests came from user division managers, often through their FISRB, and direct from the CISRB, for large systems. Ideas stemming from awareness of new technical possibilities also came from staff within Infoservices. Planning and Control attached rough estimates to the amount of development effort required for each system proposed. Such estimates formed the basis for the five senior Infoservices managers to decide which systems Infoservices should propose to develop over the next year.

The initial plan contained some slack so that it could be refined over the course of the year to take account of changing

priorities. Even so, as needs for new applications emerged, their priority often had to be assessed relative to those already scheduled. As one development team leader put it, plans were changed when a user manager 'went up the hierarchy' to get what he or she wanted. The criteria used by the five senior managers in judging priorities were to a large extent political. It was deemed important to meet the requests of powerful divisions or Board members, because there was seldom the information available in advance to make meaningful cost-benefit analyses as to the relative advantages of developing two different systems.

The overall plan was elaborated into more detailed action plans at departmental, group and team level. As a rule, the lower the level of these action plans, the shorter their timescale. As part of compiling the annual workplan for Infoservices as a whole, outline annual workplans for software production tasks were compiled by the three S & P Group Managers and by the Data Administration Manager. These gave rough allocations of projects to each team for the year. More detailed task specifications and targets were allocated to teams as the year unfolded, as the completion of one phase of a project produced a more precise specification and estimate for the next, as senior Infoservices management added new projects to the Group's load, and as the demand for maintenance and minor modifications on existing systems clarified. The increasingly fluid lower levels of action planning thus served on the one hand as a way of dealing with unpredictabilities in demand for work that stemmed from the responsibilities of most development teams for maintenance work. On the other hand, they were a way of dealing with 'technical' unpredictabilities in terms of difficulties in assessing user requirements at the outset of a project. Hence the need to refine project estimates as information became available during the course of a development.

(a) Low-Level Action Plans in S & P

For the S & P analyst teams, lower-level action plans were drawn up on the basis of targets and specifications for feasibility studies, functional requirements analyses and installation work. Those for feasibility studies were made jointly by a team leader and group manager. Those for functional requirements analyses were drawn up by the systems analyst performing the feasibility study, and checked by their team leader, before going to the Group Manager and the Manager of S & P for approval. They were then proposed to a user representative, who could accept them, reject them, or ask for the project to be scaled down. The result was a contract between Infoservices and the user, known as the Project Index (PI). Estimates for installation work were again drawn up by team leaders and Group Managers, usually at the same time as those for functional analysis and agreed by the user as part of the Project Index. Overall, team leaders had an important role in estimating.

For the S & P programming teams, estimates and specifications used in drawing up workplans came from the outputs of the functional requirement analysis stage of projects, when 'ball-park' figures for systems design and programming effort appearing in the Project Index were refined. Programming estimates were made by the analyst performing the functional analysis in consultation with the programming team leader. They had to be agreed by the Group Manager and the Manager of S & P, and usually went into a second Project Index to be signed by the user. For small developments the programming effort might be estimated and agreed directly at the feasibility stage.

All those involved in producing estimates agreed that they were made entirely on the basis of experience and without any formal techniques or aids. Attempts had been made to use a matrix table giving estimated development times for systems according to their complexity and experience of development staff. It had however been found to be unreliable, generally erring on the low side.

Estimates were combined into detailed workplans by each Group Manager in consultation with their team leaders, at three month intervals. Workplans were generally compiled for six months ahead, and regarded as fairly set for three months, apart from minor adjustments to the project load permitted by the slack built in. Some time was left free so that maintenance calls could be slotted in. The remainder of the plan was more tentative. These three-monthly revisions took account not only of new projects being added, but also of slippage within phases of current projects, and of unexpectedly heavy calls for maintenance or minor enhancements. The extensive nature of quarterly revisions can be appreciated from the judgement of the Accounting Manager that about half of the projects planned at the beginning of the year actually took place during the year.

Action plans drawn up within S & P were also used to control and coordinate activities of user staff involved with computer systems. A programme of user activities needed for a system to become operational was usually agreed between user and S & P staff at the conclusion of each functional requirements analysis. The activities scheduled mostly applied to the installation phase, when new data-handling procedures had to be in place, and became part of the plan each team leader drew up for each project they were responsible for.

Group Managers and the Manager of S & P all stressed the importance of obtaining user commitment to both the system specification and this schedule of activities, as S & P staff embarked on system design and programming. The acceptance by the user of the output of functional requirements analysis was seen as a two-way commitment by the Manager of S & P.

(b) Low-Level Action Plans in Data Administration

The Data Administration Group's dual responsibility for development and maintenance meant that low level action plans had to be considerably more fluid even than within the S & P

development groups. Each development project was estimated as to the effort involved in database work by one of the two team leaders at the beginning of the year, or when the Infoservices management group decided to add it to the project load. These estimates were checked by the Manager of Data Administration and then included in the Feasibility Study Report and the Project Index. The Manager of Data Administration held that these early estimates were very unreliable. When S & P completed the functional requirements specification, Data Administration team leaders made a second estimate for the database development work to be carried out in parallel with the programming of the main system. These estimates were approved by the user as part of the total development.

Even at this stage, the Data Administration Manager did not place a lot of faith in the estimates. A major part of developing the database for an application was constructing the 'data model'. It was very difficult to judge how long it would take to code the database management instructions until this had been designed. He considered the technical uncertainties in this area of development not only to be very high, but also as persisting until an advanced stage of the development. These estimates were nonetheless used as a basis for charging the user and for allocating work to Data Administration staff.

The larger and more complicated data-modelling and computer system design phases were allocated to experienced staff only. Because there were only a handful of such staff, it was often necessary to schedule these tasks up to six months in advance. Programming could be done by anyone in the group, and the Data Administration Manager preferred not to allocate these phases of projects more than a week or two in advance, so that he could allocate his staff flexibly to deal with 'panic' maintenance calls.

Overall, then, the Data Administration Manager faced technical complexity inherent in database design work combined with uncertainty stemming from the unpredictability of user demand for

maintenance and enhancement work. In response, he scheduled definite targets far in advance for the most complex elements - 'data-modelling' and computer system design - and left the rest of his resources uncommitted to be scheduled for low level design, programming and maintenance tasks at a few weeks' or even days' notice.

11.2.5 Control of Individual Performance

Group Managers delegated day-to-day control of meeting targets for development work to team leaders. This section summarises the different means which team leaders saw as important for achieving the quality and productivity of performance required from individual team members.

(a) Professional Skills and Commitment to Targets

The most fundamental mechanism of control of individual performance was team leaders trusting team members to use skills and experience to meet targets set for them.

Team members used their team's work plan to allocate blocks of work to individual team members. Work was allocated to individuals on the basis of what it was felt they could handle. They allocated blocks of a few weeks at a time, matching the level of complexity to the experience of the individual. Team leaders interviewed claimed they did so 'by gut feel'. They looked at the work that needed to be done to achieve a milestone, and broke it up into stages that they felt they could estimate on the basis of their experience.

On analyst teams, feasibility studies, implementation work and minor enhancements were nearly always allocated to one person. Functional analyses were also commonly one-person jobs, but could involve two or more. In both programming teams where data was collected, the number of people engaged on a project could vary

from one to the whole team.

(b) Structured Programming Techniques

There were no definite or prescribed techniques for performing systems analysts' tasks. Quality was assured partly through some norms or standards as to what should appear in the reports produced at their conclusion. Team leaders, however, attached greatest importance to the non-codified skills of their team members, which enabled them to carry out development and installation work according to these standards. Functional requirements were represented by flow diagrams depicting the highest level design of the system, annotated in ordinary English, rather than any formal design language.

For the work of programming staff i.e. system design and coding, there was a greater, but rather variable use of codified methodology. Structured Programming was seen by most team leaders and managers as having an important role in assuring quality of code produced, although, according to team leaders there was still at least as much reliance on experience and 'tacit' skills. The two were seen as complimentary. The advantage of structured programming was seen in that it made arriving at a good design easier and more reliable. Likewise, the manager of Technical Practice felt that a good designer/programmer would probably arrive at the same design without the technique. The technique would however speed up the process for less experienced individuals and make it clear **why** the design was a good one.

The use of structured programming was the result of an Infoservices management group decision of some years previously to use Jackson Method Structured Programming, but there was some variation in the degree to which it was applied in different groups. Until about a year before fieldwork, there had been no concerted attempt to make all groups adopt the methodology. The degree to which they had done so appears to have been the result

of the beliefs of particular team leaders and members as to how useful structured programming was.

According to the Manager of Technical Practice, prior to the centralisation of Infoservices staff, a reasonably uniform set of beliefs had been established at the main site, and structured programming had been standard. In contrast, staff at other sites had remained more remote from the senior management view on the techniques, and had developed a more informal approach to programming standards, facilitated by the smaller size of their groups. When the majority of them had been transferred to the main site, standards had become diluted. The periodic use of programming contractors had a similar effect.

The Accounting programming team leader remaining at a remote site was still untrained in structured programming, and sceptical of its value. He had been asked by the Manager of S & P to 'provide evidence that his staff were using structured techniques', but appeared unconcerned. His staff continued to use a less rigorous 'modular programming' approach. In addition, those responsible for system design and programming in all groups reported that they could not use structured approaches when modifying older applications systems which had been developed before the introduction of structured programming.

In the Data Administration team, the fundamental form of quality assurance was again the use of knowledge and tacit, non-codified skills acquired through experience. The Manager did not see much of a role for structured approaches or formal methodologies:

> "You can design a database by applying logic, but I prefer to rely on gut feel and black art."

(c) Availability of Help

A second element in assuring quality of work was the availability of help to software producers from team leaders and colleagues. Table 11.4 shows data collected from five S & P staff with the Accounting and Manufacturing groups, concerning the extent to which they consulted their colleagues and team leader for help in general. Although this is a very small sample, the data is strengthened by the fact that it was collected in each case over ten days in ten successive weeks, and thus represents the behaviour of these individuals on several different occasions. Table 11.4 suggests that staff appear to have turned to colleagues as much as to their team leader, although team leaders saw providing help with technical problems as a central aspect of their role.

(d) Project Milestones and Informal Monitoring

'Milestone' reviews of the outputs of project phases were points at which team members were made accountable for how they had used skills with discretion in meeting targets and specifications. Above all they had an important quality control function.

Reviews were conducted as follows. Feasibility study reports and functional specifications were reviewed first of all by S & P managers outside the development team and then by users. Once coded, systems were subjected to the acceptance tests agreed with the user. For large developments, the Manufacturing programming team leader reviewed both system- and program-level design, by 'talking through' with the team member concerned. Within Data Administration, quality was controlled through informal reviews, one at the computer system design stage, prior to low level design and coding, and the second once the code had been produced. The computer system design was checked by another group member of similar experience to the developer, but the checks to be performed were not formally specified. Code was reviewed through the system designer and programmer conducting a

Table 11.4: General Help From Team Leaders and Colleagues

		Help From Team Leaders		Help From Colleagues	
	N	Av. No. Times per day	Av. Total Time per day (min)	Av. No. Times per day	Av Total. Time per day (min)
Senior Systems Analyst:	1	0.4	10	0.6	14
Junior Systems Analysts:	2	1.3	14	1.1	17
System Designer:	1	0.4	5	1.0	7
Program Designer:	1	0.7	6	0.3	5

Table 11.5: Discussing Interfaces with Team Leaders and Colleagues

		With Team Leaders		With Colleagues	
	N	Av. No. Times per day	Av. Total Time per day (min)	Av. No. Times per day	Av Total. Time per day (min)
Senior Systems Analyst:	1	0.1	3	0.4	12
Junior Systems Analysts:	2	0.3	16	0.3	3
System Designer:	1	0.2	2	0.6	4
Program Designer:	1	0.3	40	0.0	0

Table 11.6: Informal Reporting Patterns

	No. Resp.	Average No. Times Per Day	Av. Total Time (min) Per Day
Senior Systems Analysts:	1	1.5	37
Junior Systems Analysts:	2	1.5	17
System Designers:	1	2.0	18
Program-level designers:	1	1.0	13

'walk through' of the functioning of the system, with one or two other group members present.

None of the team leaders interviewed regarded structured techniques as a basis for close monitoring of detailed progress between milestone points, or as a means for putting pressure on their team to produce more quickly. The basic mode by which team members reported on progress to team leaders was informal. Table 11.5 shows data on frequencies of informal reporting provided by the five S & P staff who completed diary sheets. It shows a pattern of at least daily reporting, at all levels. None of the team leaders required their people to submit any formal reports apart from their time-sheets, although one analyst Team Leader held weekly verbal reviews of progress and problems.

According to team leaders, they used their daily contacts to establish whether there were signs of a serious technical problem which might cause slippage, or whether it looked like a particular block of work would not be completed according to its estimate. They were not generally concerned with applying pressure to get people to work more intensively, although this was occasionally called for.

11.2.6 Low-Level Coordination

(a) Informal Standardisation of Skills

The process of setting specifications and time targets already described was the basic mechanism for coordinating work between different teams or different specialisms within teams involved in producing software. Analysts performing requirements analysis specified the work that needed to be done by programmers within S & P, and by software producers with the Data Administration Group. In general this was made possible for a set of informal standards that had evolved as to how specifications should be notated, which can be seen as informal, company-specific standardisation of skills. The form of specifications was

generally flow charts, annotated in English. Specifications used to co-ordinate between analysts and programmers, and between analysts and the Data Administration group were not based on rigorous structured techniques.

(b) Direct Lateral Contact

There were however occasions when notation alone was seen as insufficient for communicating what programmers or Data Administration staff were supposed to do. Team leaders saw the type of notation used as still leading to some ambiguity in the representation of user requirements at the functional analysis stage. Part of this ambiguity stemmed from uncertainty as to what user requirements were, but another part stemmed from imprecision in the notation used. Whilst some ambiguities could only finally be resolved when the system had been coded and installed, a mechanism for minimising the need for 'retrofitting', practised in both the Accounting and Manufacturing Groups, was for analysts to 'talk through' their requirements analysis when handing it over for system design and coding.

Some direct lateral contact between analysts and programmers was thus required. During functional requirements analysis or programming, further problems of coordination could arise when a specification did not fully define the interface between the parts of system being developed by two different team members. In such cases, the interface had to be clarified as development work proceeded. As Table 11.6 shows, this was done either by direct lateral contact, or else by referring problems to the team leader.

(c) Structured Programming Standards

Whilst structured standards do not seem to have played an important role in coordinating between analysts and programmers, the manager of Technical Practice, in common with the majority of direct producers interviewed, saw structured programming as vital for coordination between those responsible for producing software systems and those responsible for maintaining them. The Manager of Technical Practice saw it as vital that systems be developed according to rigorous structured programming standards if it was going to be possible for them to be handed over from the team that had developed them to his own group, for subsequent maintenance. Otherwise, code was very difficult to read for someone other than the person who had actually written it. Structured programming thus appears to have been an important element of standardisation of skills by which development and maintenance staff could co-ordinate their work.

11.2.7 Managing the Local Project Environment - Control of User Activities

S & P management was also concerned with controlling the execution of various undertakings of user staff specified in project plans. These activities did not generally assume much importance until the installation phase, but at this point they could be crucial to the system becoming operational by the intended date. In effect, managers and team leaders found they had to manage the part of the environment they were developing a system for, in order to make sure they could meet its expectations. This localised environmental management was carried out by Infoservices staff at a number of different levels.

The Manager of S & P dealt with those user activities which required high-level initiation or sanction within the user organisation. These included acquisition and installation of major items of hardware. Group managers and team leaders were

responsible for relating to user managers responsible for training staff and setting up data-handling procedures in readiness for a new system. For installation phases, team leaders worked from a project plan with a schedule of activities for the user as well as for his own staff. There was a general consensus that in the past this had been one of the most problematic aspects of project control. Team leaders had often found that user managers and staff resisted the idea of a new computer system even after its development had been formally agreed, and Infoservices had no formal sanctions with which to get users to accomplish what was necessary. The result had been delays over which they had had no control, but which had reflected badly on Infoservices' reputation at Board level.

According to the manager of S & P, the introduction of the current form of Project Index had gone a long way to solving this problem. The basic way of dealing with this form of environmental uncertainty was thus contracting. The Project Index functioned as a tool for managing the local project environment because it spelt out the user's obligations as well as those of Infoservices. It made it clear to user managers that it was their responsibility to motivate their staff to co-operate in the necessary preparations. The manager of S & P saw regular management contacts and reporting to user managers during development as a further way of reinforcing user commitment.

However, group managers and team leaders continued to see user commitment as an issue in project control from time to time. One team leader had encountered very strong 'passive resistance' from a key user manager during an installation. The strategy he had adopted was to break down his project plan into daily activities - for the user and for his own staff. He then sent daily reports to the user manager with the intent of 'frightening him'. He was emphatic that the daily progress reporting this required from the systems analyst concerned was completely unnecessary from the point of view of controlling his work. The team leader considered the man was 'too good' for his work to be broken down into less than four week blocks under normal circumstances.

11.2.8 Sources of Slippage and Mechanisms for Adjusting Plans

The mechanisms for control and coordination so far described all served to make execution of work to meet targets predictable. S & P managers considered that their collective project management skills in setting specifications and targets, and motivating people to meet them had produced a generally satisfactory record on development projects. The Accounting manager estimated that over the last year 60% of his projects had been completed to schedule, and of the 40% that had overrun, only about one third had overrun by more than a ten per cent margin it was usually agreed should be funded by the user.

Slippage from targets did however occur to a significant extent. It was usually attributed to time spent in resolving ambiguous user requirements, demands of maintenance exceeding what had been built into the workplan, or recalcitrance on the part of the user division. Only in Data Administration was there reference to purely technical problems giving rise to slippage. Below-standard productivity from software developers was also not seen as a significant source of slippage.

This slippage created the need for team leaders and managers to collect information on progress, and to make adjustments to action plans.

(a) Vertical Information Systems

Monitoring of projects was carried out through complementary formal and informal systems. In the **formal system**, a computerised project reporting system took data from daily time-sheets filled in by programmers and analysts indicating the amount of time they had spent on the project or projects currently allocated to them. It produced weekly and monthly reports for Group Managers on time expended compared to targets for project phases.

Whilst they felt that this provided an overview of how the group's projects were doing overall, it only indicated how much development time was still to be put in for each phase of a project, by subtracting the amount of time logged from the original estimate. Group managers considered that it did not provide all the information they needed to make adjustments to schedules or staffing. For this they were dependent on the informal monitoring of progress already described, carried out by team leaders in the course of regular contacts with their team. They saw this as a way of identifying problems and potential slippage at as earlier a stage as possible so that targets could be adjusted if necessary.

Team leaders used the information they gathered first of all to call on expertise from elsewhere in Infoservices e.g. from Technical Practice, to try and solve problems before they caused slippage. Second, they passed on information on significant problems and slippage as it occurred to their group manager. Group managers interviewed reckoned on talking informally to each of their team leaders once or twice a week.

Within Data Administration, the group manager considered daily informal contacts with his two team leaders sufficient for the information he required. Within the larger S & P groups, collation of information was more elaborate. Team leaders compiled monthly written reports which they reviewed with their group managers, summarising progress, identifying problems and commenting on staff performance. These reports contained re-estimates of the amount of work still to be performed on tasks where slippage had occurred, or where the emergence of a technical problem meant that original estimates were no longer tenable.

Group managers used their team leaders' monthly reports to compile a monthly summary report required by the manager of S & P, detailing the status of all projects and estimates for the amount of work needed to complete them. They also had principal responsibility for reporting progress to the user, submitting

monthly reports to the user manager nominated in the Project Index as responsible for the project. We have already seen that regular management contacts served as a means for reminding the user of what was required on their part. They also ensured that requests for changes in budgets or timescales 'did not come out of the blue'.

(b) Renegotiating Delivery Dates

The primary adjustment mechanisms available to group managers was renegotiating delivery dates. If slippage was occurring that looked likely to produce an over-run in cost of less than 10%, according to the terms of most project indices, the user division was liable for the extra development time. The only adjustment needed from the group manager was to negotiate a later delivery date. If the project in question was considered by the group managers and the manager of S & P to be a politically sensitive one where it was important for Infoservices to be seen to deliver on time, they renegotiated the delivery date for some other project allocated to the team in question.

(c) Raising a Supplementary Project Index

In many projects, additional requirements were proposed by the user during Functional Analysis or even System Design and Coding. If they were thought to be minor, these were initially accepted by S & P without extra charge. During the course of development, slippage might occur, however, as the ramifications of these extra requirements became clearer. In such cases, on hearing of the extra work through the monitoring system, group managers would persuade the user division to agree to a supplementary project index, to cover the extra cost, and to delayed delivery. Even if slippage could not be directly attributed to new requirements, group managers would still put a case for additional funding on the grounds that extra work had been requested. Asking for additional funding was seen as a way of

passing on uncertainties in the development process to the user. In almost all projects, there were extra requirements around which a case could be made.

(d) Utilising Slack Resources

If a project was over-running and it was not possible to justify a request for additional funding from the user, additional resources had to be supplied from within Infoservices. For small over-runs, this took the form of the use of time charged to maintenance, or asking team members to work overtime. Senior team members were expected to work overtime without extra payments, although team leaders felt there were limits to the amount they could ask them to do so. Maintenance time and the possibility of overtime for senior staff functioned, then, as slack resources to deal with minor slippages.

For more serious slippage, or when it was considered vital that a system be seen to be delivered on time almost regardless of the cost to Infoservices, the group manager and the manager of S & P might decide to add extra personnel to the project. These might be taken off less crucial projects on the same team, from another team within the group, from another Group, or from elsewhere in Infoservices. If workplans were very tight everywhere, contractors were brought in.

11.3 MECHANISMS FOR ESTABLISHING COMMITMENT TO WORK

Infoservices managers and team leaders identified the following means for mobilising commitment to meeting project targets, as a basis for controlling performance at the individual level.

(a) Recruiting "the right people" and Socialising Them Appropriately.

Managers in both S & P and Data Administration had definite ideas of the sort of staff they required, who could be relied upon to do the type of work typical of their department with the requisite skills, sense of responsibility, and level of aspirations.

The manager of Technical Practice was responsible for the formation of new staff within S & P. He emphasised the importance of encouraging commitment to work and to the team, and the orientation of solving problems rather than bringing them to managers. At the same time he felt it was important to select people who did not want too much challenge or variety in their work, but who were looking for autonomy and responsibility and autonomy in doing the same basic class of data processing tasks year-in year-out. Hence his preference for MSC TOPS graduates over university graduates.

In contrast the Data Administration Manager required people who expected to be challenged much more, with correspondingly greater commitment to seeing a task through, regardless of its complexity and the need to work outside office hours. He preferred more ambitious university graduates. This appears to have been because of greater complexity and uncertainty of work in this group. The Data Administration Manager however found it very difficult to recruit staff with the aptitudes he required, given the levels of pay on offer. He was also reluctant to 'take on passengers'. As a consequence, his group was two under its permitted head count.

(b) Supportive Style of Supervision

Team leaders saw sustaining team members' motivation as linked to maintaining daily contact with them, although it was important that they allowed team members to take responsibility for their

tasks. They would enquire about progress in a general way, and provide help if difficulties had arisen. They considered that more experienced practitioners needed less contact and help, and worked better if left to their own devices.

A further important aspect of regular contact was to 'keep people in the picture', informing them of what was going on within their team in terms of progress on current projects or new projects expected. A third aspect of supervision lay in establishing personal rapport with subordinates, so that they felt committed to their team, and the norms of effort established within it. The Data Administration Manager saw this team commitment as linked to an identification with the specialism of his group, and the desire to achieve status as a competent practitioner within it.

Most team leaders however saw the need to depart from this 'professional' style of supervision, with its underplaying of managerial authority, on occasions. Periodically, it was necessary to remind staff directly of the sort of performance required. This was emphasised most strongly by the manufacturing programming team leader. He considered it necessary from time to time to police the length of lunch breaks, to stop people 'stealing time from the company'. Every couple of months he would hold an information meeting, when he would inform his team as to what was happening in the rest of the group, and also raise issues of their 'general attitude' and how it was important to use their time well. At times he also found it necessary to "lean" on individuals, and ask them "have you done this yet?"

Other team leaders, too, recognised the importance of knowing "when to lean and when to back off" as the essence of successfully maintaining their team's motivation. It was particularly important to "lean" with younger staff, so that norms became established. The Manager of Technical Practice, who was responsible for the newest recruits, also stressed the importance of putting pressure on staff.

(c) Providing Job Interest and Responsibility

Team leaders saw giving people considerable discretion and responsibility in the performance of tasks which provided variety and interest and which had a visible impact from the point of view of the user as basic preconditions for motivation. They recognised that the sort of impact expected by a senior team member was different from that sought by a novice. The latter liked to see what was achieved with each day's work, perhaps in the form of producing a section of code. More senior people would expect to see bigger achievements e.g. a complete system designed, enhanced or installed and functioning properly, but at much bigger intervals, of weeks or months.

Analyst team leaders saw giving team members the chance to have direct contact with user staff as an important element of job interest. One team leader also thought it important to keep 'administrative hassles away' from team members to enhance their job interest, ensuring that they had the minimum of formal reporting responsibilities.

The Data Administration Manager considered that work in Data Administration was more demanding and interesting than ordinary applications development in S & P, and on this basis it was possible to demand greater commitment. In particular he felt that maintenance 'panics' often required a great deal of initiative, responsibility in decision-making, and working until the job was done, if necessary out of office hours.

(d) Providing Training and Development

It was considered desirable to allocate individuals with work that challenged them and offered opportunities for development. Managers also placed emphasis on the role of annual assessments, in which plans for development through being given more responsible and complex areas of work and training were drawn up.

(e) Pay and Promotion

Team leaders generally considered that pay and promotion were only important for motivation in a negative sense. If an individual felt they had not been promoted sufficiently or were underpaid they would be difficult to motivate. Pay and promotion were on the whole seen as background preconditions for motivation rather than as positive stimuli producing it. The exception to this was the Data Administration manager, who saw pay rises and promotion as vital to maintaining commitment. He saw promotion simply as a way of putting people on a higher salary scale, and attached little significance to job titles. What people actually did within teams depended on their level of competence. Not giving promotion was seen as the main means for encouraging people to leave if they were judged not to be meeting the norms of commitment.

Whilst pay and promotion were decided according to standard Telecorp procedures, the manager of S & P delegated most decisions on promotion and the merit part of pay rises to the group managers. In turn they consulted with team leaders. Two S & P team leaders interviewed felt they could get the promotions they considered appropriate for their staff, although sometimes rather later than they thought ideal.

Table 11.7 shows data obtained from 13 direct producers within the Accounting and Manufacturing groups of S & P which indicates that the promotion system was seen by producers themselves as encouraging a commitment to use skills with discretion and responsibility . It was seen as rewarding the 'ability to solve problems independently' first and foremost. The second most important criterion for both analysts and programmers was 'understanding of the application area'. This suggests a further way in which promotion criteria shaped the 'local cosmopolitan' orientation. Direct producers saw they were valued if they had understanding of the application area, of user procedures and

Table 11.7 Perceived Importance of Various Criteria for Promotion.

Mean scores for importance on a scale of 1 to 9, broken down according to position in the division of labour and overall.

	Team Leader	Systems Anal.	Analyst-Prog.	Junior Analyst	Systems Des.	Program Des.	Overall
N	2	2	1	4	2	2	13
Problem Solving ability:	7.5	8.5	7.0	7.8	7.0	8.0	7.7 (s.d.=0.9)
Performing according to procedures:	6.5	5.0	5.0	7.0	5.5	6.5	6.2 (s.d.=1.6)
Good technical knowledge:	5.0	7.5	7.0	5.3	6.5	7.5	6.2 (s.d.=1.6)
Understanding of application area:	7.0	8.5	7.0	7.3	7.0	7.0	7.3 (s.d.=0.9)
Management ability:	7.5	6.5	-	5.8	7.5	5.0	6.3 (s.d.=1.6)

problems, rather than simply of technical issues internal to the development process. They felt evaluated in terms of how well their systems performed in an organisational context as well as according to more abstract technical criteria.

It is also significant that those interviewed saw themselves as being judged at least as strongly regarding how well they performed according to procedures, as according to their technical knowledge. This is evidence that an emphasis on adherence to standards co-existed with one on technical creativity.

11.4. THE NATURE OF TENSIONS AND CONFLICTS

11.4.1 Issues for Managers

When asked to identify their major concerns, managers at various levels identified the following range. It is significant that improving the quality or productivity of individual performance of direct producers was not seen as an explicit issue. Managers were concerned with improving their control over aspects of organisational performance rather than how hard or well direct producers worked.

Senior Management:

(a) **Product and process technology strategies.** Product strategy revolved around questions of the sort of products and services the division should be supplying in order to compete with what external suppliers could offer. Process technology strategy concerned the options for doing so in terms of hardware and software approaches. These were the realms where senior managers saw the most important 'strategic choices' as being made, in order to maintain

Infoservices's position with the corporation.

(b) **Environmental management**, in the sense of intervening in Telecorp as a political system, in order to influence the thinking of powerful actors as to the sort of information systems that should be developed, and the expectations they had of Infoservices.

(c) **Developing team-leading and project management skills.** The Manager of S & P saw the presence of team leaders and group managers with sophisticated project management skills as central to the functioning of his department. In particular, he identified the skills of estimating and allocating work to suit the capacity of individuals, a knowledge of the various 'tools' available for project control, and/or 'when to lean and when to back off'. He felt that the 'best schoolroom was right here', that the best way for people to develop such skills was by being given the opportunity to develop them. However, he was worried whether the next generation of team leaders and group managers would turn out to be as effective as the current ones.

Middle Managers i.e. group managers within S & P and Systems Support, and **Team Leaders:**

(a) Scheduling developments in the light of changing user priorities during the course of the year.

(b) Obtaining accurate formulations of system requirements from users at the outset of a development, to avoid over-runs at the installation stage through time-consuming 'retrofits'.

(c) Maintaining user cooperation in the activities necessary on their side, so as to facilitate rescheduling delivery dates when necessary.

In addition, there were some tensions experienced by team leaders and group managers, with respect to the degree to which their sub-units should go their own way in terms of how work was organised and problems were tackled.

On the one hand, there was an emphasis on team leaders and group managers taking responsibility for negotiating their own targets and meeting them. The team leader or manager encouraged commitment to work and to the group or team. In two groups, in particular, the Accounting programming team and the Data Administration group, this seems to have led to distinctive practices which led to problems for middle managers.

In the first case, there was resistance to S & P wide standards. The team leader saw such ideas as an unnecessary interference in the way he managed his team. There was a tension between a need to set uniform quality standards across Infoservices development groups and the local system of 'high trust' work organisation that had grown up around a team leader.

The Data Administration Manager experienced a rather different contradiction. Whilst he valued being left to his own devices, he felt that he was given too little attention by the manager of Systems Support. He wanted more understanding of the problems he had in running his team. In particular, he wanted help finding new staff of the kind he wanted, and in fighting battles with user divisions in terms of the type of database development they were prepared to commission. There is a suggestion here that there was a connection between the extent to which the Data Administration Manager took responsibility for his own area without involving his superior to any great extent and a difficulty he found in exerting organisational influence.

4.2 Issues for Software Producers

This section presents findings on the nature of work experiences and the nature of conflict stemming from them for 13 individuals within the Accounting and Manufacturing Groups of S & P, obtained from structured and unstructured portions of the interview schedule. In the tables that follow, the sample is broken down into six small groups, according to the type of responsibilities held: two team leaders, two senior analysts with responsibility for functional analysis of major new systems and enhancements, four junior analysts, with responsibility for minor enhancements, one analyst programmer working on minor enhancements, and, on the programming side, two systems designers and two program-level designers.

Table 11.8 shows that overall work goal achievement was on the whole moderate. Clearly, there was a basic harmony between expectations of work and what was on offer for the bulk of those interviewed, suggesting that such tensions as there were occurred within a paradigm of 'high trust' or 'professional' work organisation that was acceptable to both managers and producers. There was at the same time evidence of tensions. Two staff were less than satisfied according to both of the measures used - an analyst programmer working on an analyst team, and a program designer on a programming team, both within the Accounting Group. In addition, one senior analyst within the Manufacturing Group felt that his job was 'satisfactory' to the extent that it provided what he wanted out of a job, whilst it 'met his most important personal work-related goals' 'only slightly'. In order to illuminate these tensions, we now examine in more detail data collected on experiences of current work and on promotion prospects.

Table 11.8: Views on overall goal satisfaction

Mean scores for satisfaction on a scale of 1 to 7 (4 = 'about right'), broken down according to position in the division of labour, and overall.

	Team Leader	Systems Anal.	Analyst-Prog.	Junior Analyst	Systems Des.	Program Des.	Overall
N	2	2	1	4	2	2	13
Work achieving most important personal goals:	5.5	4.0	3.0	4.3	4.5	4.0	4.3 (s.d.=0.9)
Job measures up to what you want:	5.5	4.0	3.0	4.8	5.0	3.5	4.5 (s.d.=1.1)

Table 11.9: Experience of utilisation of skills

	Team Leader	Systems Anal.	Analyst-Prog.	Junior Analyst	Systems Des.	Program Des.	Overall
N	2	2	1	4	2	2	13
Time demands: (Scale: 1='ext. light' to 7='ext heavy')	6.0	5.5	5.00	5.5	5.5	5.0	5.5 (s.d.=0.8)
Technical demands: (Scale: 1='ext. light' to 7='ext heavy')	6.0	4.5	4.0	4.5	4.0	4.0	4.5 (s.d.=1.0)
% time below technical capacity:	35	40	-	20	45	25	31 (s.d.=2.2)
% time trivial or routine:	15	20	10	23	25	10	18 (s.d.=0.8)
% time show what you can do:	60	55	10	68	45	45	53 (s.d.=2.5)

Table 11.10: Experience of autonomy in carrying out tasks:

	Team Leader	Systems Anal.	Analyst-Prog.	Junior Analyst	Systems Des.	Program Des.	Overall
N	2	2	1	4	2	2	13
% time using own approach:	75	80	30	68	90	75	72 (s.d.=2.1)
% time making all important decisions:	75	45	50	43	80	55	58 (s.d.=2.5)
% time specs already agreed by others:	40	60	50	48	65	85	57 (s.d.=2.8)
% time every step has to be approved:	45	30	10	78	15	40	45 (s.d.=3.0)
% time given detailed instructions:	15	20	00	28	45	50	28 (s.d.=1.8)

Table 11.11: Impact and Visibility Of Contributions

	Team Leader	Systems Anal.	Analyst-Prog.	Junior Analyst	Systems Des.	Program Des.	Overall
N	2	2	1	4	2	2	13
% time involv. in business decisions:	20	40	00	10	10	45	21 (s.d.=2.2)
% time important contrib:	75	45	70	40	65	60	55 (s.d.=2.4)
% time contr. to business success:	50	45	100	43	70	25	48 (s.d.=2.8)
% time work of special interest to management:	55	70	100	33	45	55	53 (s.d.=2.9)
% time higher mgt doesn't know what ind. is doing:	50	15	10	17	10	30	23 (s.d.=2.3)
Freq. discuss work with higher mgt:							
Weekly				1			1
Monthly	1	2		2	1	1	7
Six monthly			1		1	1	3
Yearly	1						1
Don't Know				1			1

Table 11.12: View on Salary

	Team Leader	Systems Anal.	Analyst-Prog.	Junior Analyst	Systems Des.	Program Des.	Overall
N	2	2	1	4	2	2	13
View on salary, relative to comparable firms: (Scale: 1='much lower' to 7='much higher')	5.0	4.5	5.0	4.8	4.0	4.5	4.6 (s.d.=1.0)
View on salary plus perks, relative to comparable firms: (Scale: 1='much lower' to 7='much higher')	5.0	4.5	5.0	4.8	4.0	4.5	4.6 (s.d.=1.0)

Table 11.13: Views on Probable Timing of Next Promotion

	Team Leader	Systems Anal.	Analyst-Prog.	Junior Analyst	Systems Des.	Program Des.	Overall
N	2	2	1	4	2	2	13
Less than 6 months:	1			1			2
6 months - 1 year:		1	1	3	2	1	8
1 - 2 years:		1					1
2 - 3 years:						1	1
Over 3 years:	1						1

Table 11.14: Satisfaction with Promotion Prospects

Mean scores for satisfaction on a scale of 1 to 7 (4= 'about right') broken down according to position in the division of labour, and overall.

	Team Leader	Systems Anal.	Analyst-Prog.	Junior Analyst	Systems Des.	Program Des.	Overall
N	2	2	1	4	2	2	13
View on probable timing of next promotion: (1=much too soon, 7=much too long)	5.5	4.0	4.0	4.5	4.5	4.0	4.5 (s.d.=0.7)
View on probable size of next promotion: (1=much too small, 7=much too big)	3.5	3.5	4.0	3.8	4.0	4.0	3.8 (s.d.=0.5)

Table 11.15: Attitudes to Learning Advanced Software Engineering

	Team Leader	Systems Anal.	Analyst-Prog.	Junior Analyst	Systems Des.	Program Des.	Overall
N	2	2	1	4	2	2	13
View on learning new techniques, on a scale of 1 to 7: (1= 'very opposed', 4= 'indifferent', 7= 'very keen')	5.5	7.0	7.0	5.7	5.0	6.0	5.9 (s.d.=1.3)

(a) Skill Utilisation

Table 11.9 shows that on average individuals experienced themselves as under 'above average' time demands, whilst technical demands were perceived as 'average.' No qualitative data gathered during interviews suggested that time demands were a subject of resentment, still less overt conflict. Direct producers regarded the amount they had to work as to be expected in a job of that kind, and there were no experiences of under-utilisation. The picture of moderate technical demands is born out by the fact that on average they felt they were given 'the chance to show what they could do' for only about half the time. Only one programmer experienced slightly less than 'average' technical demands. A different programmer and four analysts considered that they were given the opportunity to show what they could do for less than half the time, but each felt under 'average' or 'above average' technical demands. Overall, there appears to be little evidence of serious tensions over lack of skill utilisation. The view of one analyst,

> "There's a good balance between interest in the job and in people here,"

would appear to be typical. However, given the importance attached to utilisation of skills, the moderate technical demands suggest one factor behind the merely moderate levels of overall goal satisfaction.

(b) Control over work

Table 11.10 shows a rather complex picture in terms of the extent to which different individuals controlled the conduct of their work. On average, they felt that they had the 'freedom to use their own approach' for more than two thirds of their time. As one analyst put it:

> "I feel I have a lot of control over my projects here."

Significant exceptions were the analyst-programmer in the Accounting group, who felt this to be so for less than one third of his time, and one junior analyst. The sample on average felt they were making the most important decisions most of the time (almost 60%). Three staff felt that they were making the most important decisions for less than half the time, but none felt they controlled the approach to work for less than half the time. Overall, control of work methods does not seem to have been an issue. Although tasks were circumscribed to a considerable extent by specifications (over half the time on average), the experience of working to specifications does not appear to have undermined perceptions of having control over work. Specifications were accepted as an inevitable part of the job.

Junior analysts felt on average that every step they took had to be approved by someone else for almost 80% of the time, which was significantly above the 45% average for the sample. The most junior team members appear to have had to seek regular approval for the choices they made in exercising discretion as to how they conducted their work. Qualitative data indicated that junior analysts tended to feel that they were not getting enough formal training, and were not given sufficiently clearly-defined areas of responsibility. There is then some evidence that junior analysts were not being given as much responsibility in carrying out the work as they would have preferred.

One senior analyst found control of his work restricted for a particular reason that was not connected to systems of control and coordination within Infoservices. The division with which he did a lot of work had become seriously dissatisfied with Infoservices' level of charging and had set up its own small data processing department, complete with systems analysts. They had designed some of their own new systems. From the point of view of this particular analyst, the most worrying development was that they gave him extremely detailed change requests for minor enhancements to systems that Infoservices had developed. He regarded this as a much greater threat to his job control than

any new techniques or reporting procedures that his own management might introduce. As a consequence he was keen that Infoservices should move over to providing package solutions, in order to make their services 'fairer' to users, and to restore his control over the process of requirements analysis.

(c) Experiences of Structured Techniques

Qualitative data on individuals' experiences of structured techniques suggest that experiences of working to standards also did not in general affect feelings of being in control of the approach to work. Whilst none of the analysts interviewed had any direct experience of structured analysis, they shared a positive orientation to structured programming. Some merely accepted it as the way things should be run; others considered it a necessary standard approach to coding for a large installation, that it made programs easier to read and modify, and eased communication between analysts and programmers. Only one felt that it made coding more boring. Only the analyst-programmer from the Accounting Group felt that there was a problem in the general area of programming standards. He felt there were too many standards, that some were useful and others not, and that as a consequence they were often ignored, particularly when there was pressure from above to finish a job. Several staff shared the view that structured programming standards and documentation practices needed to be enforced more.

The four programming staff interviewed were divided in their attitudes to structured programming according to whether or not they were using it. The two in the Manufacturing Group who had been trained in the technique and were using it felt that it was in principle a good idea, because it made code easier to read and maintain. One felt that it made design work more difficult and longer, but that it was worth it. However, she felt there was a problem in that different people did not as a rule arrive at the same design or code using the technique, given the same

specification. It had not eliminated personal variation. Moreover, it was not used in rectifying faults, so systems tended to lose their structure over time. Both were clear that the technique had no effect on the extent to which their progress was made visible, or on the way in which they were managed. The two in the Accounting Group had not been trained in it, but felt they had seen what it was about. They saw it as something with which Infoservices management wanted to make programs more standard in style, so that they were easier to maintain. They felt that it was just a formalisation of what any good programmer did, and was too rigid for complex systems. They also felt that it wouldn't work as a way of making code easier to maintain, because people wouldn't conform to it.

(d) Visibility and Impact

Table 11.11 shows individuals' perceptions of the extent to which their work was visible and contributed to business performance. They felt they were 'involved in decisions affecting business success' only to a minor extent. At the same time they felt they were 'making an important contribution', 'contributing to business success' and undertaking 'work of special interest to higher management' for about half the time on average, with considerable variation from individual to individual but no significant variation between groups. Three individuals, all analysts, felt that they were making an important contribution for less than half the time. One was the senior analyst who felt restricted by the data processing staff in a user division, another a trainee analyst who felt she was not receiving proper training. Six individuals, distributed across all groups, felt they were making a contribution to business success for less than half the time. Given the importance attached to 'making a contribution' and 'contributing to business success' described in Chapter 7, it is likely that these experiences were a significant source of tension for the staff concerned.

(e) Pay and Promotion

Table 11.12 shows that almost all those interviewed considered that their pay and benefits were rather higher than what they could expect in a comparable job in another company in the area. This area, too, would not appear to have been a site of tension or conflict.

According to Table 11.14, all those interviewed were basically satisfied with prospects for their next promotion. Table 11.13 indicates that most expected to be promoted within the next year, corroborating the impression given by Infoservices managers that regular promotions were the rule, and an important element in maintaining commitment.

11.4.3 Overview: The Nature of Conflict Between Individual and Organisational Priorities.

The moderate levels of overall goal achievement, skill utilisation, control over the approach to work, and satisfaction with promotion prospects indicate a basic harmony between the form of organisation and control of work and individual aspirations. An acceptance of 'professional' labour control by both managers and producers avoided explicit tensions over the intensity of work required, or over the use of specifications associated with action planning. There is some evidence that the introduction of structured techniques brought out tensions between individual desires for autonomy in the approach adopted to tasks, and organisational needs for code that was easily read and modified. These fears appear, however, to have been allayed by actual experience of the technique. Neither specifications nor structured techniques were experienced as an attack on the control of work of direct producers, in the sense of shifting the basis of control of individual performance away from 'professional' responsibility. Whilst they did involve adherence to standards, structured techniques appear to have been accepted as simply a good way of going about producing software.

There were, however, some tensions. It appears that work in S & P offered only satisfactory levels of stimulation and responsibility, except perhaps at team leader level. This sheds light on the Manager of Technical Practice's preference for ex-TOPS trainees as opposed to graduate recruits, because of their lower expectations, and his view that work in S & P required responsibility and expertise, but did not offer great variety in terms of technical challenges. There were also feelings on the part of a significant minority of staff that their work did not have sufficient visibility in terms of making a contribution to organisational performance. Overall, the impression is of staff who felt moderately extended, but who felt that the work they were doing was rather 'backstage'. This may well have been a response to the degree to which they were occupied on minor enhancements and maintenance, rather than new developments.

There were also some tensions concerning a lack of responsibility given to junior analysts. Unfortunately, no interviews were possible with junior programmers, to establish whether the same was true for them. Such tensions seem to have stemmed from the way that senior direct producers, i.e. team leaders, allocated portions of work to junior team members within a 'professional' paradigm of labour control, rather than from a replacement of 'professional autonomy' with behaviour formalisation and labour market segmentation. It would appear that junior staff could expect more responsible and challenging tasks as they gained experience and promotions.

More senior team members appear to have derived challenge from their overall responsibility for systems, and from having direct contact with the user and defining their needs. This, however, depended on Infoservices maintaining its position as supplier of software services. This is illustrated by the case of the systems analyst who found himself receiving completed requirements specifications direct from the user. The possibility of user divisions employing their own data processing staff appeared to pose a greater threat to the job control of

some direct producers within Infoservices than any moves on the part of Infoservices managers. For direct producers to have the desired control over their work it was necessary for Infoservices to be operating in such a way that it was clear that it, as an organisation, had control over a certain sphere of operation.

11.5 INITIATIVES AND POSSIBLE OUTCOMES

This section draws on data obtained from team leaders and managers at various levels as to strategies for change they were adopting or could foresee. It explores the consequences of these for the work of direct producers and their job structures, and indicates that none of the initiatives amounted to a shift in the basis of labour control away from the current form of 'professional autonomy'.

(a) Diversification in Application Areas

In terms of product strategy, whilst Infoservices management were not aiming to expand the scale of their operations, they had identified one particular area for expansion of the scope of their operations. This can be seen as a bid to safeguard the Division's position given the uncertainties of their role in providing more traditional accounting and manufacturing data processing. It concerned supplying data processing services to divisions engaged in developing 'product software'. This had potential consequences not so much for the nature of work for software producers within Infoservices, but for those working in 'product software' divisions.

The general manager felt that the management of product software divisions placed a great deal of emphasis on the technical sophistication of the software that went into their products, but were relatively primitive in terms of their use of computers

within their own production process. They failed to use the computer to document their design and code, provide other development aids, or keep records of what had been developed, customised, sold and shipped. They were naive in issues of data security against unauthorised access and accidental damage. He felt that Infoservices should take responsibility for managing the interface between design (i.e. development) and manufacture of product software. It could bring to bear greater sophistication in issues of documentation and data security from the longer-established data processing field.

Infoservices management were aware that this was a contentious issue with product software divisions and that their objective would be difficult to achieve. Managers in product divisions tended to see the Infoservices approach as an attempt to appropriate control over an area of their internal operations.

(b) Further Use of Structured Programming Techniques

The manager of S & P intended that there should be a greater use of more rigorous structured programming techniques and other standards for system design and programming. A basic reason for this was that structured techniques led to better software and development projects that were easier to manage.

The introduction of new programming techniques was the responsibility of the Manager of Technical Practice. He saw the development of a separate maintenance function within his group as closely linked with the adoption of more rigorous design, programming and documentation standards. As well as leading to better quality developments, these standards were to provide the basis for systems to be passed on to the maintenance group once they had been developed. He linked structured programming standards with the introduction of other formal procedures - such as one for 'release control', by which a development group was supposed to conclude a development by documenting exactly what had become operational and what was being handed over to

Technical Practice for support.

The manager of Technical Practice attributed slow progress on new techniques and procedures to two factors. Most of his resources had to be devoted to maintenance, and there was considerable resistance within development groups to 'being told what to do'. It was difficult to convince experienced practitioners that formal procedures were an improvement on the dictates of their experience. Influencing development techniques and procedures would be a slow process.

It was also difficult to persuade development staff to adopt procedures associated with giving up maintenance work. In spite of the release control procedure, contact between developers and users continued, and developers retained access to the actual code running. They often persisted in making minor enhancements, without the knowledge of Technical Practice. The Manager considered that the interface with development groups was still the major issue in controlling the work of his group.

In terms of the impact of these initiatives, on balance it appears unlikely that there would in fact be serious resistance to new techniques, or that they would have much effect on the viability of the existing 'high trust' form of labour control. On the one hand, the Manager of Technical Practice was aware of resistance to more rigorous standards, and the Accounting Programming Team, who did not use structured programming, appeared to be against the techniques. On the other hand, Table 11.15 shows that direct producers interviewed were on the whole keen to learn new techniques. The evidence from staff already using structured programming techniques was that they were generally enthusiastic about the quality benefits, whilst recognising that some coding operations had become more routine. The techniques were not being introduced to undermine the principles of job control established, and it appears likely that programmers would come to an accommodation with the degree of rule compliance implicit in them.

The issue of whether or not the manager of Technical Practice would succeed in using structured standards as a way of separating maintenance from development work is harder to judge. The S & P applications group managers did not appear to share his commitment to taking maintenance work completely out of their groups. This would appear to be a situation where middle managers differed in their views as to how work should be organised. The view of the manager of Technical Practice would appear to represent a concern with the control and coordination of maintenance work which was at odds with the preferences of development staff, who apparently derived satisfaction from doing some follow-up maintenance work. This is then an example of a tension over issues of job control that was not directly linked to management desires to improve levels of individual performance. It stemmed from a particular view as to how work could be best coordinated.

Even if the manager of Technical Practice were to succeed in separating out maintenance completely, it seems unlikely that this would mean a departure from a 'professional' principle of labour control, or a radical reshaping of job structures. The effect would appear to be a concentration of maintenance work with trainees in his group, but no other restriction of the job control of experienced producers, nor any segmentation of career structures.

(c) Introduction of Structured Analysis Techniques

There was interest from both management and direct producers in introducing structured analysis techniques. Both were concerned with the problem of the unreliability of user specifications, and saw structured analysis as a way of improving communication with user staff. The manager of S & P and the Accounting group manager both expressed the intention of choosing a structured analysis technique and training staff in it in the near future. The Manufacturing group manager was more circumspect. He felt there

were issues in determining which structured analysis technique would be right for Infoservices. It would also be necessary to wait for a new 'green field' development, since it would be difficult to apply a structured approach in enhancing a system which had not been designed according to the technique.

On the side of direct producers, at least one analyst team leader in the Accounting Group had been experimenting with a particular structured approach, Yourden Structured Analysis. He was doing this on his own initiative and only for his own work. He saw this technique for representing the functions to be performed by a system as a way of communicating more precisely with users. He regarded it as a complement or enhancement to his non-codified skills rather than as a substitute for them. This individual gave a clear indication of the conditions under which analysts would accept new techiques. He felt they would accept and use them provided they were seen by management as a tool available to software producers, rather than a set of procedures that had to be followed. This suggests that the acceptance of new techniques might revolve around the extent to which the producers concerned were able to participate in their selection and introduction, so that they felt they 'owned' the new technique.

This sheds some light on the resistance to procedures reported by the manager of Technical Practice, the reluctance of the development groups to take up more formal procedures that were being designed by people working in a separate section.

Overall, the conclusion is that structured techniques for analysis or programming were not intended as a threat to the 'high trust' principle. Whilst they were associated with increased prominance for rules in the development of software, it was possible for producers to see these rules as compatible with having control over the way they did their work.

(d) Increased Use of Software Packages

The Infoservices general manager, the manager of S & P and the Accounting group manager all saw increased use of software packages as vital for Infoservices' future. They saw Infoservices as facing competition on cost grounds from external suppliers using packages. The solution was to buy in and customise packages themselves, rather than attempt to increase the productivity of Infoservices staff working on 'bespoke' developments. As well as 'conventional' packages, written in 'procedural' languages, it was necessary to look at the options in terms of 'parameter driven' or 'user friendly' packages based on '4th generation languages', which users themselves might be able to customise.

The manager of S & P saw an increased amount of package-based development work as having a much greater impact on the organisation of work than greater use of structured techniques on their own. Within five years, he saw a smaller development work force, most of whom would be analyst programmers, responsible for assessing user needs, selecting a package from the market and carrying out all stages of any necessary customisation. He saw these staff as being on average more highly paid than current development staff.

This development would again not seem to imply any threat to the 'high trust' principle. It was consistent with staff having control over the way they did their work, and responsibility for substantial tasks. Indeed, it would appear to have some advantages in terms of the way it would give individuals responsibility for all stages of a complete system development. This vision of the future would however involve the redrawing of occupational boundaries.

It would require analysts and particularly programmers to acquire new skills and give up existing ones, and identify with a new mix of tasks. Rather than leading to a vertical segmentation of career structures, it would appear to lead to the removal of the separate programmer career structure. It is conceivable too that

the removal of the need for considerable volumes of fairly predictable programming work might lead to decisions to recruit at a higher level than the TOPS trainees favoured by the manager of Technical Practice. However, it is also likely that those senior staff who had a great deal invested in the area of applications programming, such as the manager of Technical Practice, would find ways of retaining a separate career path.

Overall the possible effects of the introduction of packages indicate the ways that the content of professional technical work can be shaped by a combination of product and process innovation, which increase productivity without leading to direct management control of 'routinised' areas of work and labour market segmentation. Packages effectively lead to an automation of those parts of the task environment - application programming - that might be simplified by structured techniques. Rather than a move toward a 'machine bureaucracy', they lead to a redrawing of the task boundaries of an organisation based on 'professional labour control'.

(e) Increase in 'Staff' Positions for Software Producers

In the years leading up to the period of fieldwork, there appears to have been an increase in the proportion of software staff in support roles within Infoservices, as opposed to those working directly on development or maintenance work, or in line management. There were three reasons for this. First the priority on relevance of systems, concerns with looking at corporate-wide data processing needs and taking into account issues of integration, led to a need to have a specialist staff assessing the overall picture and planning future developments. Second, the emergence of packages meant that some staff had to be occupied in a staff role of surveying the market for packages and assessing them. Finally, there were the staff roles in Technical Practice concerned with implementing new techniques and development methodologies.

The four preceeding change strategies imply a continuing or growing need for such 'staff' positions. Whilst their emergence was apparently not associated with any basic undermining of the 'high trust' model by which direct producers were controlled, there was however evidence of one important area of tension.

This concerned an attempt by the management to give the responsibility for feasibility studies for development projects performed within S & P to the Systems Evaluation group within Systems Support. The rationale was the perceived need to control quality and viability of new systems from the earliest stages of their development. Infoservices senior management saw the new Systems Evaluation group as developing more rigorous standards and expertise in this respect.

In practice, S & P team leaders and systems analysts were not willing to accept the specifications and estimates produced by the new group, arguing that the staff concerned were not capable of producing accurate estimates. As a result, feasibility studies were returned to them. This resistance to the removal of the job of performing initial feasibility studies and estimates suggests that the control of initial estimates was a vital part of the way team leaders maintained control over their work and their ability to set targets for their staff which they could control in a 'high trust' mode.

CHAPTER 12

CONCLUSIONS FROM THE CASES

INTRODUCTION

This chapter examines the data from the cases to see what generalisable conclusions can be drawn in answer to the five sets of research questions set out in Chapter 6. In doing so, it is recognised that the base of data from four organisations and 68 software producers is slight. However, it is in fact possible to argue the generality of a range of conclusions. These are advanced in effect as hypotheses, drawn from exploratory work in a little-studied area, to be tested by future empirical work on a wider scale.

The first section addresses the questions concerning the nature of task environments for software-producing organisations and their relationship to business strategies. The second section concerns the second and third group of research questions. It deals with the nature of divisions of labour and career structures in software production, and draws conclusions as to the current state of software occupations. It also looks at the organisational structures, mechanisms of control and co-ordination, and mechanisms for producing commitment employed within software-producing enterprises, and examines how these phenomena can be understood as being shaped by the nature of organisational task environments.

The third and fourth sections focus on the last two sets of research questions, which are the main concern of this thesis. The third section concerns the nature of conflict that occurs in the context of the organisational arrangements described. The final section draws conclusions as to how tensions and conflicts within enterprises on the one hand, and changes in business environments and business strategies on the other, are shaping the development of software occupations at the level of the enterprise. This leads to a wider discussion of how such an analysis of enterprise-level processes leads to an understanding of the lines of development of software occupations on a national scale, and to a consideration of the evolution of professional technical occupations in general.

12.1 BUSINESS STRATEGIES AND TASK ENVIRONMENTS

Strategic Options

All four of the organisations studied had competitive strategies which emphasised differentiating their products and services from those of competitors, rather than establishing cost-leadership on standard products. Whilst it cannot be assumed that this is true of all software-producing organisations, it is consistent with an image of the industry as perpetually in the throes of technical change and product innovation, where product models or paradigms are rarely stable for long enough for suppliers to begin to compete principally on grounds of cost.

The case studies provide a basis for distinguishing two main types of differentiation upon which software-producing organisations, supplying products and services to businesses, can derive their competitive advantage.

First, there is innovative differentiation (Exco, and some sections of Telexco and Infoservices). Here a company seeks to develop new types of product, often involving areas that have not previously had computer systems applied to them, so that the products being developed are themselves innovations in the sense of the commercial application of technology. There are no established models or markets, or any well-established competitors, but potential for sales in the form of ground-breaking software products which can, in Porter's (1985) terms, add value to customers' operations. Such software-producing organisations seek competitive advantage by being amongst the first in expanding new markets.

Second, there is differentiation in established markets on the basis of quality. Here, a software-producing organisation markets products or services in application areas that are well charted, and seeks to attain competitive advantage on the basis of delivering superior quality without putting itself at a cost

disadvantage to established competitors. It can attempt to increase sales on the basis of a standard price and superior quality, or, more likely, use its quality reputation to charge a premium price. It can establish a quality reputation within a well-defined market niche or set of niches, for example by designing software packages for a particular type of user organisation and then customising the packages to suit site-specific needs, and providing comprehensive back-up and support services (Sysco, and for the most part, Telexco). Alternatively, it can seek to achieve differentiation across a broad range of related application areas (Infoservices).

The cases clearly illustrate how strategic options are shaped by what stage product market life cycles have reached. They show first of all that, in most industries, emerging product markets lead to innovative differentiation. More interestingly, they suggest that mature product markets in software are characterised by competitive strategies of focus and differentiation on the basis of quality, i.e. a concern with meeting customer needs in particular niches, rather than cost-leadership on standard products. Arguments supporting this as a general hypothesis are that the very tractability of software products invites customisation, and that organisational users in general have a considerable investment in existing procedures and configurations for their activities. They are likely to consider closely the appropriateness of a system and the value it will add to their operations, and are likely to be prepared to pay a premium price.

The cases also show that there can be different emphases in the type of differentiation being sought within one organisation, if it is dealing with two or more product markets at different stages of maturity. Part of Exco engaged on Project X was coming to terms with the imminent emergence of new product configuration and markets in office automation, whilst the rest of it was concerned with differentiation on the basis of quality in an 'ageing' market for message-switching systems.

Infoservices, too, can be seen as combining an emphasis on customisation and quality using existing technologies with strands of differentiation through technical innovation. In meeting the data processing needs of a large corporation, it had to operate in many different markets for software products. In seeking to do so in a way that was more attractive than external competitors, it had to supply some systems for which there were mature product configurations, and others where there was rapid innovation, involving '4th generation' languages, for example. There is here an indication that corporate data processing departments have to cope with both sorts of differentiation simultaneously, or else arrive at an arrangement by which, for example, the more innovative products are purchased from outside.

The case studies also suggest that software-producing organisations selling on an open market deal with their environment by competition rather than any form of environmental management. Infoservices is distinctive amongst the four cases in that it sought to influence the nature of demands on it through various political fora within the corporation it served. It was the only one where some form of environmental management was used to shape the way customers saw their needs, so as to make it easier for them to be met. This was an option available to it precisely because it was an 'in-house' department. It was possible for managers to fashion channels of influence to a set of decision-makers who controlled their 'market'.

The Task Environments Shaped By Business Strategies

In considering how business strategies shaped task environments in these four organisations, a number of different aspects of both of the basic dimensions of 'complexity' and 'unpredictability' are apparent. These are illustrated by the characterisation of task evironments for each case shown in Fig. 12.1.

Fig 12.1: Business Strategies and Task Environments

	Business Strategy	Task Environment
Telexco	Focussed differentiation for established products. Innovative differentiation and collaboration for new products.	Established Products: Technically: Complex and Stable. Commercially: Complex and Stable. New Products: Technically: Complex and Unpredictable. Commercially: Complex and Unpredictable.
Exco	Differentiation through innovation, new products.	Technically: Complex and Unpredictable. Commercially: Complex and Unpredictable - 'Production Smoothing' problem.
Info-services	Focussed differentiation through quality, with established products.	Diverse Technically: Complex in all areas, Unpredictable in some. Commercially: Complex and Unpredictable.
Sysco	Differentiation on basis of quality and innovation, Environmental Management.	Technically: Complex and Unpredictable in assessing user requirements, Stable from system design onwards. Commercially: Complex and Stable

The cases suggest first that complexity in the task environment of a software-producing organisation generally has both 'commercial' and 'technical' aspects. Commercial complexity stems from the knowledge needed to understand the general nature of user needs and the relevance of products and services offered by the organisation to them, so that goods and services can be marketed and enquiries from potential customers turned into

orders. Technical complexity stems from the knowledge of the principles of requirements analysis, software design and programming, and installation, needed to develop or customise software systems once a sale has been made or firm interest established. Both commercial and technical complexity are apparent in all four cases.

The cases also indicate that task unpredictability has 'commercial' and 'technical' components. Commercially, there is unpredictability on the one hand in the nature of what is required by users (Exco, and Project X within Telexco). This aspect of unpredictability determines the frequency of situations in which complex commercial knowledge has to be applied. On the other hand, there can also be unpredictability in the level of demand (Exco, Infoservices). The first aspect of commercial predictability would appear generally to imply the presence of the second. Technical unpredictability takes the form of novelty of technical problems encountered in producing software, i.e. frequency of 'exceptional' circumstances in which complex technical knowledge has to be applied (Exco, Project X within Telexco).

Although no sophisticated measures were used to assess the degree of task unpredictability in the four organisations, the descriptions of organisational tasks in the cases provide a basis for identifying variations in the type and degree of unpredictability experienced. These can be related to different emphases in business strategy.

There is a clear suggestion that a strategy of innovative differentiation leads to considerable unpredictability in both technical problems encountered and in assessing user needs in the market place (Exco and part of Telexco). Moving into new products and new markets brings technical and commercial uncertainty together. Exco, in particular, faced not only novel technical problems and novelty in assessing user requirements, but also unpredictability in the volume of work it could expect. Together

with the large size of projects, this meant there was a problem in matching its resources to demand.

In contrast, a strategy of focussed differentiation in a mature market would appear to lead on the whole to greater commercial and technical stability (Sysco and part of Telexco). The volume of demand is fairly predictable, as are the sorts of technical problems requiring solution. In the case of Sysco, one aspect of technical unpredictability remained, however. Stable demand and a fairly predictable set of programming tasks still left room for considerable technical unpredictability in the process of defining user requirements, making this the most uncertain aspect of the organisation's task environment. It would appear that a strategy of focusssed differentiation in a mature market can be associated with considerable technical unpredictability.

An understanding of the type of unpredictability experienced by Infoservices lies along slightly different lines. It was serving a corporation with many divisions competing for attention. The fact that its services were requested through a political system of resource allocation rather than a market meant that, of the four organisations, it had to face perhaps the greatest unpredictability in demands in terms of what the priority projects were. The degree of demand unpredictability faced by a corporate data processing department clearly stems from the character and organisational politics of the channels through which requests reach it, to some extent regardless of whether or not it is dealing with new or well-estabished types of computer system.

Infoservices is also unique amongst the four in that its task environment does not seem to be adequately characterisable simply in terms of complexity and unpredictability on their own. With its range of activities in the service of many customer divisions, it dealt with many different levels of unpredictability in technical problems and in understanding user requirements, according to the novelty of the application area

being considered. It can be considered as having a task environment that was diverse as well as complex and unpredictable. The range of demands from the corporation it served meant that Infoservices effectively had a business strategy of competing with outside providers over a considerable range of products and services. It illustrates that 'strategic breadth' in a software-producing organisation leads to task diversity.

12.2 ORGANISATION, JOB STRUCTURES, CONTROL AND COORDINATION

12.2.1 Divisions of Labour and Career Structures

Our concern here is with how organisational structures and the various software specialisms with which they were staffed can be understood as responses to task environments, and with drawing conclusions as to the nature of career structures in software production. This is essential background for understanding the nature of conflict experienced by software producers. It also permits certain conclusions to be drawn as to the current character of these occupations, in particular whether or not there are separate 'professional' and 'technician' career paths.

Within the departments responsible for the production or customisation of software, there were basic differences in terms of organisational structure, divisions of labour and career structures between the data processing and telecommunications software organisations. These are summarised in Fig. 12.2

Figure 12.2: Divisions of Labour and Career Structures

	Divisions of Labour	Career Structures
Telexco (Telecoms)	Product-based units within software production dept., comprising 'designers' and 'programmers'. Software staff also within Marketing department.	Staff are graduates in mathematical disciplines. Promotion from 'programming' to 'design'. At senior levels, choice between marketing and development posts. Otherwise, only one career structure. Closed internal labour market.
Exco (Telecoms)	Product-based units within software production dept., each with design team and several programming teams. Software staff also within Marketing and Planning Departments.	Staff are computer science graduates. Promotion from programming to design. Single career structure, but no closed internal labour market.
Sysco (Data Processing)	Within Production Department separate pools of systems analysts and programmers. Software staff also within MArketing Department.	Staff from a variety of educational backgrounds, graduate and non-graduate. Promotion from programming to analysis, to management, in production or marketing. Single career structure, but no closed internal labour market.
Info-services (Data Processing)	Main software production department divided into groups responsible for different application areas. Each group has separate analyst and programmer teams, although also has analyst-programmer teams. Other departments also employ software staff, e.g. Planning, and in specialist development work.	Parallel career structures in general programming, systems analysis and specialist development work. Mix of graduates and non-graduates in general work, graduates only in specialist work. Within each structure, promotion from less to more responsible work. Managers and staff positions recruited from amongst producers. No internal labour market.

(a) Divisions of Labour and Career Structures in Telecommunications

Data from the two telecommunications software organisations, (Telexco and Exco) suggest the following characterisation of software workforces engaged in more 'technical' areas of software production.

They consist almost entirely of graduates, in computer science, maths, or else 'hard science' subjects with a significant computing element. Organisational structures within production departments consist of groups with a product focus. Within product groups, there are experience-based division of labour between 'designers', responsible for high level functional and logical design, and 'implementors' or 'programmers', responsible for low-level logical design and coding.

These two functions are not separate specialisms, however, but different stages in a single career structure. Transition from 'programming' to 'design' is earned gradually, as junior staff are given more and more responsible areas of design work, and marked by promotions in terms of job title and salary increases. More responsible positions are filled on the basis of experience rather than qualifications, either educational or professional. Membership of professional institutions does not appear to play an important role on the way to seniority.

If a firm is engaged in areas where skills are transferable (Exco), more senior positions are filled by people who had have gained their 'low-level' experience elsewhere as well as people promoted from within. When skills are more firm-specific (Telexco), senior positions are more likely to be filled from an internal labour market. The cases suggest that differences in the degree to which skills are firm-specific stem from the degree to which business strategy is concerned with innovation. A firm engaged in very innovative activity (Exco) will need more generic skills than one which has focussed on a particular type of product (Telexco) over a number of years.

The picture of software occupations in a 'technical' area such as real-time telecommunications is one of a single 'professional' job structure within a software-producing organisation. There are no less-qualified 'technician-level' producers to perform the simpler development tasks.

(b) Organisation and Job Structures in Data Processing

The two data processing cases suggest a rather different and less uniform characterisation of software-producing workforces in this kind of application area.

Data processing organisations utilise staff from a variety of educational backgrounds, graduate and non-graduate, with only a minority who have specialised in computing in some form during higher education. Most acquire skills in producing data processing software skills once they get a job in the field. They do so through brief initial formal training provided by an employer, but above all through experience.

In data processing workforces, there is a basic division of labour between the work of 'systems analysis' and 'programming'.

Systems analysis involves the study of user requirements and high-level design of systems in terms of specifying the functions to be performed. Programming involves logical design and coding of systems. This distinction between two specialisms can be understood as the separation of two different types of complexity inherent in developing data processing software. A major part of the task is to analyse very complex existing procedures within a particular realm of the user organisation that is to be computerised, and to prepare the organisation to receive a computer system that will have a substantial impact on its functioning. These activities are arguably much more complex than for most telecommunications software systems. They are also rather different in character from the production of logical

designs and codes to meet requirement specifications. This is arguably a simpler task than for real-time software such as that involved in a telecommunications switching system. The transformations made on data in commercial data processing applications such as accounting or payroll are basically straightforward in nature, whilst those made in a telecommunications system are very complex.

The cases indicate, however, that, more than twenty years after programming and systems analysis first appeared as separate specialisms, there is still no definite boundary between them, nor are they clearly separate career structures.

In one case (Sysco), staff progressed from low-level programming work to more responsible high-level logical design work, and then to systems analysis. Programming and systems analysis were different points on the same career path rather than distinct career structures. In the other (Infoservices), some people transferred from programming to analysis, but it was also possible to gain seniority as a programmer, or to start on the analyst ladder as a trainee analyst. Programming and systems analysis were distinct, parallel careers.

The origin of these different career structures can understood as follows. Managers either find a lack of ready-formed data processing staff coming from the educational system with skills that can be utilised immediately, or else choose not to trust the 'raw product' of vocational educational and training institutions. Instead, they take staff with a variety of backgrounds and levels of experience and develop their skills in a direction which they see as meeting the particular needs of their business situation, promoting them as they gain expertise. This results in career structures in different firms that emphasise different skills at senior levels. Differences stem from different perceptions of what is required by the task environment.

An emphasis on 'career programmer' grades appears to be linked to perceptions that ease of maintenance of software once developed is vital, that structured programming is the key to this, and that structured programming increases the knowledge needed for high-level logical design (the Technical Practice Manager within Infoservices). A typical view taken by programming managers in such circumstances is that the continuation of separate specialist programmers and a programming career path are key to the development of expertise in structured programming and effective maintenance of software.

In contrast, analyst managers (such as the Accounting Manager within Infoservices) may see systems analysts as becoming involved much more in the customisation of bought-in packages, rather than the specification of new in-house systems. This is seen as decreasing demand for specialist programmers, on the grounds that there is less programming to do. Systems analysis skills in specifying customisations then become much more crucial to establishing competitive advantage. Under such circumstances, programming may become part of the responsibility of staff who are primarily analysts (Accounting Group within Infoservices). Alternatively, programming may become a 'training grade' that staff pass through before becoming systems analysts (Sysco). In both cases, no need is seen for a specialist programming career path.

Whatever the differences between the degree to which programming and systems analysis are distinct, the two data processing cases suggest, however, that, like telecommunications software production, data processing software production is characterised by an absence of vertical segmentation of career structures. Senior people came to be so through passing through junior positions. People with higher education qualifications in applied computing may expect to advance more rapidly, but all have to start as a trainee programmer or analyst.

This absence of vertical segmentation suggests that the analyst and applications programmer career paths within data processing organisations should both be characterised as 'professional', because there are no technicians. However, the presence of a sizeable proportion of non-graduates in the main development workforces of both Infoservices and Sysco, coupled with the preference of one Infoservices manager for recruiting non-graduates because they aspired to less challenging work than graduates, suggest that both these career paths have elements of being 'technician-level', even though they were characterised by 'professional responsibility' as a form of labour control.

This impression is further strengthened by the existence within Infoservices of a separate career structure for the apparently more technically-demanding work of developing and maintaining databases, with higher educational standards required for entry. However, to characterise the specialist career path as 'fully professional' and the mainstream careers as 'technician-level' is unsatisfactory. Systems analysts and applications programmers do not merely perform less responsible tasks assisting more highly-educated database programmers. The latter supply specialist support work to the former.

The general picture emerging is of, on the one hand, a career structure for systems analysts, where new entrants, not necessarily with a background in computing or a university degree, start by acquiring some applications programming skills. With seniority they attain considerable responsibility in the use of systems analysis skills and knowledge, concerned with applying computer systems to business applications, rather than in any technically intricate aspect of programming or software design. In some organisations, senior applications programmer posts may also exist, to meet a demand for considerable volumes of more complex types of applications programming, associated in particular with large in-house developments and the use of structured programming. On the other hand, alongside these career structures, there may exist more 'technical' lines of programming, for example in database work, or in systems software

generally. Both types of career structure would however appear to lead to management positions.

(c) Software Producers In Functions Other Than 'Production'

All the organisations studied indicated that the career structures described within the sphere of software production could also lead to positions in departments concerned with things other than direct production of software.

In organisations selling software systems on an open market, the complexity of dealing with initial enquiries and assessing the requirements of potential customers leads to the need for a separate marketing function. This is staffed by experienced software producers, with technical knowledge of software systems and products.

Large software producing organisations appear to be characterised also by the existence of further 'non productive' or 'staff' functions containing staff with software production backgrounds. In the case of Exco, a planning and control department and a software tools department had both been created within the two years preceding fieldwork, and appeared to be responses to unpredictability and complexity in technical aspects of large development projects. In Infoservices , the focus of one 'staff' department was also on relieving the degree of technical complexity faced by a large production department of over 60 producers, by providing up-to-date information on applications software packages that were available for each applications area.

In contrast, the Manufacturing Section of Sysco and Telexco both had small development staff numbers of around 30, and no specialist staff functions. The functions performed by 'staff' departments in the other two cases were however still provided, but by staff within the production department. 'Staff' positions were present in all of the cases, but not always within a department distinct from production.

In none of the four cases did these 'staff' positions correspond to separate software specialisms in the form of a distinct job structure. They were all filled by people who had gained their skills as software producers in the main production department. They provide examples of the diversity of jobs for experienced practitioners in software producing organisations, and an indication that software staff work in 'technostructure' positions, as well as in 'direct production'.

12.2.2 Coordination, Control and Commitment

The cases imply that control and coordination are achieved in software-producing organisations through a combination of action planning and 'professional' control of labour. The basic complexity of task environments in software mean that producers are trusted to use skills with discretion in meeting specifications and targets for extensive blocks of work, usually weeks long, allocated by a team leader. Systems of action planning can be seen as ways of turning the demands of the task environment into well-specified and reasonably independent pieces of work that can be allocated to responsible 'professionals'. Action plans buffer the work of direct producers from the various forms of unpredictability present in the task environment. Here we first of all examine the evidence from the cases as to the nature of systems of action planning and ancilliary coordination mechanisms found under different conditions. We then discuss the characterisation of 'professional' labour control emerging from the case studies - in particular the relationship between commitment to meeting project targets, supervision and adherence to procedures.

Coordination Through Action Planning and Standardisation of Skills

Different forms of action planning and the use of other modes of coordinating work reflect different types of unpredictability in the task environments of software-producing organisations.

Action plans are drawn up at two basic levels of detail in software-producing organisations: an overall schedule of projects and detailed specifications and targets for tasks within each project.

In terms of the higher level of planning, a stable task environment associated with a business strategy of differentiation on the basis of quality leads to a stable flow of work, and managers can set up a basic project schedule for several months at a time (Sysco and Telexco). The frequency with which the basic project plan has to be revised will also depend on the size of projects. Very large projects also lead to a more stable basic work plan (Exco). Unpredictability from month to month as to priorities for systems in relatively small projects is likely to lead managers to choose to use regular centralised replanning of the basic project schedule (Infoservices). For an environment that is commercially very unpredictable, associated with a strategy of achieving innovative differentiation, it may not however be possible to plan so that there is orderly passage of staff from one project to another. This creates a need to combine action planning with the use of slack resources in the form of staff engaged on low-priority activities during the wind-down phase of projects, in anticipation of the requirements of new projects (Exco).

In terms of how action plans are drawn up at a more detailed level, systems are in general specified and estimated at a number of increasingly detailed levels. There are a number of 'functional requirements' or 'system design' levels, and then a number of 'implementation' or 'programming' levels.

In data processing organisations the emphasis in this phased system of action planning is on dealing with the unpredictable nature of problems in assessing user requirements. It is a way of refining accuracy of the system's functional design before actual code is written, necessary because of the difficulty of getting accurate statements of user requirements at the outset.

Specifications and targets are also a basis for agreeing and refining contracts with clients or users. They are a medium for managing relationships with clients.

In telecommunications organisations, successive levels of specification are again concerned with dealing with the unpredictable nature of user requirements, and for drawing up contracts. They are however also concerned with the technical unpredictability of the processes of logical design and programming, which are in general greater than in data processing. Rigorous high-level design work is a way of confronting technical problems at as early a stage as possible.

These systems of action planning are all based on the use of some form of standardisation of skills, by which specifications produced by producers working on one level are made comprehensible to those working at the next. Standardisation of skills can however take a number of forms. Formal structured design conventions may be used to make high-level designs comprehensible to implementors (Telexco). Alternatively, there are less formalised types of standardisation for this level of work (Exco, Sysco and Infoservices). Informal design conventions are learned by staff as they progress through the organisation, drawing on design practices and notations commonly used within the industry, such as annotated flow charts. For ensuring that lower-level specifications are readily comprehensible, standardisation of skills in the form of structured programming is widely used. Standardisation of skills in the form of structured programming is also a key element in coordination between those producing systems and those maintaining them.

In terms of how low-level forms of action planning are shaped by the nature of business strategies and task environments, there is some suggestion from the cases that greater unpredictability either in defining user requirements or in the technical problems encountered during logical design and programming, lead to more 'centralised' specification. By this we mean that 'high-level' specifications are required to be done with greater rigour,

leaving less to be filled in and decided at subsequent stages. Telexco responded to a new task of increased technical unpredictability by increasing the rigour and effort put into high-level design work. Exco, dealing with great technical unpredictability in developing innovative projects, also concentrated its most experienced staff on high-level design work, and responded to quality problems by increasing the detail to which designers worked.

There does not however appear from the cases to be any evidence that the use of formal structured design for drawing up specifications is associated with more unpredictable tasks. Whilst managers in Telexco saw the use of formal techniques as important in such a context, managers in Exco did not. They concentrated on the expertise of staff involved in high-level design, rather than the formal rigour of the techniques used.

It is also significant that there is no sign of a move away from centralised action planning within projects with greater task unpredictability. As we have seen in Chapter 5, the work of Galbraith (1977) and Mintzberg (1979) suggests that greater unpredictability may make action plans unwieldy, and lead managers to prefer project-based groups which coordinate through mutual adjustment, with action plans drawn up only at very low levels. In contrast, the cases suggest that high levels of technical unpredictability were dealt with by centralised action plans, with a functional separation between those drawing up specifications and those implementing them (Exco, Project X within Telexco). Greater unpredictability was dealt with by putting more emphasis on the rigour of high-level plans, rather than putting 'designers' and 'implementors' on the same team, to coordinate by mutual adjustment. The preference was to keep more experienced 'high-level' staff in separate groupings from less experienced 'low-level' staff.

Residual unpredictability in assessing user requirements as well as in technical problems that might be encountered in all four cases appear to be catered for by systems of vertical information

collection and regular adjustment of low-level action plans. Vertical information collection consisted largely of regular but informal data gathering by team leaders. Adjustments were made first of all by renegotiating deadlines with users, and second by bringing slack resources to bear. This took the form, variously, of re-allocating staff from some 'background' activity, or from a less crucial project, or else the use of overtime.

'Professional' Control of Individual Performance

The form of labour control in software production can be characterised in terms of a number of elements, indicating the inadequacy of a simple 'professional autonomy' versus 'behaviour formalisation' continuum for describing forms of labour control. Discretion co-exists with rule-adherence, and high levels of perceived autonomy in the approach to work are also found together with close supervision, albeit of a particular kind.

(a) The Team Leader as 'Client': Project Reviews

Blocks of work are allocated to software producers by team leaders. As a 'professional', a producer's primary 'client' is his or her team leader - the agency to which they are accountable in using discretion to complete a substantial and complex piece of work. The cases show that project reviews have a central role as the place in which producers are called to account by their team leader, sometimes also by their colleagues or a customer representative.

(b) Selection

'Professional' control of individual performance is based on selection of the workforce. Software-producing organisations recruit people who have proven they have the necessary skills and commitment to work through gaining experience in similar organisations. The only exceptions are recruits to the most junior positions who are selected on the basis that they have

been educated to a level such that they are able to acquire necessary skills and orientations to work. Preferences as to educational backgrounds may vary. The cases suggest that selection policies match the technical skills and orientations of recruits to the level of complexity and unpredictability in an organisation's task environment. Firms engaged in innovative and highly technical projects prefer graduate computer scientists (Exco). Those working with estabished data processing technologies prefer people with a less technical computing background, who show some interest in business applications of computing.

(c) Skills and Commitment to Work

The team is significant in software production work organisation as a site where skills are imparted, and where a responsible orientation to project targets is learned.

In terms of skill acquisition, a common model is that of the less experienced learning through working on an assignment with a more experienced colleague. The role of the team leader in allocating progressively more complex tasks, and in providing regular help, is also of great importance. Opportunities to move on to progressively more advanced work and to receive formal training appear to be not only mechanisms for developing staff, but also ways for motivating them to adhere to project targets and stay with the organisation.

In terms of the reproduction of commitment, we saw in Chapter 7 that software producers have goals of being able to utilise their skills, of making the most important decisions in their sphere, of making a visible individual contribution, and of having an impact on organisational performance. The cases suggest that such goals appear to have been reproduced and mobilised into commitment to meeting project targets on the one hand through a process of socialisation within teams, for which team leaders see themselves as primarily responsible, and on the other through the

promise of advancement.

The basic mode of socialisation is regular informal contact between team leaders and team members, by which the former establish a commitment to meeting team goals in exchange for a promise of organisational advancement and the opportunity to develop skills further and take on progressively more responsible work. Career structures can be seen as functioning not simply as hierarchies of skill, but also trust. Individuals are offered advancement on the basis that they demonstrate adherence to norms of responsibility and productivity as well as the skills necessary for more complex, higher level design or systems analysis work. At the top of a direct producer career ladder is the position of team leader, where occupants are 'responsible' enough to set targets and specifications as well as execute them.

Opportunities to obtain better positions in other organisations are a further incentive for producers to utilise and extend their skills in meeting project targets. The only exceptions to the dual motivating effect of internal and external advancement opportunities occur where skills are firm specific (Telexco).

There is some evidence of a difference in emphasis in the techniques of socialisation used for junior producers between the telecommunications and data processing organisations. Team leaders in the former tended to emphasise less direct approaches, such as 'leading by example', drawing on already existing orientations towards performing well. In the latter, particularly Infoservices, team leaders referred to the need to assert norms directly. This may be because non-graduate recruits had not acquired the same strength of identity in being responsible and committed to their work as the graduate recruits found elsewhere.

(d) Rules and Standards

Labour control in software production is based on a combination of the exercise of uncodified expertise and a commitment to

achieving project targets with the observation of formal rules and procedures. Structured programming (all four cases) and structured design (Telexco) are seen by team leaders and managers as mechanisms that assure quality, consistency, readability and maintainability of work.

The picture presented by the cases is that structured techniques are not used as a basis for 'deskilling' or 'proletarianisation' of software producers. Their use does not coincide with vertical segmentation of career structures, the use of 'technician' grades, or a form of labour control based on 'rule adherence' as opposed to 'responsible autonomy'. Within a context of 'professional' labour control, structured design appears to function as a basis for drawing up specifications for large and complex systems, so that technical problems can be dealt with at as early a stage as possible (Telexco). Structured programming is a basis for ensuring compatibility and coordination of the work of programmers, as well as making quality predictable. Structured techniques were understood by managers and producers alike as techniques to be mastered in a context of responsibility and discretion.

Other types of rules and standards are important in assuring the quality of work in a similar way. In particular, producers engaged on customising standards software systems (Sysco and Telexco) have to observe design and coding standards associated with the systems they were working on.

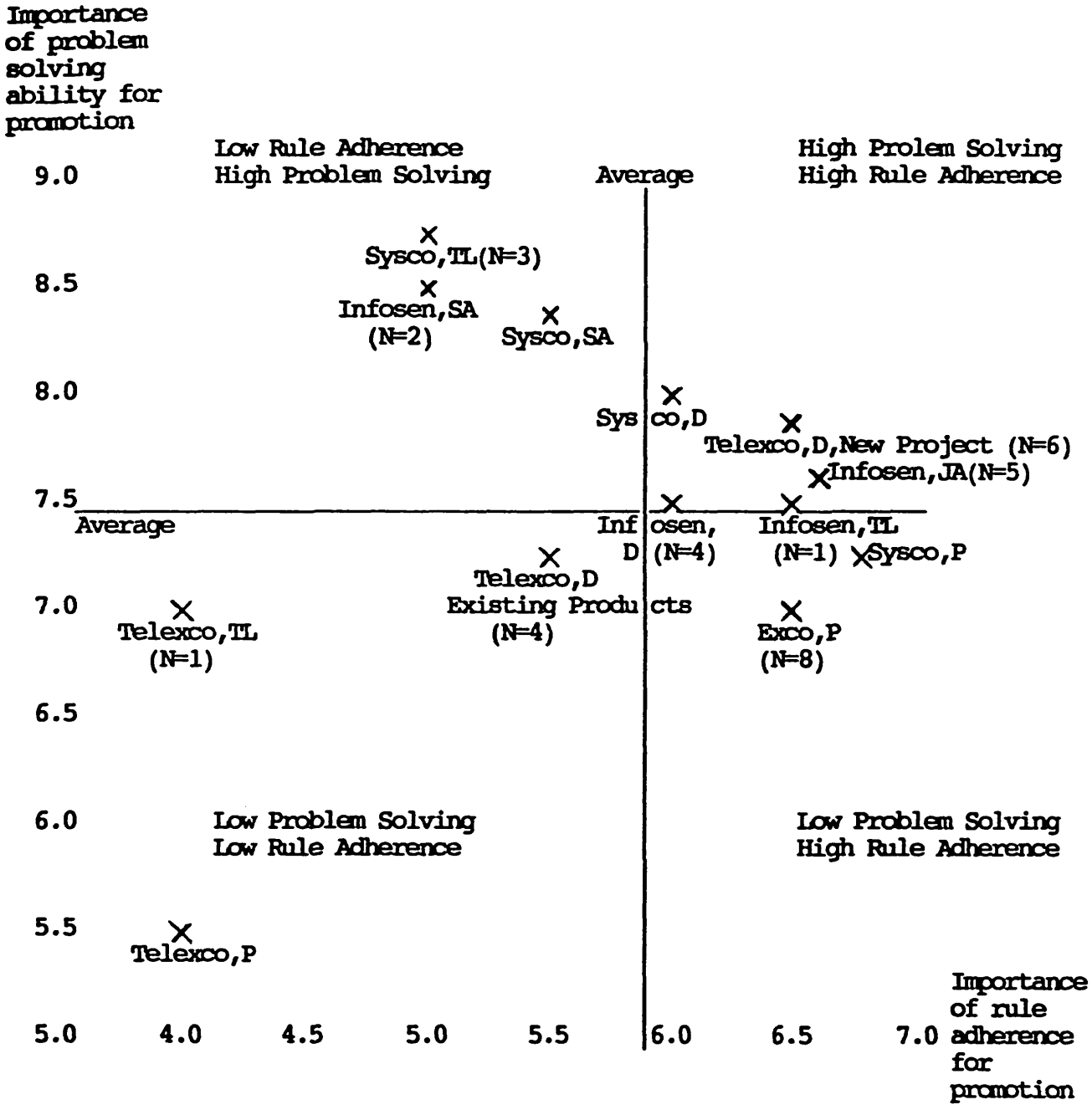
Figure 12.3 shows the extent to which 'junior' and 'senior' staff in the four organisations experienced themselves as being evaluated for promotion on the basis of their ability to solve problems independently, as opposed to their ability to observe rules. It shows that both senior and junior staff in all four organisations considered that the main criteria on which they were judged was their ability to use discretion in making decisions. However, they also saw rule adherence as important. Professional responsibility and commitment coexist with rule adherence. The balance of the two criteria appears to vary as

much within the workforce of a particular case as from case to case. There was no significant correlation between the two different assessment criteria. This suggests that the sorts of labour control found in software production vary considerably in their emphases.

It is possible to draw one admittedly tentative conclusion of a more specific nature from the diagram. It might be expected that senior staff in organisations with high task unpredictability which are using structured design as a basis for rigorous 'high-level' design should experience themselves as more rule-bound than those in organisations with more stable tasks, where less rigour is needed in drawing up specifications and designs. The perception of the Project X team members in Telexco, that they were being assessed on the basis of both their problem-solving ability and the extent to which they adhered to procedures, provides some support for this hypothesis.

Overall, however, figure 12.3 simply illustrates the inadequacy of a simple continuum, running from 'professional control' to 'bureaucratic control,' for characterising the nature of control of individual performance in software-producing organisations.

FIGURE 12.3:
IMPORTANCE OF PROBLEM SOLVING AND RULE ADHERENCE IN PROMOTION ASSESSMENTS



Key:

TL = Team Leader
SA = Senior Analyst
JA = Junior Analyst
D = System Designer
P = Programmer

Team leaders as linking action plans to 'professional' labour control

The cases also indicate some basic features of the way low-level action plans are drawn up within projects and the way targets and specifications are allocated to producers that appear to be crucial aspects of the way action plans 'mesh' with 'professional' control of labour.

First, software production team leaders are involved in setting estimates, usually in collaboration with some representative of a marketing function. Negotiations between these two functions reflect norms of productivity which team leaders consider reasonable, counterposed to concerns of commercial viability. Negotiations concern trade-offs between features that cost a lot to develop and those that cost less, but which might add more to the price. In all four cases there does not appear to have been any great conflict between the concerns of team leaders to establish targets acceptable to producers and those of marketing staff to keep prices down. Prices appear to have been based on the levels of productivity that were possible when direct producers work under a regime of having responsibility themselves for meeting targets.

Second, specifications and targets were on the whole allocated to individual producers within their team, by a team leader, who knew their level of expertise, and who could decide on a level of complexity and size for an allocation accordingly. As well as ensuring that estimates were realistic for direct producers, the judgement of team leaders was also vital for matching the complexity of blocks of work to the capabilities of team members.

12.3 THE NATURE OF CONFLICT

The cases indicate that there was little explicit conflict

between software producers subject to 'professional' forms of work organisation and their managers. Levels of goal achievement for producers were on the whole high, and the majority of producers felt fulfilled by their work. There is no evidence in any of the four cases that managers were dissatisfied with such a model of work organisation, or wished to replace it with a more rule-bound form of labour control. At one level, the simple answer to one of our fundamental research questions is that software production, as archetypical professional technical work, appears to be characterised by little conflict between organisational and individual goals.

The four case studies however provide a basis for conceptualising the root of a basic lack of explicit conflict, and also for understanding the tensions that did occur.

First of all, on the evidence of all four cases, software production is characterised by a lack of explicit conflict over norms of productivity or quality implicit in project targets. Managers do not on the whole think it important that producers worked harder, or to higher quality. Nor do producers see themselves as being worked too hard, as having to meet unrealistic quality targets, or as being asked to produce work to 'commercial' standards that they regarded as unsatisifactory from a 'technical' standpoint. The explanation for this basic lack of conflict over individual performance standards would appear to lie on the one hand with the way that team leaders were involved in the setting of targets. On the other hand, it lies with the selection and socialisation of producers, which shape their goals of performing technical work of a commercial nature, and their basic acceptance of the nature of specifications.

This suggests that software producers are, in line with Larson's (1977) view of professional technical workers in general, thoroughly incorporated into the structure and goals of business organisations. For the most part, they see their interests as served through using their skills to accomplish technical tasks stemming from business concerns. This they see as leading to

advancement in terms of financial rewards, and opportunity for greater technical influence in commercial applications of the technology of which they are practitioners.

They experience no fundamental antipathy to business priorities as representatives of a higher technological rationality (Gorz 1976). Nor do they see themselves as having class interests opposed to those who own and manage business organisations, because the latter exploit them as wage labourers (Carchedi 1975). The analyses in the case studies of the way the work goals of software producers are mobilised into a commitment to project targets show how such incorporation is achieved. Software producers learn to link the expression of their 'professional' identity, someone who has technical skills that are rewarding to use, with work in the context of a business organisation and rewards of organisational advancement. They strike a 'bargain' with organisational employment. They internalise an acceptance of the nature of project targets allocated to them in exchange for autonomy, lack of close surveillance and self-expression in performance of their work.

Whilst concluding that 'professional' control of labour in software production is characterised by an absence of fundamental conflict, there is nonetheless evidence that the terms of the bargain are not always easy to maintain. Tensions of a variety of natures emerge under different conditions.

(a) Structured Techniques and 'Professional' Control over Work

We have seen that, on the evidence of our four case studies, structured techniques are used as a basis for assuring quality of work and for coordination between producers. They are perceived by software producers on the whole as follows. On the one hand, they improve the quality of their work, and are of decisive advantage from the point of view of maintaining systems once installed. They appeal to the part of producers' work identity which attaches values to being responsible for an effective

product. On the other hand, they are seen as potentially restrictive and inhibiting of flair. They clash with the part of work identity concerned with autonomy.

Most producers appear to come to a basically favourable reconciliation between these two views. This involves seeing structured techniques as something to be mastered and then used or departed from as experience and judgement dictate. Such techniques are not a source of conflict or resentment, provided producers feel they have the right to depart from them at their discretion. This appears to be the case in each of the organisations studied. On balance, structured techniques are not seen as giving rise to destruction of skills. Whilst removing discretion in some areas, they are seen as augmenting professional capabilities in others. Producers are content to experience a mixture of initiative and formalisation in their work.

The only situation in the cases where structured programming was in fact resented was one section of Infoservices, where it had not yet been introduced, and was regarded as a management imposition. Together with the views of systems analysts in the same case, as to how structured analysis techniques should be introduced, this suggests a hypothesis as to the circumstances in which the introduction of formalised techniques are least likely to produce tensions concerned with producers feeling that their control over work is threatened. If producers are given control over the selection and introduction of new techniques, they are more likely to experience them as additions to their set of professional tools. They may indeed give rise to the introduction of areas of rule-bound and even routine work into the sphere of 'professionals'. However, any 'professional' work contains elements of 'Technicality' as well as 'Indeterminacy' (Jamois and Pelloile 1972). The development of technique in any sphere of professional technical work means that practitioners are constantly having to come to terms with the fact that it is often effective to use codified approaches to solving problems based on methods that were once inspired and innovative.

(b) Tensions Due to Underutilisation and Misutilisation

On the evidence of the case studies, the most significant forms of tension experienced by software producers take the form of feelings of underutilisation - feeling there is not enough to do - and misutilisation - feeling that work is not challenging enough. Across the sample of 68 software producers, Table 12.1 shows that there are significant correlations between measures of skill utilisation ("chance to show what I can do"), time demands of work, technical demands of work, control over important technical decisions, the degree to which individuals feel they are making an important contribution and the extent to which they are contributing to business goals. These correlations suggest that when feelings of lack of utilisation occurred, they tended to be accompanied by a lack of satisfaction of important work-related goals.

Misutilisation and underutilisation can occur under a number of circumstances within software-producing organisations. Unpredictable product market demand in an organisation seeking differentiation through innovation, combined with a large size for projects (Exco), can lead to 'labour hoarding' at the end of projects. This can lead to feelings of low skill utilisation, and lack of influence and control over work-related decisions for junior staff engaged on project 'wind-down' activities.

Table 12.1: Experiences of Utilisation, Control over Work and Visibility of Contribution

Kendall Correlation Coefficients.

	'Chance to Show What I can do':
Technical Demands:	0.48 **
Time Demands:	0.45 **
Control Over Approach:	0.24
Making the Most Important Decisions:	0.53 **
Making an Important Contribution:	0.39 *
Contributing to Business Success:	0.42 **

* = 1-tailed significance of less than 0.01.

** = 1-tailed significance of less than 0.001.

Experienced staff working on customisations of existing products in an organisation seeking differentiation on the basis of quality in a stable environment can also feel misutilised, as a result of the lack of challenge presented by a familiar set of technical problems (Telexco). Although it had a commercially

stable environment, the unpredictability of work involved in analysing user requirements within Sysco appears to have maintained levels of stimulation for experienced staff in that organisation. Customising the same products for ever new circumstances apparently offered continual opportunities to use skills.

The most common context of misutilisation, found to some extent in three of the cases (Exco, Sysco and Infoservices) is however that of work allocated to junior staff. These tensions would seem to be a result of the way that work is allocated between senior and junior positions on 'professional' career ladders, associated with the mode of coping with task unpredictability by separating off 'high-level' design work and giving it to the most experienced producers.

The result is a set of contradictory experiences for junior software producers. Whilst encouraged to develop knowledge and the ability to use discretion, the work they are presented with is heavily specified, subject to procedures and of limited scope. In one case (Sysco), there is a further element of ambiguity in the expectations placed upon them. They are expected to abide by structured programming conventions which they see their superiors ignore or adhere to as they wish.

There is here evidence of a basic ambiguity in the process of 'professional' socialisation. In order to reach the levels where the more responsible requirements specification or design work was carried out, producers had to pass through the realms of 'implementation' or 'programming', which typically offered little technical challenge or influence. The fact that there does not in any of the cases appear to have been any organised questioning of this sort of arrangement, or resistance to it, suggests, however, that junior staff were able to 'defer gratification'. They anticipated the improved work interest that would come with advancement, either within their current organisation or elsewhere.

(c) Career Opportunities

There is also evidence within the cases of tensions between the preferences of producers for the sorts of careers they would like to develop and what is on offer within their employing organisation. Within data processing organisations (Sysco), an emphasis on systems analysis skills may mean that the organisation does not provide senior positions for those who wish to remain performing systems design and programming as opposed to requirements analysis. Within more 'technical' areas of software, similar cases of frustrated 'technical' ambition are possible. Most producers are apparently content to achieve a balance between gaining more purely technical experience and gaining more responsibility for a particular product, even if it is not state-of-the-art. However, a minority tend to find that a system of spending several years at a time on one product does not satisfy their ambition for techncial skill development (Telexco).

In Chapter 5 we hypothesised that a business strategy of focus and a stress on skills associated with a particular specialised product line would lead to tensions over lack of opportunities for internal advancement beyond a certain point. This does not appear to be born out by the experience of producers within Telexco or Sysco. In both cases, the explanation would appear to be that, although the application areas concerned were specialised, they demanded general skills as well as firm specific ones. Consequently, producers could look to advancement outside the organisation as well as inside.

(d) Reactions to Vertical Information Systems

Two of the cases suggest a degree of incompatibility between a system of labour control based on allocations and targets set by team leaders and personalised, 'high trust' monitoring, on the one hand, and direct collection of monitoring information by middle managers on the other. Although computerised vertical

information systems providing information on progress direct to middle managers were tried in Exco and Infoservices, their use had been abandonned because middle managers found reports from team leaders based on information they had gathereed informally more useful.

(e) Tensions Due to Lack of Organisational Influence

A final category of tension appears where a software-producing organisation is unable to provide the 'professional' producer employed within it with a well-defined sphere of activity. Examples of this are when a corporate data processing division finds itself in competition with data processing departments set up within its 'client' divisions (Infoservices), or when a software producing organisation that had previously worked in isolation undertakes collaborative projects, in order to meet demand for new types of integrated systems (Telexco). In both cases, software producers can feel they lack influence over their work, because their organisation as a whole does. These tensions of inter-organisational origin may in fact be far more serious to the senior staff involved from the point of view of restricting their influence over work than any mechanisms of control or coordination used within their employing organisation.

Overview of the Nature of Conflict

We have seen that there is an absence of fundamental conflict between software producers and their managers, but tensions within a 'bargain' where autonomy and prospects for advancement are expected in return for a commitment to project targets. Tensions can occur over the degree to which use of formalised techniques is compulsory as opposed to subject to discretion; over the distribution of responsibilities between senior and junior practitioners; and over the balance of 'product' interest as opposed to 'technical' interest in career paths on offer. They can also arise when the employing organisation as a whole is

unable to maintain control over work in its sphere of operation.

We now examine how these tensions occurring at the level of the enterprise combine with wider processes, such as developments in product markets, to shape developments in divisions of labour and career structures in software production.

12.4 THE SHAPING OF SOFTWARE WORKFORCES

This section is concerned with drawing conclusions as to how organisational processes shape divisions of labour and career structures at the level of the enterprise, in the context of different business strategies and task environments. This forms the basis for a discussion of the development of software occupations as an example of the evolution of an area of professional technical work. We begin however, by summarising the evidence from the cases as to how software occupations had been shaped up to the time of fieldwork, and by examining what 'professionalism' meant for these workers.

The Nature of Software Work Organisation and Occupations

All four cases indicate that the forces shaping divisions of labour and the nature of career structures in software production were not related to conflict between managers and producers over levels of performance. Managers were satisfied with a 'high trust' form of work organisation. They were content with the results obtained by mobilising 'professional' commitment and skills, established through the promotion of staff up job ladders, combined with the use of structured techniques as a mechanism for quality assurance and coordination. Managers in these software-producing organisations do not appear in general to be interested in changing the basic mode of labour control in the direction of behaviour formalisation. Nor have they

introduced 'technician' grade of producers to do the more routine lower-level work currently done by junior 'professionals'. There appears to have been no move toward proletarianisation of software producers in these UK firms, nor much signs of the sort of management rationality described by Kraft and Greenbaum in US firms in the 70s.

The intention to continue systems of labour control based on responsibility and discretion of individuals in applying their skills was underlined by the concerns managers expressed with finding or developing 'the right people', staff with suitable skills and orientations. In all four cases, the future success of the organisation was linked to the 'trustable' skills it could muster, at least as much as to systems that might be devised to control and coordinate them.

This basic approach to labour control appears to hold regardless of different emphases in business environments and task environments. At one level, this is attributable to the basic complexity of software work, and the fact that means of routinising it satisfactorily have not been found. At another level, the continued satisfaction of managers with a 'high trust' or 'professional' form of labour control can be seen as stemming from the way that software producers can be selected and socialised into 'responsible' workers. There is no real need to establish 'direct control' over software producers or to 'proletarianise' their work, because their skills and orientations are not in general subversive to management goals. Whalley (1986) comes to a very similar conclusion in a study of work organisation in more traditional forms of engineering.

Our conclusions from the cases in terms of the current nature of software occupations and the forces that have shaped it are then different from those of Kraft (1977) and Greenbaum (1979). We also differ from the position of Friedman (1986), who sees management preferences for 'responsible autonomy' during the 80s as a reprieve from a move toward 'direct control' during the 70s. This reprieve has been brought about, according to Friedman,

because of more uncertain market conditions and a need for greater flexibility. What our analysis questions is the reality of past moves towards 'proletarianisation'.

The picture emerging from the cases of software occupations in the UK is then as follows. At the level of enterprises, there are unsegmented career paths by which inexperienced recruits can reach the most responsible areas of work. There is no differentiation between the educational attainment of producers found at different levels in the division of labour. All producers regardless of their level of education are generally required to start by gaining experience at the least responsible level of programming work.

There is however differentiation between the sorts of worker employed in different firms, operating in different types of software production. The basic distinction is between workforces engaged in producing data processing applications and those engaged in more 'technical' areas, such as telecommunications or systems software. The former tend to have less formal computing education, and lower-level educational qualifications generally, and are engaged on work where complexities are less to do with the production of software from specifications, more with the production of functional specifications to automate intricate business procedures.

In data processing, programming responsibilities may be allocated to junior staff as a preparation for taking on more responsible systems analysis work. In some organisations, such as Infoservices, seperate 'programming' career paths may exist, however, as a response to a perception that structured techniques are making programming tasks more complex. In neither case does 'programming' appear to be a 'technician-level' job with limited career prospects.

In some respects, the data processing occupations can be seen as a less skilled segment of the software workforce than those concerned with more 'technical' application areas. However,

according to Pettigrew (1973), the employment of less educated people as applications programmers in the early 60s stemmed from a shortage of maths graduates who could look after the systems software. In a rapidly expanding workforce, more skilled practitioners were not displaced, merely freed up to do tasks that utilised their skills more fully. The evidence from both Infoservices and Sysco is that, although entrants to data processing continue on average to have lower levels of educational attainment in comparison to producers of telecommunications software, they nonetheless can expect with time to take on high levels of responsibilty and autonomy in the sphere of requirements analysis and relating to users, as systems analysts.

To summarise the nature of 'professionalism' in software production, we have described a predominantly 'high-trust' form of work organisation, where producers exercise discretion, but at the same time observe rules and standards. We have seen that membership of occupational or professional institutions is of little relevance to producers, and that their autonomy in the performance of their work has little to do with any institutionalised occupational control over the conduct of work.

We have also seen that managers have introduced formalised techniques, such as structured design and structured programming, not as a way of 'proletarianising' producers, but as means of coordinating work and a means of assuring quality of work. Such techniques function as a form of 'standardisation of skills' to aid control and coordination of work on a 'high trust' basis.

These techniques do not stem from professional bodies or institutions outside the work place, as in Mintzberg's (1979) example of the use of standardisation of skills to control and co-ordinate the work of doctors within hospitals. In software production, the lack of occupational control means that the form standardisation of skills takes is shaped by management decisions to introduce certain techniques or standards. Such decisions are however in general made in the context of an awareness of what

other firms are doing, and what forms of training are available. There are industry-level processes shaping formalised techniques being introduced, but, on the evidence of the cases, professional institutions are not among them.

Developments in Work Organisation and Occupations

The cases indicate the following ways in which 'trusted' software workforces are being shaped at the level of the enterprise, in particular by the introduction of new techniques and forms of work organisation to cope with changing business conditions.

(a) Developments in Telecommunications Software Occupations

The cases suggest that organisations concerned with differentiation through innovation are coping with large and technically unpredictable software projects through moving toward an increasing separation of high-level 'design' from 'implementation', using formalised design techniques, within a basic framework of action planning and professional control of individual performance. Telexco provides an example of an organisation that is establishing a separate specialist design team, using structured design techniques to deal with an innovative new development by identifying technical problems at as early a stage as possible.

We have seen how the experiences of staff in both Telexco and Exco indicate the tensions associated with the use of rigorous structured design methods and a specialist design team. We have also seen that staff at all levels in both organisations were by-and-large keen to learn new structured techniques, because of the way they could be used to improve the quality of their output. Their introduction in itself does not appear to be a site of friction. However, there was removal of decision-making from implementors, which undermined the basis on which they are motivated. One conclusion is that the sorts of tensions

experienced by junior producers are likely to continue. In neither of the two cases was there evidence of any management intentions of moving away from a single career structure. Managers decided to find ways of compensating junior implementors for spells of tedious work, rather than alter the basis of control for this type of work.

Two other resolutions are however conceivable. The first is suggested by one Telexco 'designer', who felt that rigorous structured design should be combined with the automation of coding. This implies a scenario for the future where 'technical' software organisations recruit computer science graduates already skilled in design, who would work from the beginning on design, without having to serve an apprenticeship on 'low-level' work such as coding. The second possible development is the recruitment of 'technician-level' producers, who would be less qualified than computer science graduates, have lower expectations as to the level of responsibility allocated to them, and would be more amenable to control on the basis of following procedures.

The emergence of 'software technicians' in this type of software production would appear to depend on a number of sets of circumstances. First, it would require coding to become routine enough for it not to require the skills of a computer-science graduate, but too unpredictable to be automated satisfactorily. It would also depend on the relative balance of design and coding work, and whether the organisation could justify specialist 'coding' positions.

Finally, it would depend on the availability of suitable 'technician' level staff on the labour market. This would in turn depend on decisions being made by providers of vocational education and training as to the sorts of courses to offer in the field of computing. An analysis of the factors affecting decisions in this realm lies beyond the scope of this study. However, it is as potentially important for shaping the nature of software occupations as developments at the level of the

enterprise. The nature of educational provision shapes the options for managers when making decisions about what sort of workers to employ, and with what sort of career structures.

(b) Developments in Data Processing Organisations and Occupations

The cases provide evidence for two possible ways in which concerns and processes at the level of the enterprise may shape the development of data processing occupations.

First, there is evidence of concerns that programmers might in the future have to be trained more strictly in structured programming, either because it is proving more difficult to recruit programmers with the levels of skill and experience required (Sysco), or because the need to maintain systems requires that they are produced according to more definite standards (Infoservices). There is here a suggestion that programming work might become more subject to rigid procedures as a substitute for skill and experience. Should programmers recruited to be trained in this way not prove amenable to gaining the sorts of skills required by systems designers and industry consultants, there might then have to be introduced an alternative port of entry for these more responsible jobs, resulting in the effective creation of a separate job structure for programming 'technicians'. There is then a possibility of the creation of a less skilled and educated work force segment, but not as a consequence of management desires to 'proletarianise' recalcitrant 'professionals.' The motivation would be a need to utilise professional skills more fully.

Other intentions, identified within Infoservices, however, suggest a very different scenario for the evolution of data processing occupations.

The context of such intentions is an environment where product innovation in the form of the widespread availability of packaged software for business applications is causing dramatic change in

the levels of quality of service expected for a given price by customers. This leads a corporate data processing department to follow a strategy of innovative differentiation in the sorts of services it is supplying. The customisation of bought-in application packages in pursuit of such strategic goals makes programming tasks become much less significant, so that they can come within the sphere of analysts. The use of packages amounts to a simplification of the purely technical programming aspect of the task environment, freeing up resources to deal with analysis of user requirements and systems specification.

This picture does not involve any shift in the nature of labour control for systems analysts. Rather, it involves the elimination of much routine programming done by specialist programmers. It also implies the creation of new 'staff' functions for software producers, concerned with gathering information on packages available and their evaluation. According to one senior Infoservices manager, it would involve up-grading of the skills and experience of the workforce, although with a decrease in numbers. He implied the establishment over time of an all-graduate workforce, controlled through professional responsibility.

The introduction of '4th generation languages' and 'parameter driven packages,' that has gradually taken place through the 80s, is widely associated with similar developments in terms of diminishing the importance of the data processing 'COBOL programmer' (Perring 1983). Such developments illustrate the importance of changes in the nature of software products for shaping divisions of labour, as opposed to management initiatives in controlling the way that existing production processes are organised.

Intentions of introducing structured analysis within Infoservices as a way of improving the quality of system specifications appear quite compatible with the persistence of 'high trust' models of work organisation for 'analyst programmers'. The introduction of structured analysis was seen by both managers and some analysts

as having potential for improving communications with users and the accuracy of specifications. This was not linked to any ideas of shifting the basis of labour control for analysts away from 'professional' responsibility. Further, evidence that Infoservices analysts were reluctant to accept structured techniques that they did not have control over is also an indication that any change in the basis of labour control would have been opposed. It is likely that structured analysis would have been introduced as a 'professional tool', to be used at the discretion of analysts.

Professional Technical Work, Professionalism and High Commitment Management

We have seen that the organisation of software production as an example of professional technical work is shaped by processes at the level of the firm. There is a consensus between managers and producers that work should be controlled on a 'high trust' basis, and should involve the exercise of complex skills with discretion. Within this, managers may introduce a variety of mechanisms for the control and coordination of work to deal with various forms of unpredictability in task environments. Amongst these are the introduction of formalised techniques for producing software, the centralisation of software design decisions with a few experienced producers, and the use of bought-in packages which diminish the importance of actual programming in data processing.

None of these developments pose an imminent threat to 'high trust' principles of labour control, although the centralisation of design decisions, in particular, could provide the basis for the introduction of less-skilled 'technicians' in the more technical branches of software production. Whether or not this occurs would appear to depend on decisions made by training providers as to the sort of software producer they should be supplying.

Whilst the prospects for a continuation of a model of labour control based on skills and responsibility as well as a degree of formalisation appear good, prospects for any degree of occupational control over work appear poor. We have argued that software producers appear on the whole to have their work-related goals met well by the sort of work available to them, and by high demand for their services. Occupational or professional institutions have no role for them. It is also worth noting the great variety of software jobs even within the four organisations studied. The apparently increasing importance of 'staff' jobs, engaged on planning or defining development techniqies, for example, suggest that even within one organisation, software workforces may become more fragmented and specialised, with the emergence of many parallel career paths. This could weigh against the emergence of a collective professional identity.

If software producers have few of the self-regulating attributes of the 'true' professions, such as law or medicine, they can still be distinguished as 'professional technical', as distinct from other occupations where work is controlled on a high trust basis. During the 80s, many writers have noted the increasing importance of 'high trust' forms of organising in many areas of blue collar work (Kern and Schumann 1987, Walton 1985, Lawler 1986). Chemical process operators entrusted with responsibility in using discretion can however be distinguished from professional technical workers: first because there is not a common career path for them into management; and second because software producers have a much clearer identity with a body of general or transferable intellectual expertise and knowledge.

The tensions identified in the functioning of software work organisation do however illustrate some general issues in the functioning of 'high trust' organisation that are of relevance to initiatives to establish it in new sectors. In particular, three areas stand out. First, there are issues of 'high commitment' teams developing their own standards coming into conflict with organisation-wide practices necessary for coordination. Second, there are tensions in training and socialising new employees, in

balancing their need for stimulus with their lack of experience and their unproven commitment. Finally, there are issues for managers in ensuring that the organisation as a whole controls its sphere of operation, as a basic requirement for workers within it to control their own sphere of work.

APPENDIX 1

Checklist of Data to be obtained from Managers and Project Managers/Team Leaders with Selected Business Unit

1. Application areas, software services and products concerned. Perceptions of terms of competition, type of competitive advantage sought, overall performance goals.

2. Organisational structure of whole organisation and business unit, including interface between software production function and other functions e.g. marketing. Spans of control within production section.

3. Performance goals for slected business unit, and for software production department within it.

4. Job titles in production section, and numbers in each
   - areas of expertise and knowledge associated with each job title
   - perceived shortages.

5. Typical make-up of production teams, in terms of the specialisations involved, and the functional division of labour between them.

6. Use of software engineering techniques with production teams
   - 'top-down' design
   - structured walk-throughs
   - 'chief programmer' teams.

7. Frequency of contact between middle managers and production leaders, and over what sort of issues (i.e. data on the use of the hierarchy of

authority as a control and co-ordination mechanism).

8. Use of rules and procedures within the business unit as a whole, and within production teams.

9. In what terms are targets set for project groups?
   - how precise are time targets?
   - how detailed and formalised are technical specifications?

   How are targets set for individuals within project groups?

10. Are slack resources employed in any form? E.g:
    - programmers working on low-priority assignments
    - programmers that can be obtained from other departments of the organisation, or from outside on short contracts?

    If so, what sort of programmers are these generally? (high skill? low skill?)

11. Forms of environmental management attempted by the organisation.

12. Methods of monitoring the progress of projects?
    - at what points and using what measures is the progress of a project monitored?
    - in what terms are individual contributions and performance monitored? i.e. what information goes on time-sheets?

13. What information collecting and recording devices are used by higher management to deal with problems of slippage and replanning, and the redeployment of personnel to meet orders of varying priority? How formalised are such devices?

14. Use of lateral relations e.g. between members of interdependent production teams, within the software production organisation.

15. What are the major problems experienced by managers at various levels in carrying out their job?

16. Promotion assessments:
    Who does them?
    What are the criteria?
    Is administrational management involved as well as technical management?
    Are formalised performance evaluations used?

17. Is pay linked to job-title only, or can good performance be rewarded with a raise without promotion?

18. Cross sectional survey of the deployment of software producers in project teams at one point in time.

19. Investigate possibility of consulting records for longitudinal analysis of deployment of a few individuals over, say, 3 months - to gain measure of the flexibility required of the workforce.

Appendix 2:

The Interview Schedule

used for gathering data from software producers

INTERVIEW SCHEDULE FOR SOFTWARE PERSONNEL

Dick Holti DSES May 1982

I. Job title
Length of time in current position

Length of time at Hoskyns

Age: under 20	20-24	25-29	30-34	35-39	40-45	Over 45	
1	2	3	4	5	6	7	8

2. Qualifications and Education.

Qualification	Subjects	Year

3. Employment History

i) Where?

Brief Description of job content

Training received

Areas of expertise acquired

ii) Where?

Brief description of Job content

Training received

Areas of expertise acquired

4.Formal Training Received at Hoskyns:

Programming Techniques	I
Manufacturers Systems software course	2
MAS Product Kowledge Course	3
Programming Management Techniques	4
OTHER	5
Project Management Course	6
Sales Course	7
None	8

5.Membership of outside organisations:

Trade union	I
	2
	3
Professional Organisation BCS	4
BPEC	5
Other	6
None	7

6. Current Job

i) What areas have you worked in predominantly since you began in your present position, or over the last year if you've been in the same position for longer than that?

Systems Software	I
Applications Programming	2
User Oriented Systems Analysis/Consultancy	3
Computer System Design	4
Installation Work	5
Sales Support	6
Training	7
Project Management	8

Have you worked in any other of these areas previously?

Type of work	Period over which performed

ii) Utilisation of Skills

a) How often does it happen in your present job that your project offers you a real chance to show what you can do? % of time

b) How would you rate the time as opposed to the technical demands of your present job? I 2 3 4 5 6

c) How would you rate the technical demands of your job? I 2 3 4 5 6

d) How often in your present job is your assignment below your technical capacity? % of time

iii) Importance to business

How often in your present postion does it happen that:

a) You feel that you've made an important contribution to the final system? % of time

b) Your work has a direct impact on the business success of HSD? "

c) You have the opportunity to discuss your work directly with management higher than your immediate superior? "

d) The work you are doing is of special interest to higher management? "

e) You participate in decisions that affect the future business success of HSD? "

iv) Individual Responsibility

How often does it happen in your present position that:

a) you have the freedom to adopt your own approach to the job? "

b) you are making all the impotant decisions on matters within your area of technical responsibilty? "

v) Visibility of individual contributions: Extent of direct control

How often does it happen in your present position that:

a) the specifications of your part of the job have already been agreed by others? "

b) every step you take must be agreed upon by those with whom you work? "

c) No-one in higher management seems to know what you are doing "

d) your part of the project seems trivial or routine? "

e) you are given detailed instructions as to how to actually proceed with the job?

7. Technical vs Commercial orientation

i) Do you see taking part in a siftware project as taking part in:

A creative activity	I
A production process	2
Other	3
It depends	4
don't Know	5

Why do you say this?

ii) Do you think that the recognition given to applications programming skills in this organisation is:

Much too great	Rather too great	About right	Rather too small	Much too small	Don't know
1	2 3	4	5 6	7	8

What about the recognition for applications programmers? 1 2 3 4 5 6

iii) DO you think that the recognition given to systems software skills in this organisation is:(As above)

1 2 3 4 5 6 7

iv) Do you think that the recognition given to user industrial knowledge and systems analysis in this organisation is:(As above)

1 2 3 4 5 6 7

What about the recognition for industry-oriented consultants? for technical consultants?

8. Personal goal satisfaction

i) In general,how well would you say that the work you are doing is helping to achieve achieve your most important personal goals?

Not at all	Only slightly	Satisfactorily	Fairly well	Perfectly	Don't know
1	2 3	4	5 6	7	8

ii)How well would you say your current job measures up to what you want out of a job?

1 2 3 4 5 6 7

9. Opportunities for promotion

i) If you stay within this organisation,how long would you expect to have to wait for your next promotion?

Less than 6 Months	6-I2 Months	I-2 Years	2-3 Years	Over 3 Years	Don't Know
I	2	3	4	5	6

ii) Do you think that this is?

Much too soon	rather too soon	About right	Rather too long	Much too long	Don't Know
I	2 3	4	5 6	7	8

iii) Would you expect this to be to a position involving:

greater opportunity to develop and use technical programming skills I

greater opportunity to develop and use knowledge of a user industry and systems 2

programming management skills 3

project management skills 4

5

Don't Know 6

iv) Do you think that this expected step up is :

Much too Small		rather too small	About right	Rather too big		Much too big	Don't know
I	2	3	4	5	6	7	8

v) What position do you expect to be promoted to?

Io. Please indicate on a scale of I to 9 how important or unimportant the following qualities are for promotion in this organisation:

i) Ability to solve problems independently of supervision

	I	2	3	4	5	6	7	8	9	Io
Unimportant									Very Important	D.K

ii) Performing according to procedures laid down by management

	I	2	3	4	5	6	7	8	9	I0

iii) Good technical knowledge

	I	2	3	4	5	6	7	8	9	I0

iv) Good understanding of user problems and requirements.

	I	2	3	4	5	6	7	8	9	I0

II. In what sort of job do you see yourself in 5 years time?

Primarily involving:	Technical computing skills	I
	Apllications programming	2
	User systems analysis/consultancy	3
	software management	4
	software sales	5
	Project Management Skills	6
		7
	Don't Know	8

I2. How do you think that your salary here compares with that for similar positions in other organisations in London?

Much Lower	I
A little lower	2
about the same	3
a little higher	4
much higher	5
Don't know	6
	7

Would you say the same taking perks into account

I3. Can you say on a scale of I to 9 how strongly you disagree or agree with the following statements?

Disagree completely I 2 3 4 5 6 7 8 9 Agree Completely

Indifferent

(a) In my type of work,it is important to take any opportunity to increase one's earnings,even if this means chopping and changing companies.

I 2 3 4 5 6 7 8 9 IO

D.K.

(b) In my type of work,it is important to develop one's skills and experience, even if this means chopping and changing companies.

I 2 3 4 5 6 7 8 9 IO

D.K.

(c) In my type of work,it is important to develop one's carrier within one company.

I 2 3 4 5 6 7 8 9 IO

D.K.

I4.Can you say on a scale of I to 9 how strongly you agree or disagree with the following statements

(a) It is important to me that an assignment gives me a real chance to show what I can do. I 2 3 4 5 6 7 8 9 IO

(b) You don't mind if your assignment is below your technical capacity

I 2 3 4 5 6 7 8 9 IO

(c) It is important to you that your work has a direct impact on the business success of success of the company. I 2 3 4 5 6 7 8 9 IO

(d) You don't feel that it's important for you to participate in decisions that affect the future business success of HSD. I 2 3 4 5 6 7 8 9 IO

(e) It is important to you to have the freedom to adopt your own approach to the job. I 2 3 4 5 6 7 8 9 IO

(f) You don't mind if your part of a project seems trivial or routine.

I 2 3 4 5 6 7 8 9 IO

(g) You prefer to have detailed instructions as to how to proceed with a job.

I 2 3 4 5 6 7 8 9 IO

I5.a)How would you view the introduction of "Advanced Software Engineering Techniques" in this organisation?

Very opposed	I
Mildly Opposed	2
Indifferent	3
Faily keen to learn them	4
Very keen to learn them	5
Don't Know	6
	7
	8

b)Why is this?

I6.Are there any further comments you feel are relevant?

Appendix 3:

The Diary Sheets

used for gathering data from software producers

DIARY SHEET FOR WEEK ENDING

For each new assignment that you have been given during the last week, please answer the following questions by ticking the appropriate box, except for questions 3,4 & 7, for which you can write a one-figure or one-word answer in the box.

		Assignment No. 1	2	3	4
Is the overall project?	Package customisation	☐	☐	☐	☐
	Development work on a new product	☐	☐	☐	☐
	Bespoke development for a client	☐	☐	☐	☐
	Supporting a client after sale	☐	☐	☐	☐
	Sales support	☐	☐	☐	☐
	Other: please specify:-	☐	☐	☐	☐
Is the application area?	Production control	☐	☐	☐	☐
	Cost control	☐	☐	☐	☐
	Inventory control	☐	☐	☐	☐
	Financial control	☐	☐	☐	☐
	Purchasing control	☐	☐	☐	☐
	Other: please specify:-	☐	☐	☐	☐
Please enter the number of people from HSD in the project team for each assignment, including yourself. If you are the only one working on the project, please put '1'.		☐	☐	☐	☐
What hardware will the system run on?		☐	☐	☐	☐
Within the project, is this assignment- (Tick more than one if appropriate)	Preliminary systems analysis and cost estimation	☐	☐	☐	☐
	Project management	☐	☐	☐	☐
	Functional system design	☐	☐	☐	☐
	Managing a programming team	☐	☐	☐	☐
	Computer system design	☐	☐	☐	☐
	Applications programming	☐	☐	☐	☐
	Testing/Debugging	☐	☐	☐	☐
	Documentation	☐	☐	☐	☐
	Installation	☐	☐	☐	☐
	Training Hoskyns personnel	☐	☐	☐	☐
	Training user personnel	☐	☐	☐	☐
	Other:please specify:-	☐	☐	☐	☐
Have the methods and procedure you use in tackling this assignment come mainly from?	Your own knowledge and experience	☐	☐	☐	☐
	Your immediate supervisor	☐	☐	☐	☐
	Another colleague	☐	☐	☐	☐
	Consulting a manual (please write which manual)	☐	☐	☐	☐
	Elsewhere: please specify:-	☐	☐	☐	☐
How many person days has the assignment taken (if completed) or will it have taken by the time you have completed it?		☐	☐	☐	☐

DIARY SHEET FOR SAMPLE DAY

If you have spent any time today discussing your current work with your immediate supervisor and/or other colleagues, please can you fill in the corresponding box with the length of the exchange, in minutes. If you have had more than one exchange with the same person, or a number of exchanges with different people, you will need to fill in more than one box.

	Exchanges with immediate supervisor (minutes)						Exchanges with colleague 1 (minutes)			Exchanges with colleague 2 (minutes)			Exchanges with colleague 3 (minutes)		
Nature of Exchanges	1st one	2nd one	3rd one	4th one	5th one	6th one	1st one	2nd one	3rd one	1st one	2nd ome	3rd one	1st one	2nd one	3rd one
Informally reporting on progress															
Submitting formal evidence of progress															
Getting help with a problem in <u>your</u> work															
Giving help with a problem in the <u>other</u> person's work															
Discussing a problem arising from a connection between your two assignments															
Other: please give nature of exchange as well as length															

Appendix 4:

Questions used for Initial Analysis of Case Study Data

1. What software products and/or services are offered by the business unit? What is its size and relationship to any parent organisation? In what terms are its overall performance goals set? What is the market or environment facing the selected business unit, and what competitive strategy has been adopted to deal with its demands and uncertainties? If the business unit is part of a larger organisation, how is its environment and competitive strategy shaped by the larger organisation?

2. Within the selected business unit, how is the overall task organised in terms of functional responsibilities or sub-tasks of the main departments? In what terms are performance criteria set for the department or departments where software production takes place? What type of interdependencies are there between the sub-tasks of the production department and other departments e.g. marketing, hardware development? How are these organised and co-ordinated?

3. Within the software production department, what are the academic backgrounds and job histories of staff? What horizontal and vertical divisions of labour are present? What job structures appear to be present?

4. Within the software production department, what forms of action planning are used? i.e. how are sub-tasks defined, allocated, and in what terms are targets set for their accomplishment, at the levels of:

   - teams or work groups?
   - individuals?

5. What software production techniques and forms of work organisation are employed? How are various aspects of performance at the level of the team and the individual monitored and controlled? How is the performance at the level of projects and the department as a whole monitored? What are the main sources of deviation from plan, and what adjustment mechanisms are available to management at various levels?

6. To what extent do the work goals and orientations of software producers correspond to those typical of professional technical workers as identified by Ritti? What aspects of current work, the employment relationship and career aspirations are particularly important for goal achievement and for motivation to meet project targets? How are work goals and orientations reproduced?

7. To what extent do experiences of current work, the employment relationship and promotion opportunities meet direct producers' goals and aspirations? How is the use of structured techniques and other software engineering methodologies viewed?

8. What are the respects in which the orientations and work goals of software producers contribute to the functioning of the organisation to meet performance requirements, and with the mechanisms of management control and coordination used? What conflicts and tensions exist between the preferences of direct producers and the requirements of mechanisms of management control and coordination?

9. What are the main problems and tensions identified by team leaders and managers at various levels in running their organisation? Are there any obvious tensions experienced between them and direct producers? Do these tensions stem from conflict over individual performance

standards, or from control and coordination mechanisms which pull direct producers in contradictory directions?

10. What are the assumptions and strategies of various levels of management eg. senior, departmental, team leaders, with regard to the crucial problems that need to be tackled and the way that organisational structures, the organisation of work and systems of control should be changed? How do they see the role of professional autonomy on the one hand and structured techniques on the other?

11. What are the implications of these various views for the actual development of work organisation, its control and the types of autonomy currently existing for the various types of direct producer? Given these preferences, what are likely outcomes?

Bibliography

Baker, F. F. (1972) Chief programmer team management of production programming. **IBM systems journal**, 11, pp. 56-73.

Barber, B. (1963) Some problems in the sociology of the professions. **Daedalus**, 92, No. 4, pp. 669 - 688.

Becker, H. S. (1962) The nature of a profession. In National Society for the Study of Education, **Education for the professions**. Chicago: University of Chicago Press.

Becker, H.S. (1970) **Sociological Work**: Method and Substance. London: Penguin.

Bell, D. (1968) The Measurement of Knowledge and Technology in E.B. Sheldon and W.E. Moore Eds. **Indicators of Social Change**. New York: Russell Sage

Bell, D. (1973) **The coming of post-industrial society**. New York: Basic Books.

Blau, P. (1969) The hierarchy of authority in organisations. **American Journal Of Sociology** pp. 453-467.

Bohm, B. (1977) Software engineering as it is, in H.Freeman and P.M. Lewis Eds **Software Engineering** New York: Academic Press.

Bohm & Jacopini (1966) A logical basis for program structure **ransactions of The IEEE**

Braverman, H. (1974) **Labor and monopoly capital: the degradation of work in the twentieth century**. New York: Monthly Review Press.

Brooks, F. P., Jr. (1975) **The mythical man-month, essays on software engineering**. Reading, Massachusetts: Addison-Wesley.

Burns, T. and Stalker, G. M. (1961) **The management of innovation**. London: Tavistock Publications.

Carchedi, G. (1975) On the economic identification of the new middle classes **Economy and Society** Vol 4 No. 1 pp 1-76.

Chandler, A. D., Jr. (1962) **Strategy and structure: history of the industrial enterprise**. Cambridge: MIT Press.

Channon, D.F. (1973) **Strategy and structure in British Enterprise**. Boston: Harvard Graduate School of Business Adminstration.

Child, J. (1972) Organizational structure, environment, and performance: the role of strategic choice. **Sociology**,

pp. 1 - 22.

Child, J. and Fulk, J. (1983) Maintenance of occupational control: the case of professions. **Work and occupations**, 9, No. 2, pp. 155 - 192.

Computer Survey June 1963 London

Cooley, M. (1987) **Architect or bee? The human/technology relationship, 2nd edition.** Boston: South End Press.

Cutler, T. (1978) The romance of labour. **Economy and society**, 7, No. 1, pp. 74 - 79.

Cyert, R. M. and March, J. G. (1963) **A behavioral theory of the firm.** New Jersey: Prentice-Hall.

Daniels, A. K. (1975) Professionalism in formal organizations. In J. B. McFinlay (ed.), **Processing people: cases in organizational behaviour.** London: Holt, Rinehart & Winston.

Dawson, S and Wedderburn, D. (1981) Introductory Essay to J. Woodward **Industrial Behaviour: Theory and Practice.** Oxford: Oxford University Press.

Deutsch and Shea, Inc. (1963) A profile of the programmer. **Industrial relations news.** New York: Deutsch, Shea and Evans, Inc, 49 E. 53rd Street, NYC.

Doeringer, P. B. and Piore, M. J. (1971) **Internal labour markets and manpower analysis.** Lexington, Massachusetts: Heath.

Dolotta, T. A., Bernstein, M. I., Dickson, R. S., Jr., France, N. A., Rosenblatt, B. A., Smith, D. P., and Steel, T. B., Jr. (1976) **Data processing in 1980-85: a study of potential limitations to progress.** New York: John Wiley and Sons.

Drucker, P. (1970) **Technology, Management and Society.** New York: Harper Row.

Edwards, R. (1978) Social control of production at the point of production. **The insurgent sociologist** Vol 8, No. 2 & 3.

Elger, T. (1982) Braverman, capital accumulation and deskilling. in S.Wood (Ed.) **The degradation of work?** London Hutchinson.

Friedman, A. (1977) **Industry and Labour: class struggle at work and monopoly capital.** London: Macmillan.

Friedman, A. & Cornford, D. (1982) **DP Survey Report 1982** University of Bristol.

Friedman, A (1986) Specialist labour in Japan: computer skilled staff and the subcontracting system. **British Journal of Industrial Relations** Vol 25, No.3, pp. 353 - 370

Friedson, E. (Ed.) (1970) **The professions and their prospects** California: Sage Publications.

Financial Times Survey on Computer Services 17.9.80
Financial Times Survey on Computers 2.3.81, 18.1.82
Financial Times Survey on Computer and Software 5.11.82
Financial Times Computers in Business Special. 11.4.83

Galbraith, J.K. (1967) **The new industrial state.** Boston: Houghton Mifflin.

Galbraith, J (1977) **Organisation design.** Reading, Mass: Addison-Wesley.

Gorz, A. (1967) **Strategy for labour.** Boston: Beacon Press.

Gouldner, A. (1957) Cosmopolitans and locals: towards an anlaysis of latent social roles. **Administrative science quarterly**, 2, pp. 281-306.

Greenbaum, J. (1979) **In the name of efficiency.** Philedelphia: Temple University Press.

Hall, R. (1968) Professionalisation and Bureaucratisation. **American Sociological Review**, 33, pp. 92-104.

Hambrick, D.C. (1983) An empirical typology of mature industrial-product environments. **Academy of Management Journal**, 18, pp. 213-230

Hickson, D.J., Pugh, D.S. and Pheysey, D.C. (1969) Operations technology and organisation structure: an empirical reappraisal. **Administrative Sciences Quarterly**, pp. 378-379.

Hughes, E.C. (1958) **Men and their work.** New York: Free Press.

Hrebiniak, L.G. & Joyce, W.F. (1985) Organisational adaptation: strategic choice and environmental determinism. **Administrative Sciences Quarterly**, 30, pp. 336-349.

Jackson, J (Ed.) (1970) **Professions and professionalism.** Cambridge: Cambridge University Press.

Jamois, H & Pelloile, B. (1970) Changes in the French university-hospital system. In J.A. Jackson (Ed.) **Professions and professionalism.** Cambridge: CUP.

Johnson, T. J. (1972) **Professions and Power.** London: Macmillan.

Johnson, T. J. (1980) Work and power. In J. Esland and G. Salaman, (Eds.) **The politics of work and occupations.** Milton Keynes: Open University Press.

Kern, H. and Schumann, M. (1987) Limits of the division of labour. New production and employment concepts in West German Industry. **Economic and industrial democracy**, vol 8, pp. 151-170.

Kraft, P. (1977) **Programmers and managers: the routinisation of computer programming in the united States.** New York, Springer Verlag.

Kraft, P (1979) The industrialisatio of computer programming: from programing to software production. In Zimbalist, A (Ed.) **Case studies in the labour process.** New York: Monthly Review Press.

Larson, M.S. (1977) **The rise of professionalism: a sociological analysis.** California: University of California Press.

Lavington, S. (1980) **Early British computers.** Oxford: Oxford University Press.

Lawler, L (1986) **High involvement management.** San Francisco: Jossey-Bates.

Lawrence P.R. & Lorsch, J. W. (1967) **Organisation and environment: managing differentiation and integration.** Homewood, Illinois: Irwin.

Littler, C. & Salaman, G. (1982) Bravermania and beyond: recent theories of the labour process. **Sociology, pp. 251 - 267.**

Loveridge, R. (1982) Manpower planning. **London: Wiley.**

Mallet, S. (1975) The New Working Class. **Nottingham: Spokesman Books.**

Miles, R.E. & Snow, C.C. (1978) Organisational strategy, structure and process. **New York: McGraw Hill.**

Miles, R.E., Snow, C.C., Meyer A.D. and Coleman, H.J. (1978) Organisation Structure, Strategy and Process, Academy of management journal, **pp. 546-62.**

Miller, D. (1988) Relating Porter's business strategies to environment and structure: analysis and performance implications. Academy of Management Journal **Vol 31, No. 2. pp. 280 - 308.**

Millerson, G. (1964) The qualifying associations: a study in professionalisation. **London: Routledge and Kegan Paul,**

Mintzberg, H. (1979) The structuring of organisations. **New Jersey: Prentice-Hall.**

Mumford, E. (1972) Job satisfaction: a study of computer specialists. **London: Longmans.**

National Economic Development Council (NEDC) (1980) Report of the Electronics Sector Working Party on the Supply of Computer Skilled Manpower. London, NEDC.

Nyborg, P., McCarter, P.M. and Ericksen, W (eds.) (1977) **Information processing in the United States, a quantitative survey.** Montvale, New Jersey: AFIPS press.

Parkin, F. (1979) **Marxism and class theory: a bourgeios critique.** New York: Columbia University Press.

Perring, I. (1983) **Information technology and job creation: software and the software industry.** FAST series No. 13. Luxembourg: Commission of the European Communities.

Perrow, C. (1970) **Organisational analysis: a sociological review.** New York: Wadsworth.

Pettigrew, A. (1973) **The politics of organisational decision-making.** London: Tavistock.

Porter, M.E. (1980) **Competitive strategy.** New York: Free Press.

Porter, M.E. (1985) **Competitive Advantage** New York: Free Press.

Poulantzas, N. (1975) **Classes in contemporary capitalism.** London: New Left Books.

Prandy, K. (1965) **Professional employees: a study of scientists and engineers.** London: Faber and Faber.

Pugh D.S., Hickson, D.J. Hinings, C.R. and Turkner, C. (1968). The context of organisational structures. **Administrative science quarterly,** 14, pp. 91 - 114.

Reeves, T.K., and Woodward, J. (1970) The study of managerial control in J. Woodward, **Industrial organisation, behaviour and control.** Oxford: Oxford University Press.

Ritti, R. (1971) **The engineer in the industrial corporation.** New York: Columbia University Press.

Saks, M. (1983) Removing the blinkers? A critique of recent contributions to the sociology of the professions. **Sociology,** pp. 1 - 21.

Salaman , G (1979) **Work organisations: resistance and control.** London: Longman.

Schneiderhan, B (1979) **Software pyschology.** Cambridge, Mass: Winthrop.

Soma, J.T. (1976) **The computer industry** New York: Macmillan.

Stewart, R (1963) **The reality of management.** London: Heinemann.

Thompson, J.D. (1967) **Organisations in action.** New York: McGraw Hill.

Walton, R. E., (1985) From control to commitment in the workplace. **Harvard business review, 63, No. 2,** pp. 76 -84.

Weber, M (1968) **Economy and society: an outline of interpretive sociology, Volume 1.** New York: Bedminster Press.

Weick, K. (1979) **The social psychology of organising.** Reading Massachussetts: Addisson-Wesley.

Weinberg, G. (1971) **The pyschology of computer programming.** New York: Van Nostrand Reinhold.

Westin, A. & Baker M.A., (1972) **Databanks in a free society, computers, record keeping and privacy.** New York: Quadrangle Books.

Whalley, P. (1986) **The social production of technical work.** London: Macmillan.

Williamson , O.E. (1975) **Markets and Hierarchies: analysis and anti-trust implications.** Boston: Free Press.

Wood, S. (Ed.) (1982) **The degradation of work?** London: Hutchinson.

Woodward, J. (1965) **Industrial Behaviour: Theory and Practice.** Oxford: Oxford University Press.

Woodward, J. (1970) **Industrial organisation, behaviour and control.** Oxford: Oxford University Press.

Wright, E.O (1978) **Class crisis and the state.** London: New Left Books.

Yourden, E. (1979) **Managing the structured techniques.** London: Prentice-Hall.