MANAGEMENT OF AN INFORMATION

SYSTEMS (IS) DEPARTMENT

by

IAN PETER CAITHNESS

September 1985

Submitted to the University of Cape Town in partial fulfillment of the requirements for the degree of Master in Industrial Administration.

I, **IAN PETER CAITHNESS,** submit this thesis for the degree of Master in Industrial Administration. I claim that this is my original work and that it has not been submitted in this or in a similar form for a degree at any University.

ACKNOWLEDGEMENTS.

The author wishes to acknowledge the assistance and encouragement afforded him by Mr W. Jervis of the Department of Mechanical Engineering.

CONTENTS.

PAGE

LIST OF FIGURES.

LIST OF TABLES.

ABSTRACT.

This thesis discusses the principles relating to the management of an Information Systems (IS) department. To be effective an IS service must support the ultimate goals of the organization directly. In this regard, user acceptance of the services offered is essential. In addition, economic justification and verification of the technical feasibility of an IS project is essential to ensuring minimum wasted effort, and senior management's commitment to the project.

IS management problems are grouped into four major categories:

i) Problems resulting from the "abstract" nature of the IS product.
ii) Problems resulting from management orientation.
iii) Procedural problems in the development of the product.
iv) Problems resulting from excessive staff turnover in the computer industry.

Following analysis of these problems, a two-fold solution is proposed. Firstly, implementation of a methodology to improve control of the IS product. This involves;

i) implementation of a quality assurance program aimed at establishing acceptable technical standards, thereby generating confidence in the product, and at informing all concerned, responsible or affected persons of the status of the product, allowing sufficient time for errors, deficiencies or deviations from requirements to be detected and corrected with minimal inconvenience,
ii) implementation of a project measurement and control procedure, covering the life-cycle of the product, providing sufficient "visibility" to facilitate adequate analysis of the risks incurred when management decisions affect progress on the project, and,

iii) implementation of configuration control procedures to allow the product configuration to be accurately defined at discrete points in time, and changes to this configuration to be controlled.

Secondly, following a brief analysis of the personality characteristics of IS personnel, a work environment is proposed which will provide challenge, responsibility, recognition and a sense of achievement, and opportunities for advancement and personal growth. Responsibilities are organized along functional lines, distinguishing between computer facilities management, data processing production operations, systems development and maintenance, and corporate support functions. To ensure that sight is not lost of the ultimate goals of the organization, provision is made for user priorities and company strategy to influence IS activities through an IS Steering Committee.

Although various aspects of IS management have been discussed in general literature, to date, the author has found no attempt to incorporate these into a single methodology. This thesis presents such a methodology for the management of an IS department.

CHAPTER 1

INTRODUCTION

The application of computerised information systems (IS) to the development and control of business, engineering and production projects has become increasingly popular in recent years. Availability of effective computing facilities, and their efficient utilisation, has greatly influenced the level of success (or failure) attained in these projects.

The provision of efficient, effective and reliable computing systems and facilities can contribute meaningfully to the attainment of the goals of an organization. However, review of literature (5,6,10,14,25,27) reveals an ever increasing dissatisfaction with the service provided. Application software is frequently ineffective, facilities being either incomplete or non-existent, while development and implementation schedules are unreliable. This thesis examines the causes of dissatisfaction and discusses a management methodology to provide a satisfactory information systems service. Emphasis is placed on understanding the principles governing the successful management of information systems, the technique or mechanics of the management operation being considered of secondary importance*.

IS management problems can be grouped into the four distinct categories discussed in chapter 2.

FOOTNOTE

* Drucker P.F. (15) points out that most work in management sciences has concentrated on tools, techniques and mechanics of management and on improving the efficiency of a part of the organization; whereas emphasis should rather be placed on the principles applying to management, or making decisions and achieving results, and on improving the performance of the whole organization.

i) Problems resulting from the nature of the software product.

Computer software is a relatively recent development, its abstract nature distinguishing it markedly from hardware. This fundamental difference between software and hardware products results in considerable misunderstanding of how standard managerial disciplines should be applied in each field.

ii) Problems resulting from management orientation.

Senior management often lack an appreciation of the "unique complexities of software development" (10) and maintenance. This influences their approach to software development and implementation projects, and their attitude to IS management.

iii) Procedural problems.

Lack of proper definition for software products, or formal planning procedures, often results in poor estimates of project duration, resource requirements and cost.

iv) Problems resulting from excessive staff turnover in the computer industry.

Excessive staff turnover in the computer industry presents a serious threat to the successful development implementation and maintenance of computerised information systems. This in turn deters management from extensive computerisation.

This thesis describes a two-fold solution to these problems. Firstly, implementation of a management methodology to improve control of the software product, and of the development, implementation and maintenance process. Secondly, a functional distribution of responsibilities within the IS department, while allowing activities to be influenced by user priorities and the stategic objectives of the company. This organization of IS activity needs to be supported by a management style which generates

high quality and productivity from the workforce, while ensuring organizational stability.

A management methodology for IS projects is discussed in chapters 3 through 6. Factors affecting the acceptability of the service offered are identified in chapter 3, and an approach to achieving this acceptance is discussed. Quality assurance, an overriding principle applying to all aspects of management, is discussed in chapter 4. The impact of poor quality on cost, development and future maintenance effort is examined, and a "standard" for QA plans for IS projects is proposed. The monitoring and control of IS projects is discussed in chapter 5. This task is hampered by the absence of standards for defining the status of a software product. A technique is required for identifying configuration elements of a system, and estimating the effort required in the development and maintenance of those elements. Such a configuration control technique for software is detailed in chapter 6.

The second part of the thesis is devoted to the organization and control of the information systems department's activities. Personnel are the "KEY" resource in any IS project. The development of a reliable, capable, motivated workforce, and the co-ordination and control of ~~it's~~ its activities in support of the goals of the organization is the prime task of the Information Systems Manager. To do this, some understanding of his subordinates' behaviour patterns, their expectations from the work environment, and their career aspirations is necessary. The personality characteristics of IS personnel, the system of values applied in assessing the reward for their efforts, and their perception of the working environment are discussed in chapter 7. Finally, the author's approach to organization and management of the IS function in a medium to large industrial company is described in chapter 8.

CHAPTER 2

THE INFORMATION SYSTEMS (IS) MANAGEMENT PROBLEM.

The impact of poor performance in the development and operation of information systems is a major concern in the systems industry (5,6,10,34,12,25,9). Frequent failure on the part of IS departments to meet development and implementation schedules, failure to satisfy user requirements, and poor systems performance result in loss of confidence in the capabilities of these departments. In some organizations users prefer the services of "outsiders", such as software consultants and computer bureaux, to those of their own IS departments. They feel they have greater control over such organizations, and greater confidence in their results.

In the author's experience, senior IS staff are often overly concerned with the status of their department in the organizations hierarchy. They are frequently diverted from providing an information service to grandiose ideas of running the firm. It is argued that with a greater influence in the management of the organization they could create an environment in which IS personnel would have better control over the user's utilisation of computer facilities. IS personnel could be given greater authority in controlling implementation projects, and where necessary could impose methods and procedures on uncooperative users. This they reason, would improve the effectiveness of the facilities provided. However, this argument fails to take account of the lack of experience IS personnel have in general, of the environment in which these facilities are to be used. The development of effective methods and procedures in any particular field requires in depth knowledge and understanding of the specific problems experienced. With their expertise generally confined to the technical development of software, IS personnel are unlikely to provide sound or complete solutions to these problems.

The author is of the opinion that greater effort should rather be expended in establishing the principles applying to the management of information systems, in applying these principles to the development of effective techniques for managing IS projects and in understanding the organization and industry served. This would enable the department to create facilities which would be of "real" benefit to the organization and in this way render a more efficient and effective service in the solution of IS problems experienced within the organization.

A greater commitment to servicing the organization's information requirements, and the demonstration of greater reliability in the provision of facilities which can be of "real" benefit to users would increase confidence in the IS department. Greater emphasis should be placed on creating a willingness and desire in the organization to utilise the services offered by the department. IS managers could then claim to be actively participating in the achievement of the goals of the organization, and would be justified in demanding recognition in the organization's hierarchy.

Before discussing the management of information systems it is necessary to examine the problems experienced in the present environment. What are the shortfalls in current techniques and what impact do these have on the systems industry and on the organizations it serves?

2.1 THE SOFTWARE "CRISIS".

Several authors have described the problems experienced in the software industry today (6,5,10,9,25,17). Some refer to the present predicament as a "software crisis" (25a,17), implying that the management of information systems projects is generally unreliable; that IS projects are seldom completed within the contraints of their original estimated cost, production schedule, or performance specification. The author has found little published quantitative data to substantiate these claims. However, experience in assisting users in defining requirements and utilis-

ing facilities tends to indicate that they are not far fetched. Duffy and Assad (16) report findings by Dickson and Powers (14), that "70% of the information systems (IS) projects they studied were over the estimated time, and 90% were over the estimated cost" (16a).

How did this situation arise? Why do IS projects appear to be under such poor control? To answer these questions it is useful to examine, briefly, the history of computerised systems.

Until the late 1950's computerised systems were limited to high technology and scientific applications. For these early applications the prime concern was the development of reliable, usable hardware. The interest was in the performance of tasks which could not be accomplished without the processing speed of the digital computer. Software was considered of secondary importance.

With the advent of computers for commercial application - the introduction of the IBM 650 in 1956 (44) - the emphasis changed from scientific to business and financially profitable applications. Further development of digital technology could now be economically justified. Faster turn-around and more reliable output for well defined clerical procedures, such as financial accounting or the preparation of payrolls, were the obvious first applications for this new technology, since procedures for these functions are fairly standard and calculations relatively simple. In addition these activities are time consuming and repetitive, and hence, ideal for computerisation.

Encouraged by commercial success, computer hardware technology developed rapidly during the following two decades. The vast data storage capabilities and efficient retrieval and processing facilities continually opened new application areas. Competition in the market place reduced hardware prices, thereby encouraging this extended application, so that today computer applications are found in practically every area of business and industry. In addition, with the introduction of the relatively cheap "mini"

computer and "micro" processor, the implementation of computerised information systems is no longer limited to large, well financed organizations.

Early applications were dedicated to specific tasks, such as the preparation of a payroll, maintenance of financial accounts or monitoring of material issues and receipts. Software was designed for use by a single department in an organization. Transfer of information between departments was conducted manually, the computer being used to prepare reports for this transfer.

As computer technology advanced and the handling of large volumes of data became more efficient, it became evident that the integration of systems would reap considerable benefit. Data entered to the computer through one software program could then be interrogated through another. Initially this was handled by the creation of common data files, with records being accessed through a specified "key" field. The recent advent of the "data base" concept, however, allows the association of a number of record "keys". Data may now be interrogated from a number of viewpoints, e.g. the salaries office may interrogate the personnel file through an employee number and obtain a salary level for that employee, while human resources may enter the same file through a qualification code and extract a list of personnel carrying such qualifications . These advanced applications require increasingly sophisticated software for their implementation.

The implications of this trend are illustrated in Boëhm's (5) hardware/software cost trend diagram (FIGURE 2.1). Initially computer application capabilities were seriously limited by hardware capabilities and costs. However, the rapid advance of hardware technology has outpaced software advances to the extent that this relationship is now reversed. Software costs are now substantially greater than hardware costs.

For many years most managers found the concept of software costing more than hardware difficult to perceive (25b). The greater part of DP funds was allocated to the purchase, installation, operation

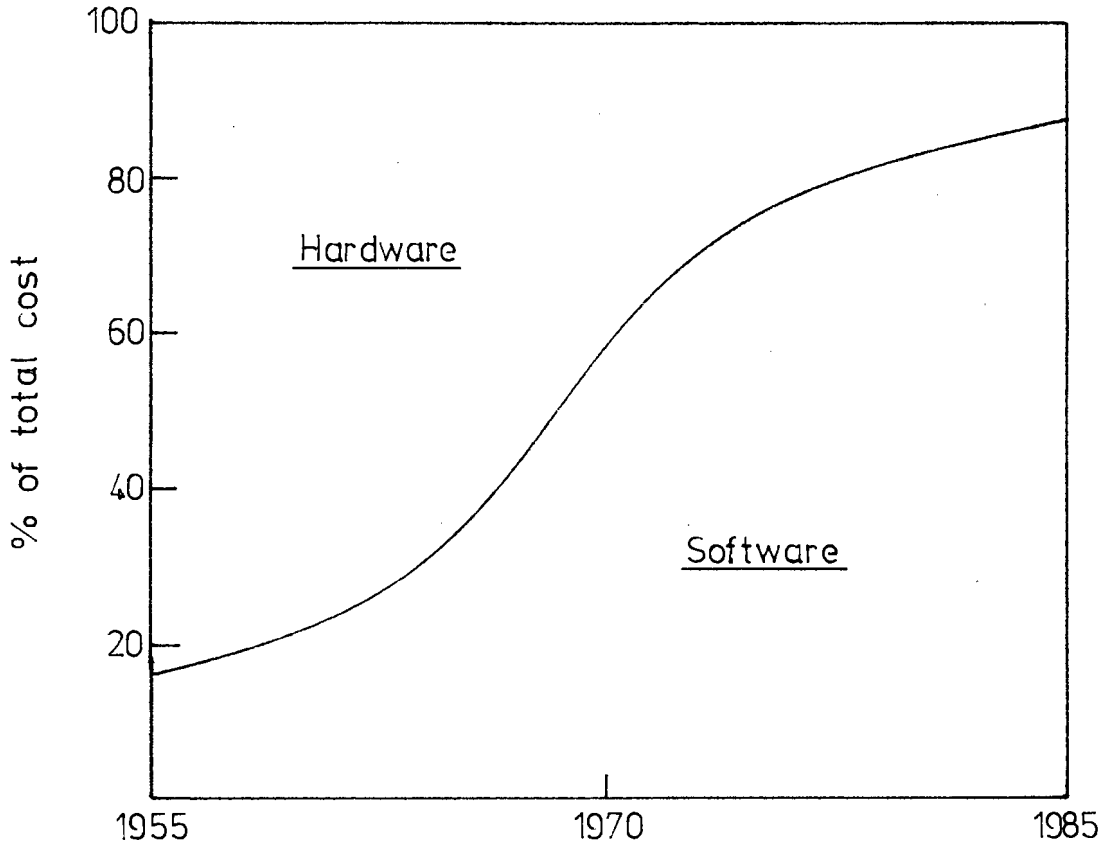


FIGURE 2.1 Hardware/software cost trends (5)

and maintenance of hardware. Software was considered almost as an "afterthought".

Software packages were often purchased without the detailed evaluation of their capabilities which usually accompanied the procurement of hardware. In general, it was assumed that the organization could "make-do" with "standard" software available from the selected hardware supplier; that simple modifications, performed in-house or by the supplier, would provide the basic facilities required by the organization. Alternatively, having installed the hardware, it was assumed that the organization's "standards" could easily be modified to comply with the system philosophy.

The desire to implement information systems as quickly as possible, in order to obtain maximum benefit from the investment in hardware, frequently resulted in software managers being required to take "shortcuts" in bringing software "online". Requirements analysis and system testing were often neglected, as was documentation. IS effort was generally concentrated on broadening the scope of the computer application, thereby proving the benefit of computerisation. Consequently "shortcuts" taken when implementing systems were seldom reviewed, and segments of the work omitted at implementation were often never completed. The result was poor performance and malfunction, requiring frequent redesigning of software, restarting of projects, and their consequent late delivery.

The extremely rapid development of computer technology (hardware and software) has also resulted in a shortage of skilled personnel. In addition, techniques for managing this new discipline have not yet been fully developed. Jensen et al (25c) sum up the "software experience of the last 10 to 15 years as one of being overtaken by events. Computer hardware technology advanced so rapidly, and the requirements for sophisticated software systems became so demanding, that a significant lag in technology, management methodology, and availability of properly trained engineers was created". In the early part of this period

software capabilities were seriously limited by hardware capabilities. As a result, software requirements were relatively simple and could be satisfied by the unsophisticated managerial and development methodologies available at that time. However, advances made in hardware capabilities far outpaced those made in the software field. This resulted in what has been described as a "crisis", with available personnel, training facilities and methodology being unable to supply the sophisticated software products demanded, within the time and cost constraints stipulated.

2.2 "CRISIS" AND ITS IMPACT

Poor management performance impacts widely on the acceptability of computerised information systems. Perhaps this can best be understood by examining the financial implications of excessive software development costs, or delays in the completion of projects. Boëhm (5) lists overall software costs for some large U.S. projects (1973).

I.B.M. OS/360	$200 000 000
SAGE	$350 000 000
Manned Space Programs (1960 - 1970)	$1000 000 000

Illustrating the effect of delays in software development on indirect costs, Boëhm (5) sites a large system having an intended life of 7 years and producing approximately $ 200 million per annum benefit to the organization. With software on the critical path, a six month delay in its completion caused a six month delay in the delivery of the system. Consequently a loss of approximately $100 million worth of needed capability was experienced - about 50 times the direct cost of $2 million for the additional software effort. These figures suggest that research into a means of improving productivity in the development of software, and of ensuring that projects are completed on time and within budget, is justified.

For smaller commercial projects, such as those in which many industrial IS departments are involved, the impact is less dramatic, but, never the less, significant. Since project-costing is generally not closely monitored, financial implications of deviations from plan are often obscured. However, schedule "slippage" can impact on relationships between users and the IS department. An example from the author's experience relates to the valuation of stocks from a history of orders and deliveries. Being assured that the system would be available in time for the annual stock take, the financial accountant made no provision for a manual valuation as had been conducted previously. However, the inconvenience suffered when the software was not completed on time resulted in considerable loss of confidence in the capabilities of the IS department.

Boëhm's well known hardware/software cost trends diagram, FIGURE 2.1 (5), illustrates the impact of exceeding budget on the cost of the total project. Twenty years ago the cost of software was minimal compared with that of hardware. In the 1980's, however, the situation is reversed, with software being the major cost element in any computerisation project. Consequently, exceeding budget on software has a major impact on the total project cost.

Software is generally on the critical path in computerisation projects (5). For those projects in which engineering controls are computerised, the software design must necessarily follow the engineering design. Consequently it may appear that software is late, other developments having been concluded prior to the specification of software requirements. Examples of such projects are, the computerised control of chemical processing plants, and control mechanisms for aircraft and machinery.

For data processing, or information systems (IS) projects, software is frequently required for the implementation of new techniques. Under these circumstances, the procedures must be designed before being computerised. Once again it may appear that software development is delaying the implementation of the new techniques. Being on the critical path in many projects, any delay

in software development will affect the implementation dates directly. This in turn affects the date on which the organization can expect to benefit from the project.

System failures often result from software's "unresponsiveness to the actual needs of the organization it was developed for "(5). Boëhm (5) cites, the example of certain hospital information systems developed in the U.S.A., which, after initial implementation, were phased out. The prime reason given for the failure of these systems was their inability to provide meaningful assistance to the users.

With the rapid advance in software technology and the ever increasing demand from industry and commerce, a concerted effort is required from software practitioners to resolve the difficulties experienced in providing effective software facilities at acceptable prices.

2.3 FORMULATION OF THE PROBLEM

Why has IS management proved so difficult, what is the difference between managing software projects and hardware projects, since the latter appear to be relatively well managed, and what are the major obstacles to the successful management of computerised information systems?

2.3.1 THE ABSTRACT NATURE OF SOFTWARE

Numerous authors have analysed the basic differences between software and other engineering products (25,17,3,10). Engineering products are, generally, material objects. As such they can be sensed, touched and observed. The results of such observation can be recorded as a measure of product performance, and used to monitor improvements made in successive design modifications. The preparation of specifications and drawings for hardware products involve the development of a number of proto-type models, the design of each successive model being a refinement of the previous design. Performance of each proto-type is monitored and the result

used to determine the design refinements necessary for the following model. This procedure is continued until satisfactory performance is obtained from an economically viable product.

In contrast, software is of an abstract, or non-material, nature. This limits the evaluation of software products. Software products are unlikely to be subjected to as many proto-type evaluations, or test stages before reaching the market, or user environment, as are hardware products. Certainly, the major performance evaluation is only conducted in the field.

Cooper (10) points out that, while many software orientated personnel consider software as fundamentally different from hardware and consequently propose that design principles and production management disciplines be modified to accommodate these differences, hardware orientated personnel feel that software is no different from hardware, and thus requires no modification of these disciplines. The solution lies between these extreme views, and appropriate application of the standard disciplines should be effected in each field.

Despite differences between hardware and software, the basic engineering design principles and fundamental production management disciplines of quality assurance, configuration management, reliability and maintainability of the end product, apply equally in both areas (10). However, in view of the fundamentally different characters of the two product types, the techniques for applying these principles might be different for the two fields. For example, for hardware, measuring instruments such as micrometers, calipers, etc. are readily available to ensure that component dimensions are within specified tolerance. For software however, different techniques for applying Quality Assurance measures to a product are required and a different set of tools from those for hardware products must be developed to support the Quality Assurance function.

The difference between hardware and software projects is perhaps best illustrated by comparing typical resource loading curves for

each project type (FIGURE 2.2 A & B) (9). For hardware, following initial conception, a number of proto-types may be developed before a full scale model is produced. Usually, only a small fraction of the total project resource allocation has been utilised at this stage. As indicated in FIGURE 2.2 A, the peak loading occurs after demonstration and testing of the initial operational capability (IOC) of the product. For software, however, the peak occurs substantially earlier than the IOC, as indicated in FIGURE 2.2B, i.e. the major resource allocation for the project has been utilised before the capabilities of the product can be demonstrated.

2.3.2 MANAGEMENT ORIENTATION

Perhaps the greatest single obstacle to the successful management of computerised information systems is the "lack of computer related experience on the part of the corporate decision makers"(10). Senior managers often lack an appreciation of the "unique complexities of software development" (10), and of the differences between the development of hardware and software.

Since software is "relatively new, intangible, and has evolved with an aura of mystic" (10), it is poorly understood. Industrial managers, being hardware orientated, tend to expect the control procedures applying to the development of hardware to apply equally to that of software. It is often felt that the allocation of additional resources to a software project will shorten its development time. Managers are seldom aware of the incompressibility of software projects with respect to time (6)*. In addition industrial and business managers tend to be short sighted with respect to software, requiring that only their most pressing needs be fulfilled. They assume that future requirements

---

FOOTNOTE :

* The incompressibility of software projects will be discussed in more detail in Chapter 5.

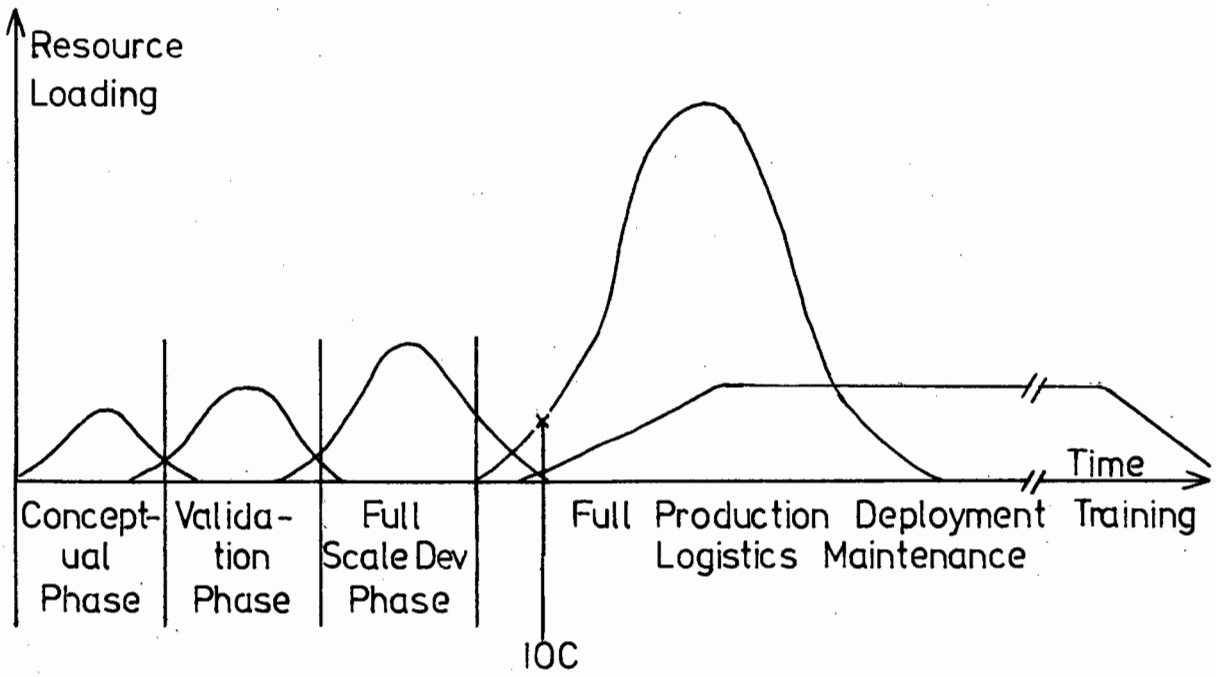


FIGURE 2.2A Hardware Life-Cycle Resource Loading (9)

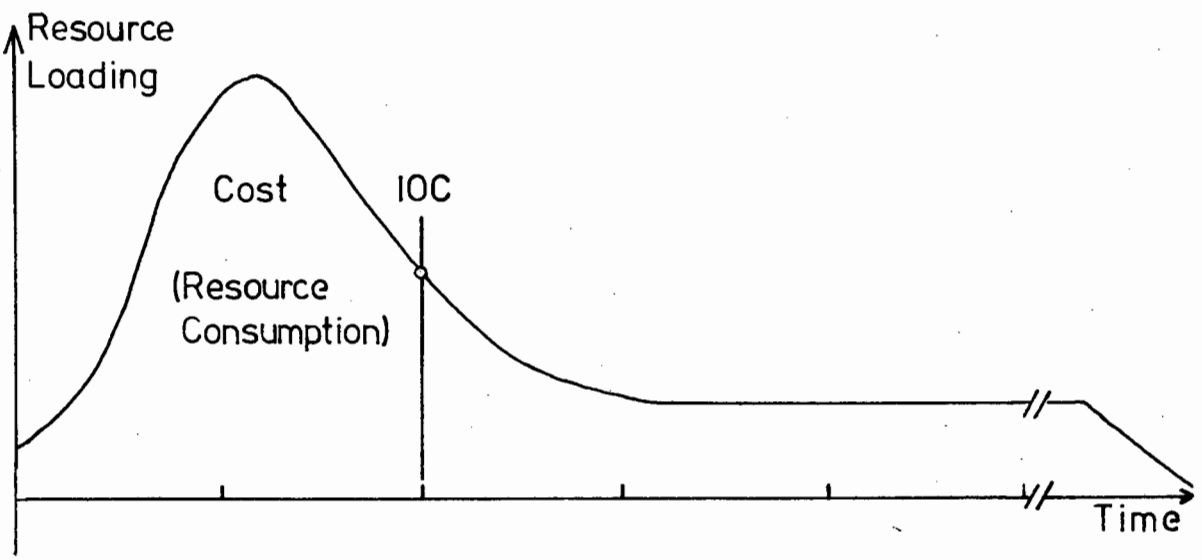


FIGURE 2.2B Software Life-Cycle Resource Loading (9)

will easily be satisfied by a relatively simple extension of the system. A term which the author has frequently heard used in this context is that the system should be "expandable". By this it is implied that the system should allow requirements, not initially specified, to be easily incorporated. Managers seldom anticipate the extensive resources required to satisfy their extended requirements. They are seldom aware of the trauma such an extension to "live" software can create, unless it has been specifically planned.

To some extent their attitude is understandable. Implementation is often so far behind schedule that it provides little or no assistance to users in the fulfillment of their commitments. (As an example refer to the development of the stock valuation system discussed in section 2.2). In addition, managers often find that the final product bears little resemblance to that which they expected.

Computerised information systems are often developed in the hope that they will alleviate problems experienced in the performance of some task. Bearing in mind the initial uncertainty of requirements, and the reputation IS has for late delivery of new products, managers might be considered justified in initially requesting only the bare essentials. In this way they hope that at least some facility will be produced and delivered on schedule.

With the proliferation of user dissatisfaction, IS project managers tend to be development orientated. Their most pressing responsibility is "the development of their system within budget and on schedule"(10). Consequently, the development process is often optimised at the expense of overall life-cycle costs. Documentation and testing are frequently curtailed when cost or implementation dates are threatened. (10,5). In addition, systems capabilities may be reduced in an effort to satisfy the users immediate requirements, or to demonstrate the capability of the IS department to supply the required service. This creates a similar problem to that of requests for partial fulfillment of requirements (discussed above). Both situations involve

sacrificing product quality in favour of the timeous availability of the system.

The frustration of both users and management at not having reliable estimates for planning purposes is often amplified by the lack of "standard software (or IS) practices". Cooper (10) points out that variations in technique require variations in interpretation. To the unfamiliar, changes in terminology, variance in the break down of project life-cycles, the setting of different milestones, etc, present dilemmas in keeping informed as to the status of software projects. Different techniques, for example, in specifying requirements, reporting on software facilities which have been included, or reporting on progress, generate confusion amongst non-IS personnel. For senior managers, such an environment makes comparison of performance between projects impossible.

Unlike other branches of engineering, in which control parameters are well defined and their relationships well understood, project control in the software industry is poorly developed. Few tools and techniques are accepted widely enough to be considered standard, and results of different techniques are seldom compared with a view to optimising control. Many project leaders/managers consider their task unique, requiring unique control procedures. However, it is seldom that these procedures are formalised.

2.3.3 PROCEDURAL PROBLEMS

The inability of IS personnel to provide reliable estimates of project duration, resource requirements, or project costs, is a constant source of frustration to senior managers. In the author's experience numerous software products have been delivered too late to assist with the immediate problem for which they were intended. As an example consider the stock valuation system discussed in section 2.2. Having initially planned the valuation under the assumption that computer facilities would be available, the user suffered the frustration of having to replan the exercise when informed that the software would not be completed on time.

The later implementation of the system, and the requirement that data be maintained for a full year, until the next stock take, before any benefit could be derived from the system, further aggrevated the situation (particularly when that benefit had not yet been proven).

Cooper (10) finds this inability to analyse software development parameters, and the consequent introduction of unknown elements into planning and control function a "major contributor to the poor image software possesses", particularly among senior managers.

A further example of the deficiency in planning and control procedures for IS projects can be drawn from the author's personal experience. No attempt had been made to formalise controls, prior to his appointment as head of an IS department in a large manufacturing organization. Project leaders were considered sufficiently in control of their projects, and were not required to file project development plans, or progress reports. Consequently the department possessed no historical data concerning the size, duration, manpower requirements or cost of previous projects. In addition no record existed of the "status" or "degree of completeness" of projects.

In this situation, IS staff are often reluctant to commit themselves to a specific schedule, arguing rightly that they can't be sure of what they need to do. No clear definition of project requirement has been produced, and the user's perception of his priorities changes with each contact.

The risk of such an approach is that many projects are never completed. Tasks which staff find unpleasant, such as documentation, are seldom even initiated. No record is kept of modifications to specification, nor of the implementation of these modifications. In the maintenance phase of a project the functional content of a system is uncertain, so that the effect of maintenance modifications is never clear. When, in the closing stages of a project, project leaders are requested to assist in

other areas, or take on new projects, it is seldom specified that current responsibilities be completed first. Consequently, the final activities of the current project are often neglected. (Of course these activities are seldom even specified).

In this environment, management can exercise little control over projects. It is impossible to monitor staff performance effectively, and resource allocation or requirements are left to the ad hoc judgement of the various project leaders.

2.3.4. THE STAFF TURNOVER PROBLEM.

Staff turnover is a major problem in the computer industry. A recent survey in the United States showed a turnover rate of 28 percent, overall, and 34 percent for application programmers (30,11a). For the IS industry in South Africa, turnover exceeded 30 percent during the past two years (48). At this rate approximately half the staff currently employed will have left the organization within two years; within 4 years it can be expected that the total staff will have been replaced (30). Clearly, for computer personnel, the tendency to "job-hop" has assumed epidemic proportions.

The costs incurred in replacing staff, and particularly in transfering responsibilities for development projects, or program maintenance, to other staff members can be considerable. Being an infant industry with a rapidly advancing technology, no generally accepted technical or control standards have been established in the software industry. As a result, programming language may be highly individual; technique highly variable; symbology, abbreviations and mnemonics inconsistent and incomprehensible to any one other than the original programmer. In addition, for large programs, the logic may be so complex as to baffle all but the most competent programmer. Consequently, once the original programmers have resigned from the organization, changes to a program require a complete redesign and rewrite (7). This argument is supported by a study conducted by Lehman (27) who

found strong correlation between staff turnover and slippage on the development schedule for software projects. The results of this study are shown in TABLE 2.1.

In the author's experience, staff turnover in the Data Processing department has presented a serious threat to the successful implementation and continued maintenance of many software systems. As systems may take 1 to 2 years to develop and implement, and the average period of service for programming staff is approximately 3 years, projects are frequently completed by programmers other than those who initiated them. Often system design and development started by one staff member is not acceptable to that staff member allocated the task of completing the project following the resignation of the first. Frequently documentation carries insufficient detail to ensure that the design instigated by the first programmer can be coded, tested and implemented by his successor. The resultant delays in the completion of projects, duplication of work, loss of continuity and systems quality, have been a constant source of frustration to D.P. managers in the past.

Software projects are seldom planned in any detail (7). Development and implementation schedules are often exceeded, and, consequently, "testing" and "documentation" are suspended until after implementation. In general, the heavy development and maintenance backlog, prevents the satisfactory completion of testing and documentation phases. Consequently, once the original programmer has resigned, the modification, enhancement or maintenance of a system can be very time consuming, often nigh on impossible.

2.4 SUMMARY.

The rapid advance of computer technology during the ~~passed~~ past 30 years has resulted in an ever increasing demand for sophisticated IS services. Advances in hardware technology and improved production methods have extended the application of computers to all areas of business and industry. The recent reversal in the

Position	Staff Turnover	
	Average for Projects Sampled	Average for Late Projects
Project Manager	68%	90%
Functional Analyst	60%	97%
DP Analyst	20%	28%
Programmer	38%	53%
Support Librarian	32%	37%
Secretary	63%	79%
Administration	33%	40%
User Representation	32%	39%
Other	9%	0%

TABLE 2.1 Impact of Personnel Turnover on Development Schedules for Software Projects (27)

relative cost of hardware compared with that of software shifted management concern to the control of software projects. The impact of delays in the provision of software facilities on the profitability of a business or industrial project is now substantial, so that the development of an efficient and effective management methodology for software projects is now essential to any organization utilising a computer based IS service.

Problems facing IS managers can be grouped into four distinct categories.

i) IS personnel are often reluctant to accept that fundamental design principles and management disciplines applying to hardware, should also be applied to software products. They argue that hardware and software are fundamentally different products and that different control principles apply to their development. However, they fail to realise that the nature of the software product simply requires that techniques for applying standard principles might differ from those applied to hardware.

ii) Limited technical experience of computer related projects results in senior managers lacking an appreciation of the "unique complexities of software development"(10). These managers seldom realise that different control procedures are applied to software development from those applied to hardware projects. They are often short-sighted when defining requirements, seldom anticipating the extensive resources required for apparently simple modifications to software.

NB

iii) Standard control procedures for IS projects seldom exist in an organization. Estimates of project cost or duration are seldom accurate, so that resource allocations cannot be controlled. In addition, no

basis for monitoring staff performance can be provided.

iv) Excessive staff turnover in the computer industry often accentuates the poor control exercised over IS projects.

CHAPTER 3

MANAGEMENT IN PERSPECTIVE

Too often IS departments are viewed as self contained, independent entities. Within these departments systems development staff frequently become obsessed with implementing the most sophisticated or advanced technology available. A prime task of management must therefore be, to keep systems development in perspective, to ensure that sight is not lost of the ultimate goals of the organization of which the IS department is but a part.

In the author's opinion the most important factors to be borne in mind in this respect include:

i) User acceptance of the services offered.
ii) Technical feasibility of any project undertaken.
iii) Economic/financial justification of projects or services.

3.1 USER ACCEPTANCE

The ultimate measurement of the success of an IS management methodology is the willingness on the part of clients, or prospective clients, to utilise services offered (9). IS management strategy should, therefore, be strongly geared to satisfying user requirements.

Cave and Salisbury (9) note that many project failures result from poorly specified requirements. This poor specification may result in operational procedures for software, which conflict with user philosophy. Since the software product must, to some extent, dictate the policies and procedures adopted by the user, this can lead to user rejection of the software product. Obviously, accurate and complete specification of requirements including an implementation schedule, must form a major objective of any IS management strategy.

3.2 TECHNICAL FEASIBILITY

Confirmation of the technical feasibility of a proposed system must be a pre-requisite to the continuation of any project. The status of the data/information required for the successful implementation of the system, the capabilities of the end user and of the development team must be considered. A system cannot be expected to perform satisfactorily if it is too complex for its end user. By the same token adequate facilities cannot be provided by a team lacking the basic skills and knowledge required for the development of those facilities.

It is often presumed that the implementation of a particular software package will solve specific problems, that the computer will inform the user of his most suitable course of action in difficult situations. In addition the computer may be held responsible for incorrect decisions taken by users or management. However, in assessing the technical feasibility of systems proposals, it must be recognised that computerisation cannot provide solutions to problems (45). These solutions are required before the computer system can be designed and implemented. Computerisation can only provide tools for storing and retrieving information, for sorting information, for performing rapid repetitive manipulation of data, or for the application of pre-defined formulae in the analysis of data. As such, computerisation aids manual decision making, providing both easy access to accurate, up to date information, and the facility for sorting this information into a format, representative of a particular situation. However, computer systems cannot make decisions*.

FOOTNOTE :

* Under certain circumstances, e.g. machine control, scientific experiments etc. programmed decisions are made. However, in these cases, all decision rules and parameters must be fully defined. Certainly for management decisions, this is not possible, and, therefore, programmed decisions should be excluded from IS applications.

3.3 ECONOMIC / FINANCIAL JUSTIFICATION

Economic or financial justification for systems development is essential to convincing senior management of the value of the organization's investment in computer systems. This requires a comparison of systems expenditure against benefits. Obviously, more efficient control of systems expenditure will improve this cost/benefit ratio.

The efficient and effective control of development expenditure requires a comparison of actual costs with planned financial disbursement. The author has found, that while project costs are monitored in computer bureaux and software organizations this does not appear common in IS service departments. These departments usually operate on a fixed budget, recovered at a predetermined annual rate from user departments. It would appear therefore, that financial management of IS projects receive greater attention when software is the major product of the organization concerned, (as in the case of a bureau), than when software is merely required as a service to the manufacture of other products.

In the author's opinion, were IS service departments to act in open competition with outside bureaux, financial management of these departments would improve markedly. Product quality would also improve. Such improvement would be motivated by users being free to utilise services offered on the open market, were those offered by their own IS departments found to be unsatisfactory, or excessively expensive*. In addition, user management would ensure

---

FOOTNOTE :

* Obviously the total cost to the organization must be considered. The provision of computer facilities is a costly endeavour. Once provided, maximum benefit from this investment should be sought. However, system upgrades or extensions should be motivated on an economic basis, comparing costs of in-house provision of the service with utilisation of services offered by outside organizations.

that they were familiar with the computer facilities utilised in their departments. Since they would be directly responsible for the costs of services utilised, they would be more likely to ensure that maximum benefit be derived from these services.

A first step toward open competition with outside organizations must be improved control of software costs. A management methodology must provide for the management of tasks in terms of resource capacity requirements and time. Initially these measurements are predicted, and the calculated costs compared with savings or benefits expected from the implementation of the software.

Techniques for minimizing project costs greatly influence the financial justification of the project. Management strategy should be aimed at maximizing efficiency, i.e. minimizing wasted effort. The sequence of activities in a software development methodology should therefore be designed to minimize the risk that the work will be abandoned, or require iteration as cumulative effort input to the task increases. The methodology should ensure that increased effort only be allocated to the task in proportion to the confidence gained in its ability to satisfy requirements.

Throughout a systems "life-cycle" emphasis should be placed on ensuring the most efficient and effective performance of the project as a whole (15) Numerous authors (34,10,9) have demonstrated the folly of reducing the cost or duration of a specific phase of the life-cycle while ignoring the impact such actions have on the overall cost, or duration of the project. Overall project efficiency may be forfeited if emphasis is placed on initially satisfying only the most prominent user requirements. Although, having fulfilled these initial obligations, attention may shift to satisfying other requirements, these later developments frequently require interfaces (with either users or other systems) not included in the original design. Enhancement of a system often requires a redesign of the original, to include these additional interface facilities. When viewed as a total project, this procedure is considerably less efficient than if

allowance had been made for these facilities in the original design. This does not imply that the system need necessarily be fully developed from inception. "Dummy" activities may be set for those facilities not urgently required at project initiation. However, including all known requirements in the original design allows more efficient development at a later stage.

Having to justify financial expenditure would enforce the introduction of cost monitoring, thus allowing more effective control of IS expenditure. In addition, IS managers would be motivated to introduce the efficiency improvements described above.

3.4 SUMMARY.

IS departments form part of a larger organization and, as such, IS activities must support the ultimate goals of the organization. Rather than allowing IS personnel to become obsessed with sophistication or technological advancement, the following factors should be borne in mind.

i) User acceptance is the ultimate measure of success of IS services. User agreement on systems operation should be obtained prior to further development.

ii) Technical feasibility must be confirmed prior to the authorisation of an IS project.

iii) Economic or financial justification should be a pre-requisite to any IS project.

CHAPTER 4

QUALITY ASSURANCE FOR SOFTWARE

4.1 CONCEPT AND OBJECTIVE

Classically, the satisfactory performance of a product is demonstrated at the end of its development, by means of a product test designed to establish whether requirements have been fully satisfied (8). For software, this approach has seldom proved sufficiently convincing for a number of reasons :

i) Software requirements are rarely fully, or explicitly defined. In addition, they are often dynamic, clarity being derived from queries raised during the software development process. Frequently software projects are linked to hardware projects, the software requirements only being finalised once the hardware development is completed. By this stage it is generally expected that software development will also be nearing completion.

ii) Software testing is rarely exhaustive. Seldom are all combinations of input data and processing sequence considered, or all branches of a program examined during the test phase of a systems development. Consequently, despite an apparently successful test demonstration of a products capabilities, errors may emerge in the newly implemented system once combinations of input and processing sequence - not included in the original test program - are encountered. Since users generally lack an understanding of software problems, these errors generate a distrust of the original test results or procedures and of the products capabilities.

Quality assurance for software, involves resolving these problems, thereby instilling confidence in the product. For software practitioners this task is complicated by a number of factors (8).

i) Many methodologies are practised for the development of software, but no standard methodology exists to ensure that the software developed satisfies specific requirements.

ii) There is no generally accepted standard for expressing technical details of software attributes, nor for defining test procedures.

iii) No industrial standard exists, as yet, for expressing plans to implement assurance programs for software development.

As a first step toward implementing a quality assurance program for software, the concept should be clearly defined, and the objectives of the program established. A generally accepted definition of quality assurance, applying equally to hardware and software, is as follows (25d,8):

> "A planned and systematic pattern for all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements and achieves satisfactory performance" *.

From the above two primary objectives for any Q.A. program can be deduced:

i) The first involves the establishment of acceptable technical standards, thereby generating confidence in both the capabilities of the finished product and in its continued satisfactory performance.

FOOTNOTE :

* This definition is derived from a combination of those adopted by the US Department of Defence and the Institute of Electrical and Electronic Engineers subsection on software quality assurance (25d,8).

ii) The second is to inform all concerned, responsible, or affected persons of the status of the product, allowing sufficient time for errors, deficiencies or deviations from requirements to be detected, reported and corrected with minimal effect on the product cost or production schedule. The goal is to provide assurance that the product will satisfy requirements, that all requirements will have been identified, and that the product will be delivered on schedule and within budget.

These objectives can be achieved through the implementation of a Q.A plan for software projects (refer Section 4.3). Such a program would be justified by improvement in productivity during the development and maintenance of software, and improved effectiveness of the end product.

4.2 JUSTIFICATION FOR IMPLEMENTING QUALITY ASSURANCE PROCEDURES FOR SOFTWARE.

Buckley (8) identifies three major reasons for introducing a quality assurance program.

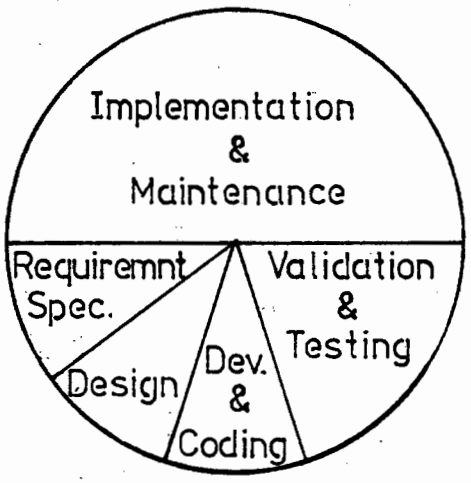
i) Legal Liability : In the event of a product failure, the purpose of the program would be to provide evidence that the producer acted as a reasonable and prudent person in developing the software. Here, only critical software, such as military or medical applications, where failure could threaten safety or result in large financial or social losses, is of specific importance. A standard for software quality assurance plans, directed at the development and maintenance of this critical software, has been prepared by the Computer Society Software Engineering Standards Subcommittee (U.S.A.) This will be refered to in the preparation of a general quality assurance plan for software development and maintenance in industry (Section 4.3), where criticality is generally not as high as in military or medical applications.

ii Customer requirements : Whether the customer be an outside organization or another department within the same organization, confidence must be established in the ability of the IS Department to produce and maintain a usable and satisfactory product. The level of assurance required depends on the relationship between the user and producer, and, to some extent, on the user's perception of his needs. Often the Quality Assurance program will be initiated by the software producer in an effort to win confidence in his product.

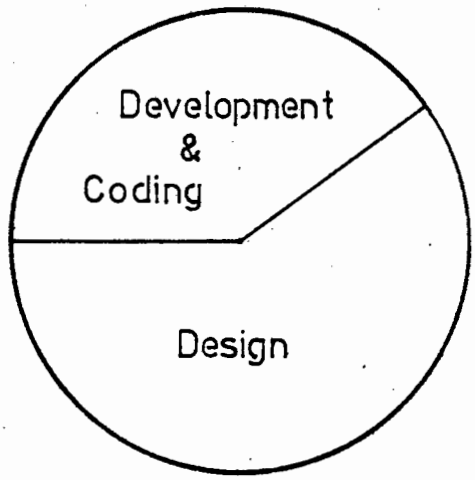
iii) Cost-effectiveness: The introduction of a QA program should be aimed at improving the cost-benefit of a particular project or undertaking. Although no quantitative evidence has been found, in the published literature, to support the notion of cost-effectiveness in the introduction of QA, it would seem advantageous that any QA program should be aimed at reducing such cost-ineffectiveness as is evidenced in the examples given below.

A basic measure of quality in software is the level of "errors" detected, be they in the pre-release development period, or in the implementation and maintenance phase of the project. As mentioned earlier these "errors" may be viewed as deviations from the original intentions or objectives of the systems designer, or from the "true" requirements of the end user. Correction of "errors" and enhancement of the system to meet user requirements form part of the project maintenance function. Thus, the maintenance effort required on a project gives some measure of the product quality.

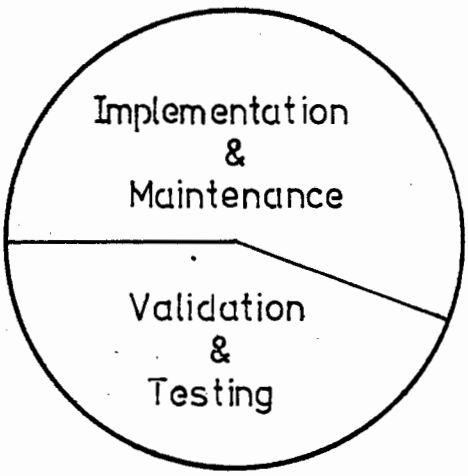
Cost, error, and reliability aspects of the system's life-cycle are illustrated graphically in FIGURE 4.1 (19). As shown, by far the largest cost is incurred in the maintenance phase. Minimizing the maintenance effort required, or improving its efficiency or effectiveness is therefore an essential element of a quality assurance program.



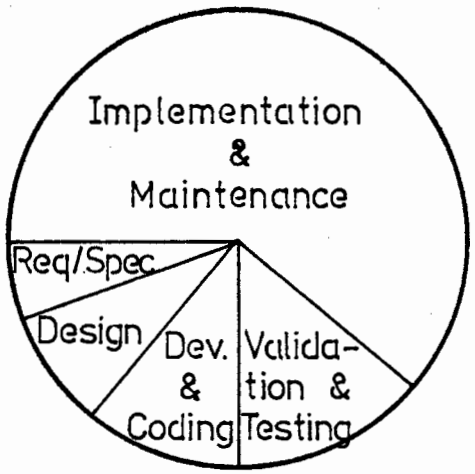
Costs per Phase



Error Sources per Phase



Error Discovery per Phase



Per Error Cost
of Correction per Phase

FIGURE 4.1 Cost, Error, Reliability Aspects of the Software Life-Cycle (19)

Literary references (5,10), and discussions with managers and consultants indicate that most IS managers are perplexed by the problem of where the additional effort, required to achieve maximum benefit, should be applied. As regards error correction in software, some guidance is provided by Sorkowitz (41), (FIGURE 4.2). He has demonstrated that the cost of correcting an error increases with time. For example, an error not identified in the unit testing phase will take longer to isolate in the system-testing phase. Where program units are tested individually, an error can easily be related to the unit under examination. However, in the system-testing phase, where all programs comprising the system are tested together, i.e. as an integrated system, it may take considerably longer to associate the error with a specific program unit.

Sorkowitz (41) also identifies a second relationship which follows a similar curve, viz. the probability of fixing a known error incorrectly increases in time, i.e. as the project progresses through the development life-cycle. This affects the ultimate cost for error correction. As an example a study is cited which showed that a requirement error detected in the unit-test stage is five times more costly to fix than if detected in the requirement-definition stage. This same error, if only detected at the integration and system-testing stage, would be 36 times more costly to fix.

These relationships between the cost of error correction and time, or the stage in the project life-cycle, illustrate the benefits to be expected from increasing the effort to ensure that errors be detected and corrected early in the process. Ensuring that the input is correct and adequate from inception is a basic function of quality assurance.

Lientz and Swanson (28) have conducted the most in depth survey and analysis of the maintenance function which the author has found to date. Their survey revealed the allocation of maintenance effort to various task types, as shown in Table 4.1. Of these, the first four, accounting for 31,2 percent of the total

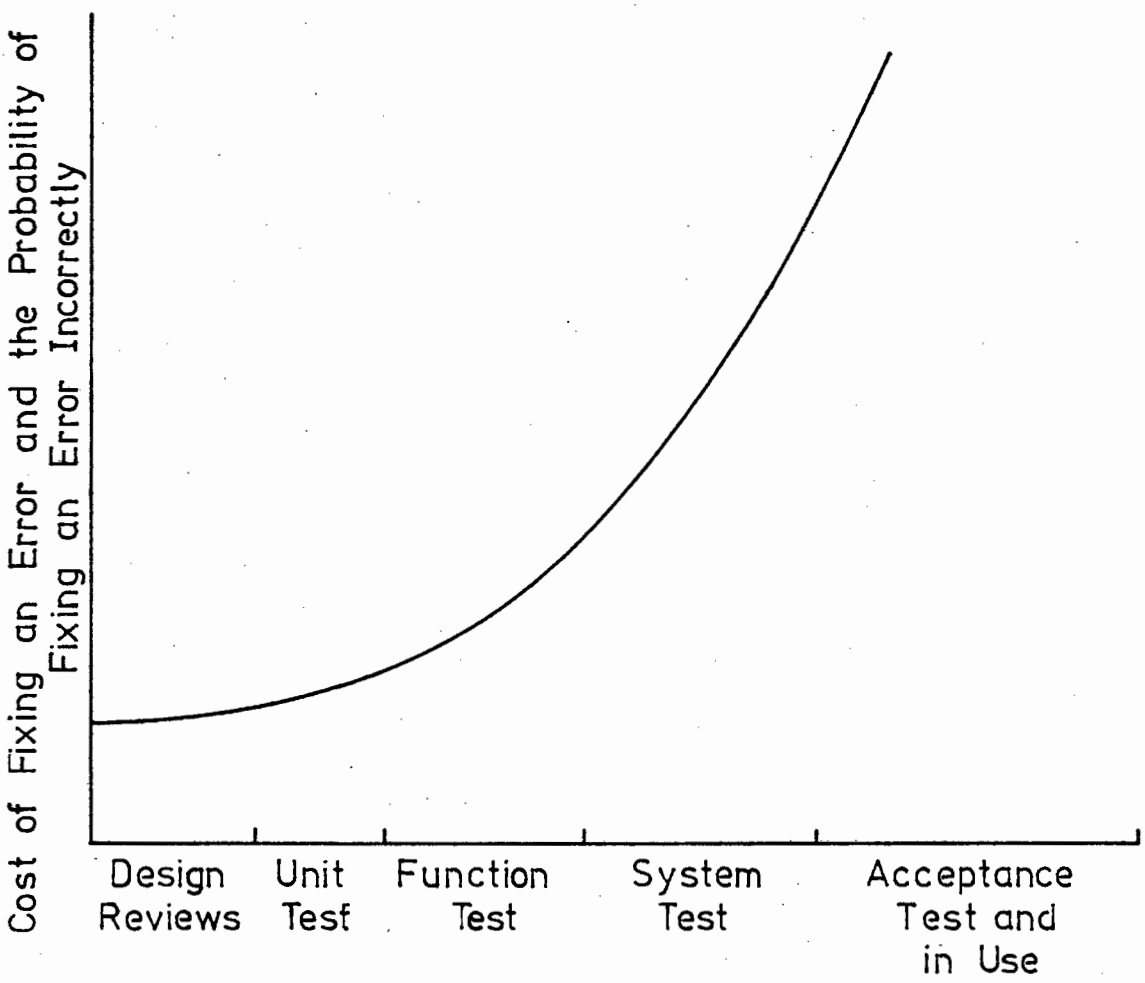


FIGURE 4.2 Cost of Error Correction as a Function of Time (41)

Task	Mean Percentage
Emergency Programme Fixes	12,4
Routine Debugging	9,3
Improvement of Program Documentation	5,5
Enhancement for Efficiency in Computation	4,0
Accommodation of Changes to Hardware and System Software	6,2
Accommodation of Changes to Data Inputs and Files	17,4
Enhancements for Users	41,8
Others	3,4
	100,0

TABLE 41 Allocation of Maintenance Effort to Various Task Types (28a)

maintenance effort can be directly associated with errors or deficiencies in the system which might have been detected earlier. A portion of the effort devoted to user enhancements, viz. reformatting (approx. 10,0 percent), might also be avoided if these enhancement had been included in the initial specification of requirements. This implies that approximately one third to one half of the maintenance effort could be avoided if suitable quality assurance procedures were applied.

4.3 A Q.A. PLAN FOR SOFTWARE PROJECTS

The preparation of a Q.A. plan for software projects is the first step toward developing a management methodology capable of achieving the quality assurance objectives discussed in section 4.1. Its purpose is the "planned" development, implementation and maintenance of software. This is achieved by informing all concerned persons of their responsibilities, of the manner in which their tasks are to be performed and of the date by which these tasks should be completed. Careful monitoring of progress allows corrective action to be taken when deviations or delays occur.

Dependent tasks can be notified when deviations from original plans or specifications have been necessary in preceding tasks. In this manner the deviations can be accommodated in succeeding work. Delays can be accommodated in the overall schedule by shrinking the duration of the succeeding tasks, i.e. by allocating the required additional effort to these later tasks.

The format and content of such Q.A. plans could vary widely. However, for a single organization, some uniform means of expressing such plans is desirable. This "standard" should be designed to suit the organization's financial and technical means, management structure, philosophy and procedure, and the type of software product under development.

A number of "standards" exist for Q.A. plans(8,17). These provide a checklist against which plans can be measured, but are not

intended to dictate a format, or procedure, for producing Q.A. plans. These published standards normally relate to the preparation of software for military purposes, or for "critical" software (8) in which failure could result in large financial losses or threaten safety, (refer 4.2 (i)). Such stringent control requirements do not apply to the software developments considered in this study, and therefore, as suggested by Buckley(8), a subset of these standards has been selected as a minimum requirement for Q.A. plans for software (see Appendix A). This standard for the organization could be upgraded at a later stage, to accommodate "critical" software. Implementation of the concept of a QA plan for software, and maintenance of the standard to ensure continued applicability in the organization, will be discussed in the following section.

4.4 IMPLEMENTATION OF A Q.A. PROGRAM

Two basic approaches may be identified for the structuring of a Q.A. program. The first places responsibility for achieving quality on a Quality Department. Divorced from production, this department analyses the causes of poor quality items and determines corrective measures required to upgrade that quality. The department is responsible for checking machine settings, inspecting final product, etc. The second philosophy involves all levels of management in attaining satisfactory quality in the end product.

In IS departments in this country, other than those of the largest organizations, the establishment of a section responsible purely for the quality assurance function would seem an excessive and unjustifiable luxury *. The author favours rather that the head

---

FOOTNOTE :

* In developing a "Standard for Software Quality Assurance Plans" the IEEE Computer Society Software Engineering Standards Sub-committee consciously recognised that a separate quality section would not be possible in all organizations, as it is in the U.S. Department of Defence (8)

of the department authorise a senior member to establish and maintain a set of "quality standards". Project leaders would then be held responsible for upholding these standards on their specific projects. Adherence to standard procedures would be monitored throughout a project and deviations reported to the department head, who carries overall accountability for product quality.

This monitoring of a project's adherence to quality standards could take the form of a quality audit, conducted at regular intervals by a member of the department not involved in the basic development. This person, following commissioning, would be required to approve the project to the level at which he would be prepared to assume responsibility for its maintenance. Routine reviews of product design and development could also serve this purpose. This will be discussed in more detail in Chapter 6.

Prior to the author's appointment as head of the IS department in his present organization, software staff had each been assigned responsibility for the development and maintenance of specific systems. Other staff members required little or no access to these systems, nor knowledge of enhancements or modifications. The assigned staff members had to attend to all operational and maintenance problems on their specific systems, to liaise with users on requirements and to develop software to satisfy these requirements.

Under this arrangement, users approached the responsible staff member directly with requests for additional or modified facilities. This staff member, in many cases a relatively junior member of the department, had authority to modify systems at his own discretion. Consequently, senior management exercised little control over the development or enhancement of systems.

Under such conditions staff tend to become undisciplined; modifications to the system are seldom properly planned or adequately documented, the argument being that only the responsible staff member need know of these alterations. Numerous

excuses are offered when the more unpleasant sections of the work, such as documentation, fail to be completed on time, and reasons can always be provided for the need to satisfy the users' immediate requirements, before concentrating on planning the job properly and producing a reliable, maintainable product.

Under these circumstances management no longer control which tasks are to be performed, by whom, or when. Instead, decisions on development enhancements and modifications are taken at a lower level and reported to management after the task has been completed, by which stage the system has often already degenerated. In addition no provision is made for enforcing discipline.

Quality Assurance techniques have yet to be accepted as tools for controlling software projects. In the author's experience Q.A. is poorly understood by software practitioners, Q.A. tasks often being considered unnecessary, time consuming and ineffective. Under such conditions quality assurance tasks are not willingly undertaken.

Enforcement of the Q.A. discipline requires that development and maintenance effort be reorganised. Instead of staff being held responsible for the continued maintenance and development of a system, they are assigned specific tasks, with agreed target dates for completion. These tasks include Q.A. functions, which are enforced by simply not assigning the staff member a new task until the previous assignment has been completed. Liaison with users on their requirements for additional or modified facilities, is conducted through senior IS staff only. All requests are considered at regular planning meetings at which staff are assigned specific projects with specific goals. Progress is monitored on each project to allow timeous corrective action to be taken on deviations from plan, and to allow advanced estimates of resource availability (manpower) to be made prior to confirmation of the planned schedule for the next planning period.

A Q.A. plan, as described in section 4.3, is prepared for each project. All documentation is carried against the project until the final review. At this stage documentation is copied and transferred to the systems file concerned.

4.5 SUMMARY

A Q.A. program for software, or IS, has two primary objectives: to establish confidence in the technical capabilities of the product, and advanced detection of errors, deficiencies or deviations from requirements. Introduction of Q.A. procedures for the development and maintenance of software, or IS, can be justified on the following grounds;

i) Legal liability; for critical software, where failure could threaten safety, Q.A. procedures could provide evidence that reasonable precautions had been taken during development.

ii) Customer requirements; user confidence must be established in the ability of the IS department to develop and maintain a stable, usable product.

iii) Cost-effectiveness; the cost-benefit achieved through the implementation of any IS product can be improved by advanced detection and correction of errors, and, by avoiding deficiencies or deviations from basic requirements during the development process.

A Q.A. plan for IS projects is intended to inform all concerned persons of their responsibilities, so that tasks can be performed in an acceptable manner and be completed within the set schedule. Monitoring of progress against this plan allows corrective action to be taken when deviations or delays occur. Minimum requirements for Q.A. plans are described in Appendix A.

The author favours an approach to the implementation of a Q.A. program which would involve all members of an IS department in

attaining satisfactory quality in ~~it's~~ its end product. "Quality standards" are established by senior management, and project leaders are then held responsible for upholding these standards in their specific projects.

CHAPTER 5

PROJECT MEASUREMENT AND CONTROL

The acceptability of a management methodology for IS projects depends largely on its ability to "produce the goods". Since users expect IS facilities to be available as and when promised, the prime objective of the "project management" function should be to ensure that these expectations are met.

Project Management comprises four basic activities:

i) planning
ii) implementing
iii) monitoring
iv) controlling

Execution of these activities involves estimating the duration and resource loading of the project, analysing the risks of various approaches, implementing the optimum plan, and monitoring progress to allow corrective action to be taken as deviations from plan are reported. For software however, the art of planning and controlling projects is still in its infancy. Inability to manage projects is manifest in excessive development and maintenance costs, delays in implementation, and excessive errors in the finished product (13) (see also chapter 2 above).

Few managers are able to provide reasonably reliable estimates of cost, resource requirements or completion dates (5,6,34,35). The traditional approach to this task has been to estimate the size of the required system in terms of the number of statements or lines of code, and then to calculate the total cost of the project from some empirical statement/cost relationship, (for example, average number of statements/day/programmer). The time required for the completion of the project is then related to resources available. In the author's experience the statement/cost relationship, or measure of productivity, is often not monitored for the specific software team concerned, but obtained from general literature.

Seldom is a formal estimate of the size of the program made. Generally resource requirements and development times are estimated "from past experience", taking only the broadly defined user requirements into consideration.

Putnam (34) suggests that this approach may be reasonably effective for small projects, i.e. programs built by one man or a small group of persons. However, for larger projects involving many programs, built by multiple teams of people, the approach is totally ineffective. The truth is that in absolute terms, deviations from the initial estimate for small systems are probably less noticeable than for large systems. However, relative to the project size, cost and duration, the deviations are probably much the same.

As mentioned earlier, the author's present organization had made little effort to establish formal planning and control procedures prior to his appointment to head the IS department. Indeed, one such proposal made approximately four years ago, met with harsh criticism and firm rejection from IS management. The practice, at that stage, was to propose a delivery date sometime in the future, without attempting to break the project down into components, to estimate resource requirements, or to establish measurable "milestones". No attempt was made to monitor progress other than to indicate how far the project had proceded along its path to the delivery date. This approach resulted in delays in implementation, excessive errors resulting in extensive maintenance requirements, and incomplete development, particularly testing and documentation. A new approach to estimating is required. More reliable techniques must be developed.

A number of studies of cost/resource requirements for software development are reported in the literature (20,34,35,36,37,43). Generally, some form of model is fitted to empirical data (34), although theoretically based models have also been developed (33). Morin (31), as related by Putnam (34), found that most of the cost estimating methods she studied involved relating system attributes and process environment factors to human effort, project duration

and development cost. These procedures usually involved multiple regression, using a large number of independent variables. She concluded that she had "failed to uncover an accurate and reliable method which (would allow) programming managers to solve easily the problem inherent in predicting programming resource requirements".

Many of the problems experienced in the management of software projects relate to the abstract nature of the end product, and to senior management's lack of understanding of software concepts. Improving the "visibility" of the product is therefore, essential to improving software control (see also Chapter 6).

5.1 THE SOFTWARE LIFE-CYCLE

The successful management of any project requires the systematic breakdown of that project into activities. The time and effort required in their execution, and the sequence of these activities, provides a resource requirement schedule for the project and a series of check points, or milestones, by which its progress may be monitored. For software, this sequence of activities is termed the "software life-cycle" (10,12,16,25), covering the life of the software from its inception to the stage at which it falls into disuse*.

FOOTNOTE:

* The software life-cycle concept is well known, with numerous authors (10,12,16,25) proposing a life-cycle breakdown suited to their specific organization and environment. Only a simplified breakdown is presented in FIGURE 5.1 for discussion of the life-cycle concept. However, the author has found this simplified breakdown useful in the introduction of the concept. Greater detail could be introduced as required in any specific organization and environment.

This "milestone approach" to project management should be applied to all projects, be they procurement, development or maintenance projects. Although a different activity breakdown may be performed for each project, standardising on a minimum breakdown saves both time and effort. In addition, standardising on a life-cycle breakdown permits the maintenance of development statistics (see section 5.4). This in turn enables performance comparison between projects, thus enabling more accurate and reliable scheduling of resource requirements and milestone achievements. For the present study, the following breakdown will suffice (FIGURE 5.1)

5.1.1 PROJECT DEFINITION

This involves the broad formulation of a problem or concept, the development or solution of which, is expected to be beneficial to the organization. The scope and object of the project should be clearly defined.

Formal approval should be required for the initiation of a project. Without this the author has experienced difficulty in controlling resource utilisation, identifying staff and user involvement, and obtaining user commitment. Project approval initiates the preparation of a Q.A. plan, identifying staff involvement and, thereby, establishing areas in which effort is required.

The importance of proper definition is not fully identified and stressed here.

5.1.2 REQUIREMENT SPECIFICATION

Project objectives are extended to include the broad functional characteristics of the system. Qualitative and quantitative characteristics of, and the relationship between, inputs and outputs are defined. Interaction with data bases, and with other systems are identified.

The purpose of this phase in the life-cycle is to establish benefits which the organization can expect in return for effort expended. This is achieved by means of a feasibility study involv-

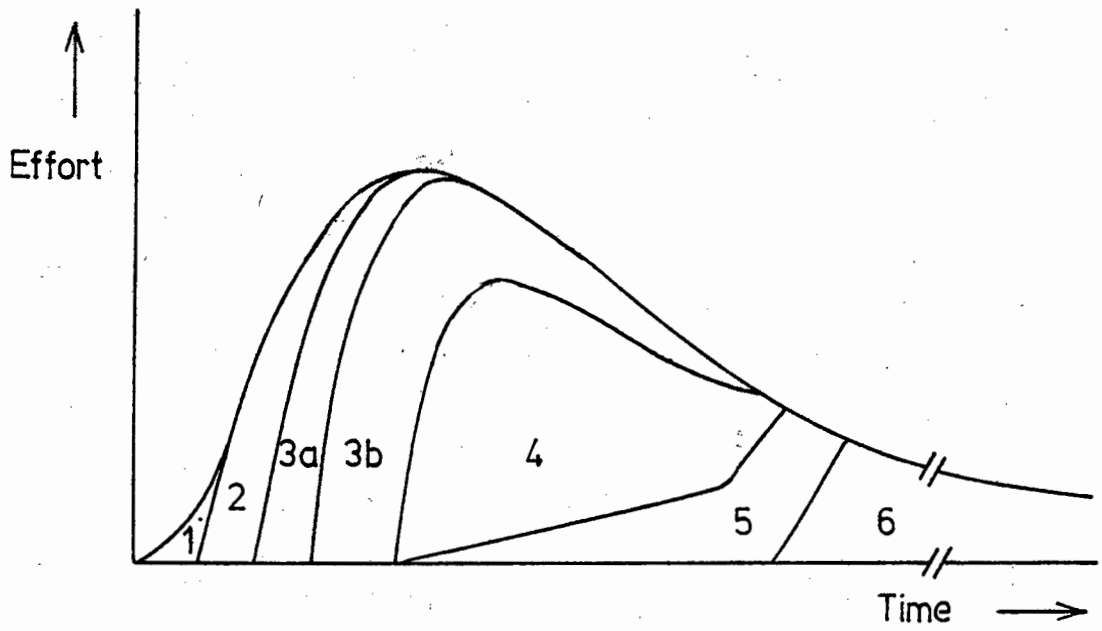


FIGURE 5.1 The Software Life-cycle

1 Project definition
2 Requirement specification
3a. Broad system design specification
3b Detail software design specification
4 System development and coding
5 System testing
6 Implementation and maintenance

ing the development of a life-cycle management plan matched to project contraints. Resource or facility utilisation requirements are then compared with expected benefits, usually in the form of a cost/benefit analysis report.

This is followed by a development recommendation. While this recommendation should be as complete as possible, since it should assist in the decision as to whether or not the project should be continued, it should not require undue detail or analysis, as this would represent wasted effort were the project to be abandoned later.

5.1.3 DESIGN SPECIFICATION

This phase in the software life-cycle involves establishing an optimum design to fulfill the functional requirements identified in the previous phase, bearing in mind the contraints and restrictions established during requirements specification. Design specification is generally conducted in two stages.

i) Broad system design

During system design specification the problem is analysed as a whole. Functional requirements are grouped logically, to minimize interaction between groups, or modules, i.e. interaction between modules is maintained only at the "highest level" possible. This enables continued development of the design specification for each module to be conducted independently of other modules, since only high-level interaction between modules will effect their further development.

ii) Detailed software design

The detail design stage refines the broad modular specification discussed above. The output from this stage is a specification for each segment or program from which program coding may easily be performed. Detailed input/output procedures, data and report formats, and decision rules are included.

5.1.4 DEVELOPMENT

The development phase consists of transcribing the detailed design (5.1.3(ii)above) into program code. Each segment should be individually tested (off-line) to ensure its conformity to the detailed specification. Thereafter, programs and system modules should be built from segments, and the interfaces between these segments tested. The aim of this phase in the life-cycle is to ensure that all program modules conform to the detail design specification.

5.1.5 SYSTEM TESTING

Following the development and off-line testing of all modules the system itself is constructed. A detailed test is then conducted on the system to ensure its conformity with the system design specification established during phase 3 of its life-cycle. In this manner all specified outputs are verified, and interfaces between the system modules identified.

5.1.6 IMPLEMENTATION AND MAINTENANCE

Once off-line testing has been conducted the software enters the "commissioning period". During this phase the system's performance is monitored on-line, i.e. under user operation, and any flaws marring its performance corrected. At the end of this stage the user is required to certify that all his requirements have been met.

5.2 MODELLING THE DEVELOPMENT PROCESS

The author has found two basic approaches to estimating project duration, cost and resource requirements. The first relates to the standard industrial practice of applying a measured productivity rate, for workers, to an estimate of the total job (usually in lines of code, sometimes adjusted by some "difficulty factor"), to obtain the total resource requirement for the project. This is then divided by the available manpower, to

obtain the project duration. Alternatively the allowed time for project completion (specified by the customer) is divided by the total resource requirements to obtain the manpower level required. This approach ignores the time dependent character of software development projects. It assumes that any "slippage" in the development schedule can be "made-up" by increasing the available manpower, i.e. it assumes that the total manpower required for the completion of the project remains constant.

A second approach to understanding the software development process involves relating empirical data on the size of a project, its duration, and the manpower loading of the project, to some empirically derived "software equation". Using extensive empirical data, Norden (32) established that software projects follow a life-cycle pattern, similar to that described in section 5.1 above. He found that when the various phases are laid-out in their correct time sequence, and the manpower loading for each phase aggregated, a project-load curve was generated (FIGURE 5.1) In addition, he observed that many of these curves followed the same pattern, rising to a peak and then tailing off exponentially in time. Since the load followed a Rayleigh-type distribution in time, the load curve has become known as the Norden-Rayleigh curve.

Putnam (34) has related the basic project control parameters, viz. development time (t_d), total effort required (K) and the level of technology or skill available (p(t)) to this curve. In developing this relationship he assumes that the total effort applied to a project is dependent on the number of problems to be solved, and the "skill" or "capability" of the software team solving them. With y(t) denoting the proportion of problems solved at time (t), the rate of progress is then given by

$$d/dt\ y(t) = p(t).K.(1-y) \qquad (5.1)$$

Integration gives

$$y(t) = K.(1 - \exp(-\int^{t} p(t)dt)) \qquad (5.2$$

Assuming a "linear learning curve"

$$p(t) = a.t$$

where $a = 1/2t_d^2$ with t_d the time to reach peak effort. It is found empirically that t_d corresponds closely to the development time.

The rate of progress on a project is given by

$$d/dt\ y(t) = (K/t_d^2).t.\exp(-½(t/t_d)^2) \qquad (5.3)$$

This represents the Norden-Rayleigh curve for software development, where

K is the total work required in the project

t_d is the development time

K/t_d represents time rate of doing work or manpower loading. In this context K/t_d^2 can be considered as representing the level of difficulty of the project.

5.3 SIZE / TIME / EFFORT TRADE-OFF LIMITATIONS

Management can influence the rate of progress made on a project by allocating effort (resources). Normally it would be assumed that increasing the effort would decrease the duration. However, for software projects this is not always the case. Often, allocation of additional resources to an overdue project only delays that project further (6). An improved understanding of the relationship between effort and completion time is thus required (34,35,36,37).

Action taken in allocating effort (K) or specifying completion dates (t_d) influence the difficulty of the project (K/t_d^2). During the development phase this difficulty can be represented as follows;

$$K/t_d^2 = (d/dt\ y)/(t.\exp(-½(t/t_d)^2) \qquad (5.4)$$

At $t=t_d$ this is;

$$K/t_d^2 = (d/dt\ y)_{max}/t_d \qquad (5.5)$$

Putnam argues that if the slope of y, being the applied effort or activity, is too great in the development phase, management will be unable to effect progress on the project in an efficient manner. The argument advanced is that many activities in the development of software are sequential by nature, i.e. activities must follow in a logical sequence. The maximum effort which can be effectively applied at any stage in the project life-cycle depends on the number of activity streams in the project plan at that particular stage. There is a minimum time within which an activity can be completed, and therefore, a minimum time within which a system can be developed. This minimum (t_d) depends on the level of complexity of the tasks or activities, represented by the difficulty factor, (K/t_d^2), with which the software team are able to cope at the time. Clearly this difficulty is dependent on the relative level of technology available to the team, and on the skills of the team members. The implication is that "the only way to effect the minimum possible development time, given a system of a certain size and complexity, is by improving the development environment" (20). According to Golden et al (20) this is achieved through the introduction of new methodologies and software tools, and not through the addition of people.

Golden et al (20) have tested this model against four systems developed at Xerox. Their results, shown in Table 5.1, reveal a close correlation between the actual effort and time taken, and the predictions of the Putnam model. Manloading for the duration of one project showed reasonable agreement with the model, FIGURE 5.2

5.4 SIZING OF THE SOFTWARE SYSTEM

Given the size of a software system and the development time allowed, the manpower effort required, as a function of time, can be determined from equation 5.2. The problem, then, is the estima-

	Development Time (Months)		Effort (Man Months)		
	Minimum Time Estimate	Actual	Estimate For Minimum Time	Actual	Calc. Value For Actual Dev. Time
A	12.0	11.0	124	167	176
B	12.7	13	83	82	76
C	12.8	15	304	150	161
D	10.2	11.7	70	51	41

TABLE 5.1 Comparison of Resource Estimates - Model Calculations vs. Xerox Actual (20)

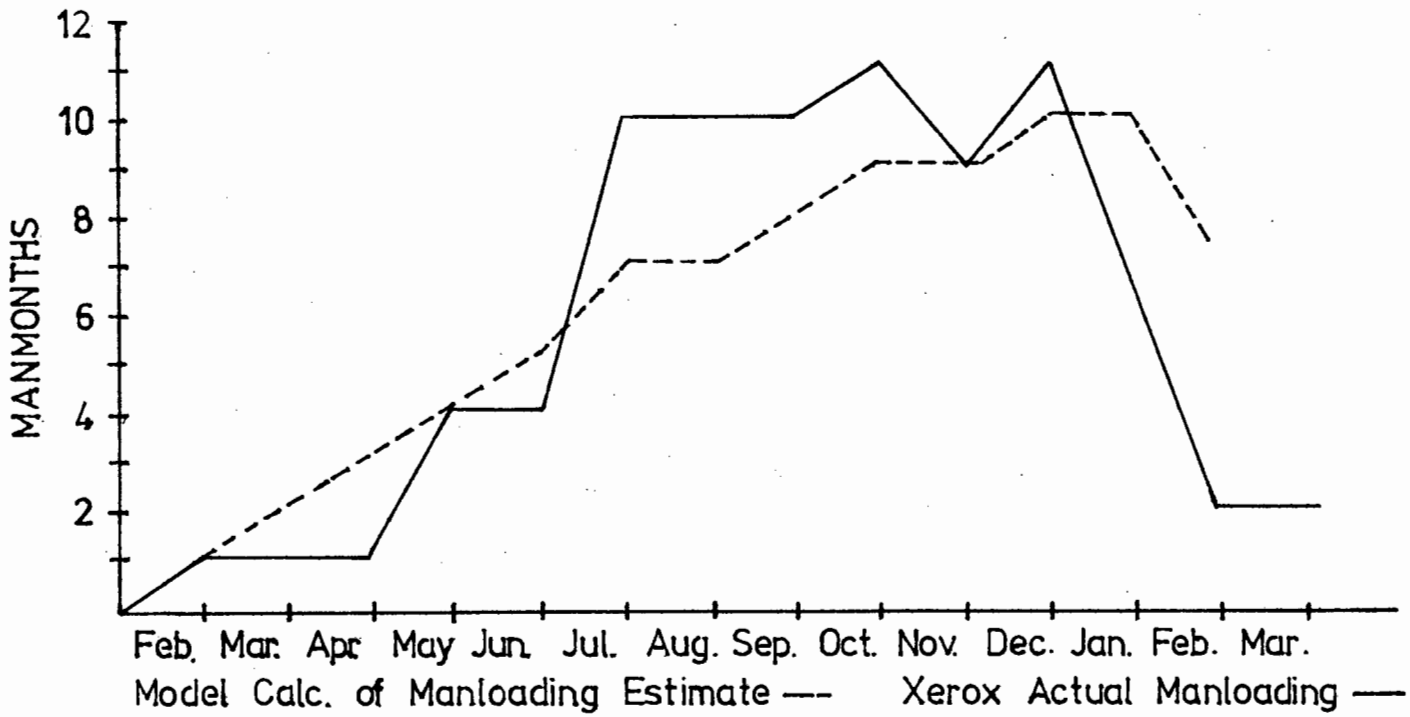


FIGURE 5.2 Manloading - Estimations, Actual (20)

tion of the size of the system, i.e. the number of lines of code.

No reasonably reliable, or proven technique for estimating the size of a system has been found in the literature. It would appear that design and coding techniques are too diverse at present. Without established standards for software design and coding, no meaningful statistics can be gathered for the sizing of particular categories of software.

Golden et al (20) suggest that the development of structured analysis and design, and its establishment as a standard for software design and coding, might simplify the problem, and allow more accurate estimates. From a detailed and accurate specification of the systems requirements, a basic design structure could be developed. Educated guesses, based on previous experience and statistics, could then be made of the size and complexity of each module. These "guesses" could be improved, as detail is added to the design.

Essentially this is the approach adopted by Putnam and Fitzsimmons (35,36,37). On completion of the various life-cycle stages of the project viz. project definition, requirements specification, design, etc. estimates are made of

i) the most likely size of the system (m)

ii) the "smallest possible" size of the system (a)

iii) the "largest possible" size of the system (b)

The expected value is then found from the equation*

$$E_i = (a+4m+b)/6$$

FOOTNOTE:

* Putnam and Fitzsimmons noted that estimates of size were generally skewed on the high side, a bias typical of a beta distribution. The characteristics of a beta distribution are used in PERT estimating, a technique commonly used for project control. The equation for E_i used here is an estimate of the expected value of a beta distribution.

and the standard deviation from*

$$S_i = |b-a|/6$$

As the system is broken down into subsystems, i.e. as detail is added to the design, these estimates are revised. Putnam et al (35,36,37) found that the expected value for the size of the system, remained within one standard deviation of the previous estimate, while the standard deviation itself steadily declined. As the project progressed, the uncertainty ratio of the size estimate, S_i/E_i, decreased. The results of this approach for a hypothetical project are illustrated in Table 5.2

5.5 SUMMARY.

Standardising on a "software life-cycle", and identification of the activities to be performed and products to be produced at each phase provide "visibility" of the software product and improve understanding, by senior management, of the software development process. The following minimum breakdown of life-cycle activities is suggested;

i) Project definition
ii) requirement specification
iii) design specification, including,
   - broad system design
   - detailed software design
iv) development
v) system testing
vi) implementation and maintenance

FOOTNOTE:

* An estimate of the standard deviation of any distribution (including a beta distribution) may be found by dividing the range within which 99% of the values are likely to occur by six.

	Function	a Smallest	m Most likely	b Largest	E_i Expected	S_i Std Dev	S_i / E_i Uncertainty Rate
First Sizing (2 weeks into system definition)	Total System	50000	-	140000	(a+b)/2 =95000	|b-a|/6 =15000	15,8%
					(a+4m+b)/6	|b-a|/6	
Second sizing (start of functional Specification)	File Handlers	25000	40000	70000	42500	7500	
	Utilities	5000	15000	26000	15167	3500	
	Systems Procs	12000	36000	50000	34333	6332	
					92000	10422	11,3%
					(a+4m+b)/6	|b-a|/6	
Third Sizing (halfway through functional spec. At final commitment)	Maintained	8675	13375	18625	13467	1658	
	Search	5577	8988	13125	9109	1258	
	Route	3160	3892	8800	4588	940	
	Status	850	1425	2925	1579	346	
	Browse	1875	4052	8250	4389	1063	
	Print	1437	2455	6125	2897	781	
	User Aids	6875	10625	16250	10938	1563	
	Incoming Msg	5830	8962	17750	9905	1987	
	Sys Mon	9375	14625	28000	15979	3104	
	Sys Mgt	13700	13700	36250	16225	4992	
	Comm Proc	5875	8975	14625	9400	1458	
					98475	7081	7,2%

TABLE 5.2 : Hypothetical Project Sizing (source code statements) following the Putnam and Fitzsimmons approach (35)

A "milestone" approach to the control of software projects can then be adopted. Duration and resource requirements should be estimated for each phase of the life-cycle, and progress monitored against the resulting plan. These estimates obviously depend on the estimated "size" of the system. It is suggested that initially an "educated guess" be made based on previous experience and available statistics. This guess to be improved as the project progresses and detail is added to the design. It has been found that size estimates remain within one standard deviation of the previous estimate while the standard deviation itself, an indication of the uncertainty in the estimate, steadily decreases.

A sophisticated model of the development process has been developed, allowing insight into the relationship between project control parameters, viz., project duration and manpower availability. For software projects, rate of progress can often not be improved by simply allocating additional manpower since the maximum rate of progress also depends on the relative complexity of the project, the level of technology available to the team and the skill of the team members.

CHAPTER 6

QUALITY CONTROL - SOFTWARE CONFIGURATION MANAGEMENT

6.1 DEFINITION AND PURPOSE

Software configuration management (SCM) may be defined as "the discipline of identifying the configuration of a system at discrete points in time" (4). Its primary purpose is the effective control of the system's life-cycle, and of the product's evolving configuration. This is achieved by methodically labelling, or identifying, the system's components, and by defining the manner in which these components interact to perform the system, or product, functions. Changes to the configuration and the maintenance of the integrity *, and traceability of this configuration throughout the system life-cycle must be closely controlled (4). However, the abstract nature of software complicates this controlling process, as no physical or tangible output can be viewed or inspected (refer section 2.3.1). In addition, delays in the development schedule, malfunctioning, incomplete satisfaction of requirements, excessive utilisation of resources or expenditure in excess of budgetted allowances can seldom be anticipated in time to allow effective corrective action.

Software configuration management is, therefore, intended to provide the "visibility and substance" (38) required to allow effective monitoring of progress, traceability of the history of the system's development and continued assurance of the integrity of the system.

FOOTNOTE :

* Integrity, in this contect, means adherence to, and satisfaction of the systems requirement.

6.2 THE CONCEPT OF A "BASELINE"

Fundamental to the management of software configuration is the concept of a "baseline", (4,25e), a reference point in the system's life-cycle at which its description or definition is approved, its integrity or compliance with requirements having been reviewed or tested.

In general, a "baseline" is formally defined at the end of a phase in the software life-cycle, as illustrated in FIGURE 6.1*. The baseline document describes the output from that phase. Minimum documentation requirements for defining such baselines are included in FIGURE 6.1.

i) The requirements baseline document should provide a functional definition, i.e. a broad description of what the system is required to do.

ii) The functional allocation baseline document is a broad design specification. It provides a broad outline of the major elements of the system, allocating specific functions to each, and defining the interaction, or sequence of operation, of these elements.

iii) The design baseline is a detailed design specification for each of the elements of the system. Each of the major elements described in the functional allocation baseline is broken down to its primary entities. Data records, input/output facilities, data manipulation and

---

FOOTNOTE:

* Additional "baselines" may be included, e.g. as for the design phase. The number of configuration "baselines" is established by comparing the control and "visibility" gained with the time and cost expended to conduct the necessary reviews and generate the baseline documents.

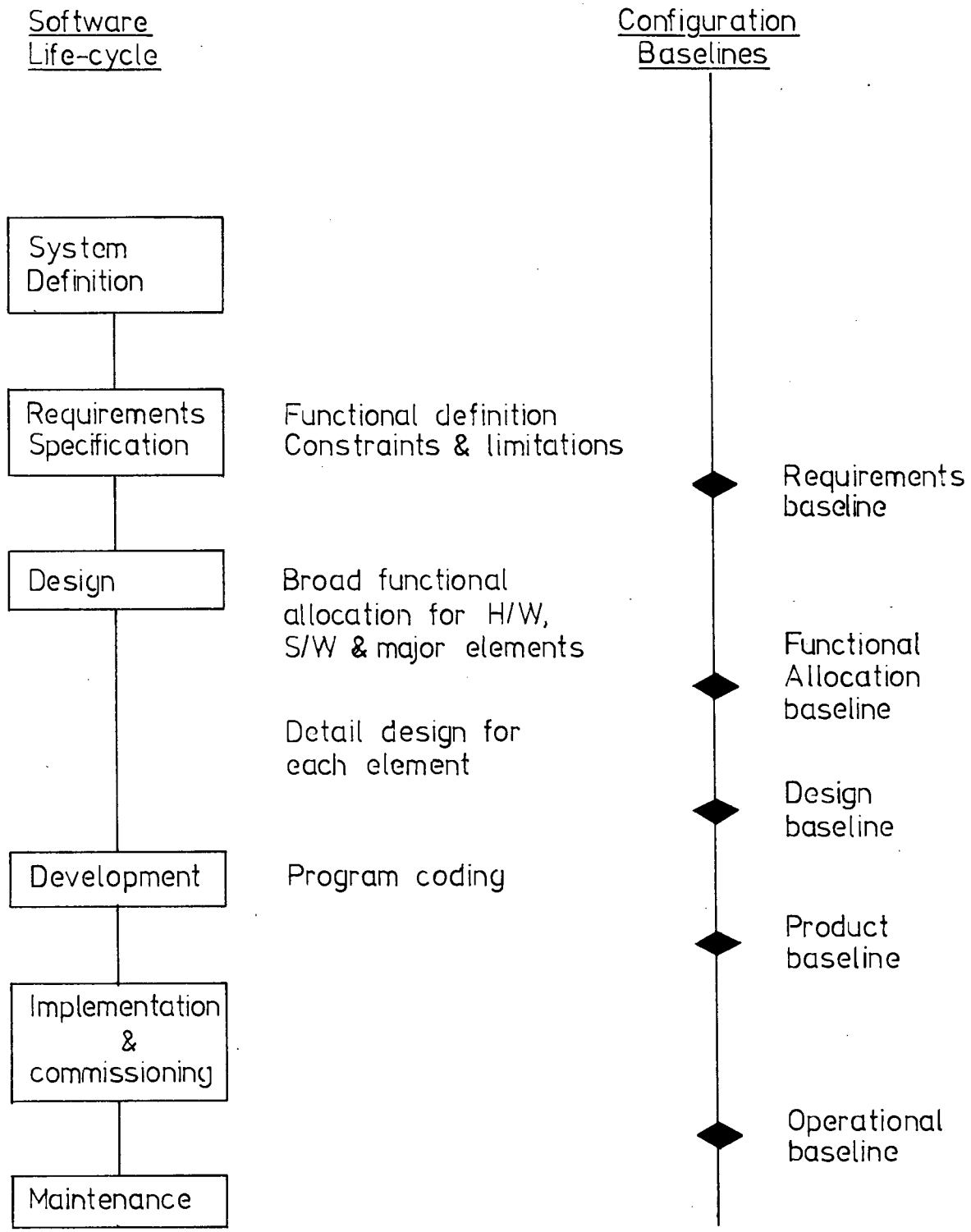


FIGURE 6.1 Configuration baselines related to software life-cycle phases. (cf (4, 25)

analysis conducted on each of these entities and their iteration, or operational sequence, are described in detail.

iv) The product baseline contains source code for each of the design/program entities identified in the design baseline.

v) The operational baseline is identical to the product baseline except for modifications made to basic entities during the implementation and commissioning phase of the life-cycle.

6.3 ELEMENTS OF CONFIGURATION MANAGEMENT

Bersoff et al (4) identify four elements of software configuration management.

i) Identification of the system configuration
ii) Change control
iii) Configuration status accounting
iv) Software configuration auditing

6.3.1 IDENTIFICATION OF THE SYSTEM CONFIGURATION

Identification of the system configuration involves careful definition of its component entities and of the manner in which these entities interact. A system baseline defines the configuration at a particular point in time. At inception the baseline comprises software in its most recent state. At any particular time the system configuration, then, incorporates the baseline, as defined previously, and any approved changes to the software (see section 6.3.2).

The role of identification in the software configuration management process is to provide "labels" for the component entities, and, where necessary, to specify their interaction. Bersoff et al (4) identify the most elementary entity in the

software configuration as the Software Configuration Item (SCI) A software baseline may then be established as a set of SCI's related to one another in the hierarchical fashion illustrated in FIGURE 6.2*. Entities still under development are termed Design Objects (DO), and do not fall under configuration management or change control.

6.3.2 CHANGE CONTROL

A formal procedure for incorporating changes into baselines is required for controlling the evolution of software, and for maintaining the integrity of the configuration. A formal change plan (CP) is required to define proposed alterations to the software system. Bersoff et al (4) identify five events which might initiate such a proposal.

i) A software deficiency in an existing baseline.

ii) Hardware problems, such as interfacing between hardware sub-systems, may sometimes require solutions through software modification.

iii) New operational requirements, often resulting from the development of a system with which the system under consideration is required to interface.

iv) Economic savings, such as budget reduction, sometimes require software modifications. Means for simplifying the development of the system might reduce the overall cost, while requiring modification to the software design.

v) Schedule change, possibly initiated by a change in development priorities.

FOOTNOTE:

* This tree-like hierarchy may be likened to that depicting hardware configuration, where an assembly break-down gives sub-assemblies, and further breakdown of the sub-assemblies gives single components.

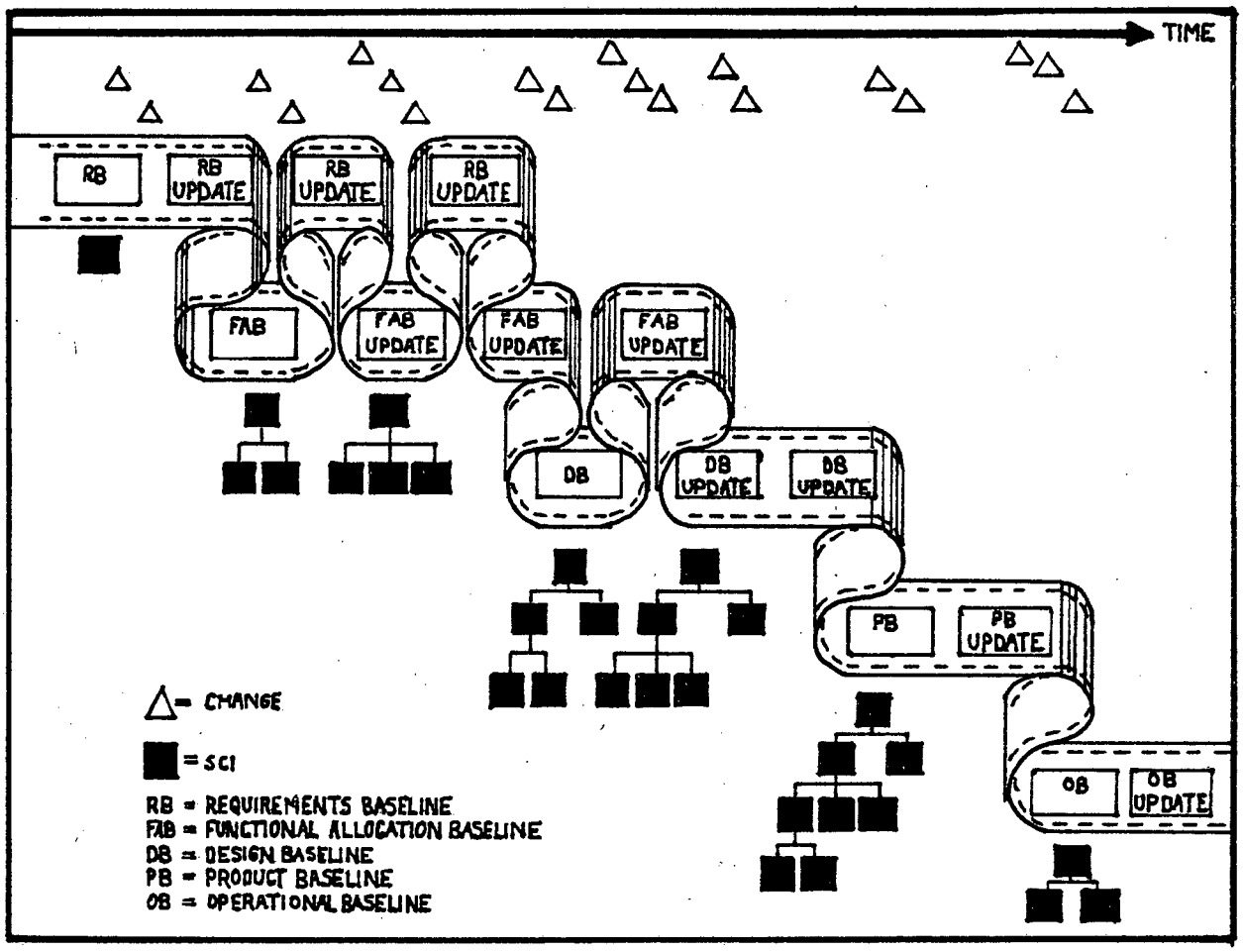


FIGURE 6.2 System Configuration Identification (4)

These change-proposals should be evaluated in terms of their impact on the total system's facilities and the development schedule. Changes proposed at any stage in the life-cycle should be fed back to the previous stages. In this manner their impact on other developing activities may be ascertained prior to formal approval or disapproval by a Configuration Control Committee (C.C.C.). This feedback process is intended to promote consistency in all development activities and to allow support documentation to be updated.

The question arises as to when a software entity should come under formal change control. Forrester (17) proposes that all documents should be subject to change control procedures once they have been formally approved, since it is at this point that the document becomes the basis for further development work. For programs, or executable software, it is important that formal controls not be imposed too early, as these might stifle the creativity of the programmer. However, if imposed too late confusion may result from programmers working to unspecified requirements. Forrester, therefore, suggests that control be imposed when a program is used by anyone other than the original programmer.

From this discussion it would appear that formal control should be imposed on all software before it passes from the original author to become the basis of further development.

6.3.3 CONFIGURATION STATUS ACCOUNTING

Software configuration status accounting is the mechanism for recording the evolution of the system and relating this to the documented baselines. For large software systems this recording, involving large data output and input, could be automated. However, for small systems a simple hand-file would be adequate, with the following being recorded in the minutes of the Configuration Control Committee (CCC) meeting (4).

i) The date of establishment of a baseline or update
ii) The date of approval of each SCI

iii) A description of each SCI

iv) CP status (approved, disapproved, pending)

v) CP description

vi) Change status

vii) Description of each change

viii) Deficiencies in a to-be-established baseline

For large systems these dates would be input to a computer and reports giving the status of entities generated.

6.3.4 SOFTWARE CONFIGURATION AUDITING

Software configuration auditing serves two basic purposes, (i) configuration verification, and (ii) configuration validation (4). Definitions of these activities vary in detail (4,25f). However, there seems to be general consensus that verification deals with the integrity and basic content of the configuration, while validation applies to the functioning of the products defined by the configuration. The author favours the following definitions (25):

i) Configuration verification: ensures that the content of each software configuration item specified at one baseline or update is satisfactorily echoed at the immediately succeeding baseline or update.

ii) Configuration Validation: ensures that the functioning and features provided by an end product in the configuration corresponds adequately with the requirements specified at that level of configuration.

The configuration auditing function is well illustrated by the "System Development Model" of Deutch (Jensen and Tonies (25g)) FIGURE 6.3. The development process is viewed as a symmetrical tree-like structure. The diverging tree establishes the content and requirements of the system in ever increasing detail. Each item in the diverging tree has a corresponding software item in the converging tree. The verification-validation process ensures

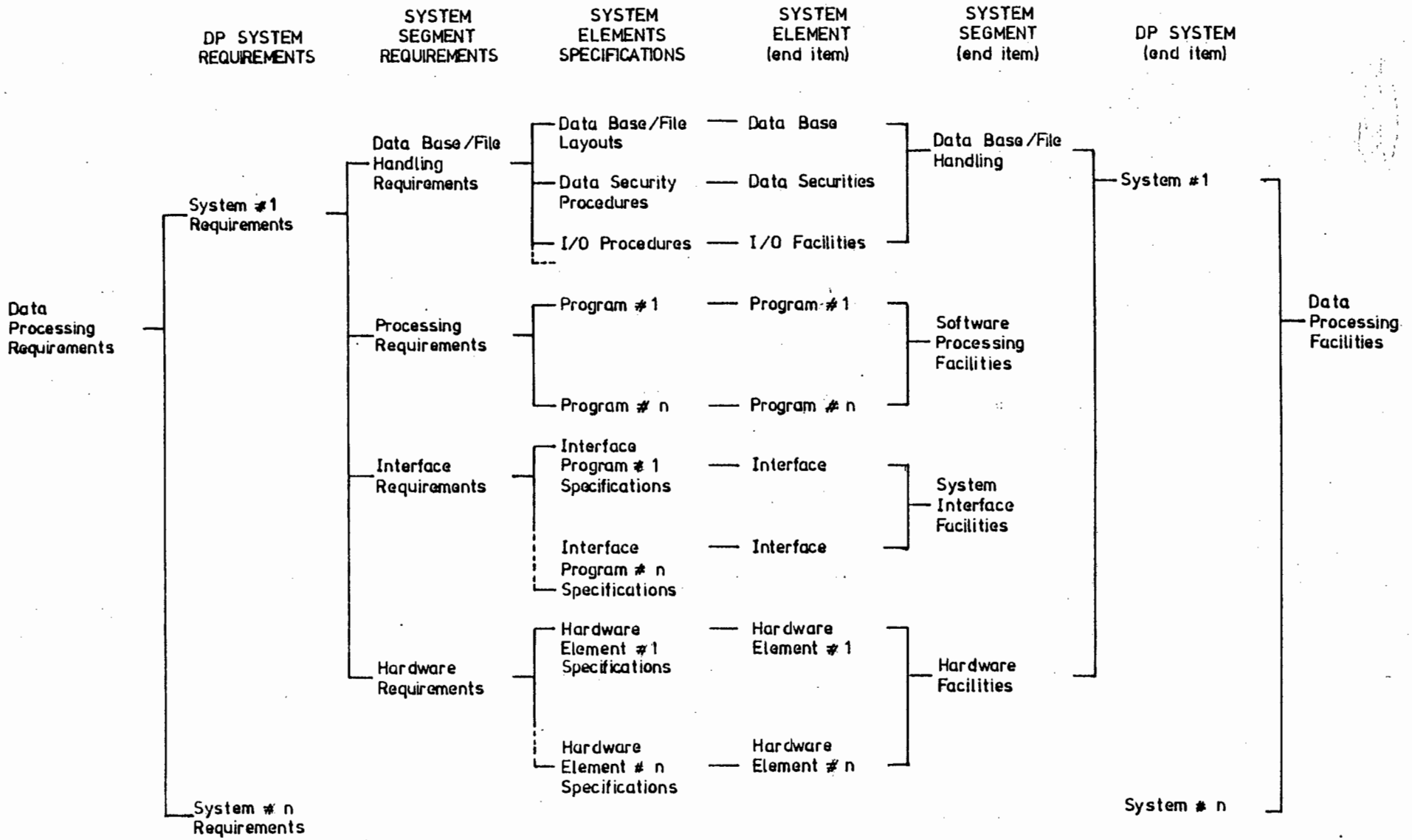


FIGURE 6.3 Systems Development Model (cf.(25g))

the integrity of the converging tree (or product) with respect to the diverging tree (or product definition). It further ensures the logical development of detail in the product definition.

Configuration auditing should be applied to each baseline, prior to its establishment. The prime focus of the auditing function for each baseline is as follows (4):

i) Requirements baseline; auditing establishes correspondence between the system concept and its application in the user environment.

ii) Functional allocation baseline; auditing ensures that software functions, corresponding with systems features identified in the requirements baseline, have been defined, and adequately developed into implementable models.

iii) Design baseline; auditing ensures that models and algorithms have been detailed sufficiently to ensure unambiguous software coding.

iv) Product baseline; auditing ensures that the performance of the product in the test environment, adequately satisfies the defined requirement.

v) Operational baseline; auditing ensures adequate performance in the working environment. It ensures that all specified requirements have been adequately satisfied.

6.4 SUMMARY.

The prime purpose of Software Configuration Management is to maintain a system's "integrity", that is, to ensure that the output from the various phases in the system's life-cycle correspond with the requirements identified in the previous phases. Configuration is defined in terms of a system's component entities. The initial configuration at any phase in the life-cycle is identified as the "baseline", and describes the output from that life-cycle phase. Before acceptance, and approval, this output is compared with the requirements identified

in the previous life-cycle phase. Future changes to this configuration must correspond with the requirements of the previous phase. The system configuration for each phase of the life-cycle is then defined in terms of a "baseline configuration" and any approved changes.

With this in mind, configuration management involves four activities;

i) identification of the system configuration in terms of its component entities

ii) change control to incorporate changes to the configuration baseline

iii) configuration status accounting to record the status of each addition or change

iv) software configuration auditing to verify the integrity of the system configuration and validate the functioning of the end product.

CHAPTER 7

THE HUMAN FACTOR

Providing facilities to control IS projects, as discussed in the preceding chapters, satisfies only part of the requirements of a successful management methodology. The major resource in IS development is human, and this "human factor" is key to the success of any project. Staff availability, capability, motivation and discipline can be deciding factors in the success of any IS installation. Consequently, development of "human resources" must form an intimate part of a successful management methodology.

Three factors relating to personality characteristics provide insight into the behaviour of IS personnel, and a guide to the development of a working environment which will realise efficient and effective discharge of responsibilities, high quality output and low staff turnover. This in turn will contribute to the successful management of the IS function.

i) distinguishing characteristics defining the individual's personal "needs" and expectations from his work environment.

ii) criteria against which the individual will value the reward given for efforts exerted on behalf of his employer.

iii) individual perceptions of the work environment and the degree to which it satisfies personal "needs".

7.1 PERSONALITY CHARACTERISTICS OF SYSTEMS PERSONNEL

Many definitions of personality can be found in the literature. In this study, personality will be defined as "the organization of the psychological systems in each individual that determines the interaction of that individual with the environment" (47). Several studies have investigated the personality characteristics of IS personnel in an attempt to understand the interaction of these

people with their job environment (1,2,11,46,47). The prime goals of these studies have been to determine those aspects of the work environment which induce satisfaction or dissatisfaction; those aspects motivating improved performance, and those influencing staff turnover.

Murray has described behaviour in terms of an individuals "needs" and perceived "press" (47)*. He defined a "need" as a "motive force within the individual", having "properties of both potential direction and strength". This force is aroused when an individual finds the environment or situation unsatisfying. His efforts are then directed towards creating a more satisfying situation. "Press" refers to the level to which this motive force is aroused by the environment or situation. Behaviour may then be regarded as resulting from the interaction of an individuals "needs" with his perception of the "press" exerted by the environment.

With this understanding, a study of work behaviour should be two fold. Firstly, individual "needs" should be defined and quantified, to provide a profile of distinguishing characteristics by which personalities can be compared. These "needs" then influence the relative importance of criteria against which the individual will value the reward given for efforts exerted on behalf of his employer, and the level of satisfaction or dissatisfaction derived from this reward. Secondly, an individuals perception of his work environment can be related to this needs profile to provide some explanation for his behaviour.

Several characteristics have been identified which distinguish IS personnel from other professionals (1,2,11,47). In compiling a profile of distinguishing characteristics for IS personnel, the

---

FOOTNOTE

* this description follows that presented in pioneer work of Henry A. Murray and his associates at Harvard University, as reported by Woodruff (47).

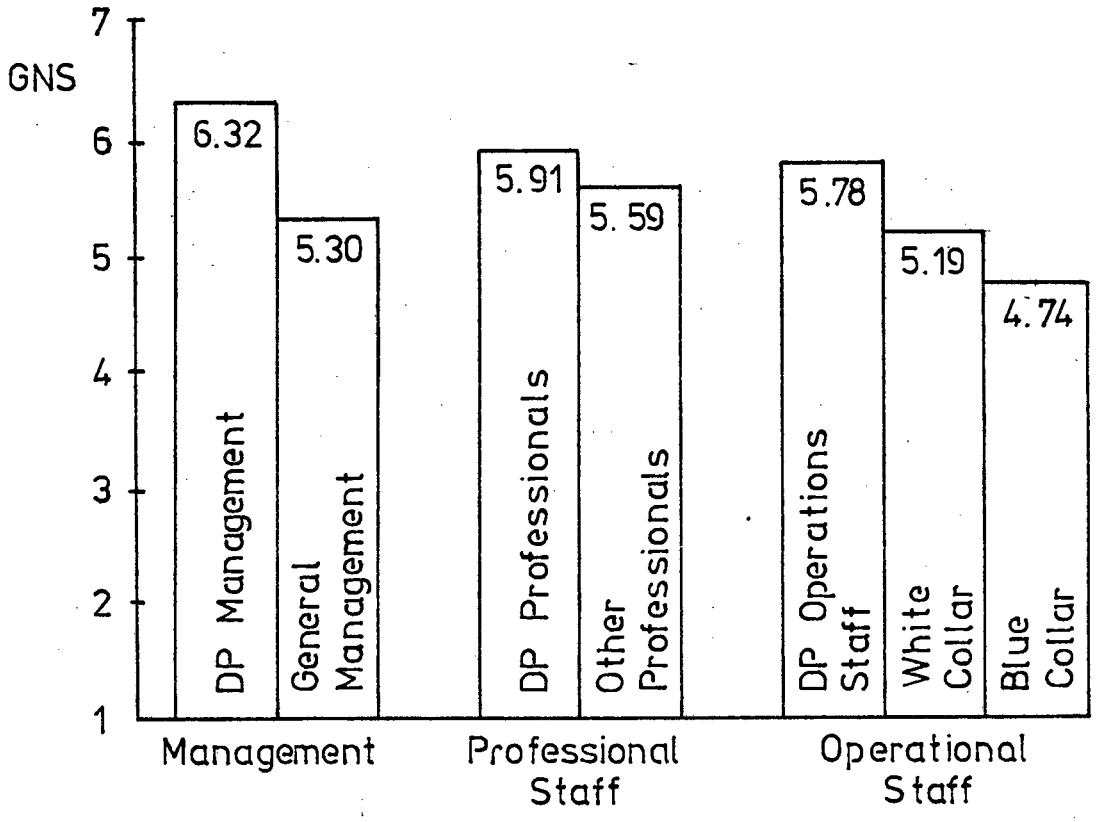
author will discuss only those areas in which IS personnel differ markedly from personnel in other professions. As such, the profile is incomplete. However, a start is made in an area which still requires extensive research if it is to be of significant value in the structuring of the IS organization.

7.1.1 NEED FOR GROWTH OPPORTUNITIES

"Growth Need Strength"(GNS) refers to an individual's need for personal growth and development (42,11b). This is generally associated with a need for personal accomplishment, for learning and being stimulated or challenged (11b). Couger and Zawacki (11) found IS personnel to be characterised by a high GNS when compared with personnel in other job categories (FIGURE 7.1). In particular DP managers exhibit a considerably higher GNS than general mangers; DP professionals - systems analysts, analyst programmers and programmers - show higher GNS than professionals in other areas; and DP operations personnel show higher GNS than either white or blue collar employees. From this it would seem that the "dynamic characteristics of the DP industry appeal to people who have a desire for challenge and growth opportunities" (11c)

7.1.2 NEED FOR ACHIEVEMENT

The high need for achievement, identified by Woodruff (47), is closely related to GNS. IS personnel were found to aspire to accomplishing difficult tasks, having an above average inclination towards challenge. In addition, they were found to have an above average level of endurance, being willing to work long hours and persevere with difficult problems. Bartol et al (2) emphasize that IS personnel are likely to work well when set challenging, but realistic goals. However, they are likely to become overly frustrated when continually set impossible work loads and deadlines (see also 7.1.3. below).



(Ratings on a scale of 7, where 1 is low and 7 high)

FIGURE 7.1 Comparison of GNS for Various Job Catagories (11)

7.1.3 NEED FOR ORDER

Woodruff (47) found that IS personnel desire a highly ordered existence, detesting confusion or disorganization. This would account, to some extent, for the frustration they experience when continually set impossible work loads and deadlines (refer 7.1.2 above).

Their need for order is emphasised by their dislike for ambiguity or uncertainty in information. Woodruff's (47) research also revealed that IS personnel have a high need for cognitive structure in their working environment and a desire to make decisions based upon definite knowledge, rather than upon guesses or probabilities. This latter desire is also evident in their need for feedback. Feedback and recognition have been identified as a major deficiencies in the management of IS personnel (11) (see also 7.3 below)

7.1.4 NEED FOR SOCIAL INTERACTION

It is generally agreed that IS personnel have a low need for social interaction (11,47). Bartol et al (2), however, suggest that this conclusion should be viewed with caution. They indicate that research measures used by Cougar et al (11) in arriving at this conclusion were limited, since no differentiation was made between the various persons with whom IS staff might come into contact, i.e. work associates, personnel from other professions, users, management, etc. One indication that this notion concerning the low need for social interaction amongst IS personnel might be inaccurate is their need for feedback and recognition by superiors. As mentioned in 7.1.3 above, both these factors have been found to be major deficiencies in the management of IS personnel.

Cougar et al (11d) also found that IS personnel exhibit a greater loyalty to their professions than to their employer organization. They tend to place more importance on the opinion of other IS professionals, than on those of colleagues in the same

organization*. This suggests a need for social recognition. However, as discussed in chapter 2, few non-system personnel understand the content of IS work, and therefore, it is to be expected that recognition be sought from within the profession, rather than from peers in the organization.

In the author's opinion this "low" need for social interaction and recognition is sometimes confused with the desire amongst systems personnel to "work alone, i.e. in a place where they (are unlikely to) be disturbed by other people"(11e). By its nature, IS work, particularly systems design and development, often requires absolute concentration and creativity, and the desire to be left undisturbed should not be confused with a low requirement for social recognition.

7.2 ASPECTS OF THE WORKING ENVIRONMENT EFFECTING WORKER BEHAVIOUR

Studies of worker behaviour have concentrated on absenteeism and turnover, and on worker performance (1,2,11,46,47). The prime goal of this research has been to identify aspects of the work environment affecting worker behaviour, so that the system of values applied by the individual in assessing his work environment may be understood. Several "models" or "theories" of worker behaviour have been proposed. Those applicable to this discussion are described, briefly, below (11,22,23,42).

7.2.1 TWO FACTOR THEORY

Herzberg (23) proposed a two factor theory of job satisfaction. He suggested that aspects of the work environment producing satisfaction (identified as motivating factors) were separate and

---

FOOTNOTE

* This conclusion has been disputed by Bartol et al (2) and will be discussed further in section 7.3

distinct from those leading to job dissatisfaction (identified as hygiene factors). Motivating factors included achievement, recognition, the work itself, responsibility, advancement and growth, while hygiene factors were generally extrinsic, or non-job-related, including company policy, supervisory style, co-worker relations, salary, working conditions, status and security(42).

Herzberg's research offers some guidance in the development of a job enrichment program intended to improve worker motivation. Improving those aspects of a job affecting hygiene factors will simply reduce the level of dissatisfaction of the workforce. It will not induce a lasting improvement in worker motivation and therefore cannot improve productivity to any marked degree. Improvement in motivation and job satisfaction can, however, be achieved through allowing "greater scope for personal achievment and recognition, more challenging and responsible work, and increased opportunities for advancement and growth"(42).

7.2.2 ACHIEVEMENT MOTIVATION THEORY

The underlying assumption in this theory is that individuals are characterised by varying levels of "need for achievement" (n-Ach). n-Ach represents a need to accomplish something important. High n-Ach represents a tendency to seek challenging tasks and to assume personal responsibility for their accomplishment, and a preference for situations which provide clear feedback on performance. On the other hand, low n-Ach represents a preference for low risk activity and for sharing of responsibility for task accomplishment with others (42).

Several studies support the theory's prediction that "enriching a job by providing more responsibility, challenge, and feedback (will) lead to increased performance, involvement, and satisfaction for a high need achiever", however, "enriching the job of a low need achiever would at best have no impact on performance and could lead to increased frustration, anxiety, and job dissatisfaction"(42).

7.2.3 JOB CHARACTERISTICS MODEL

The "Job Characteristics Model" of work behaviour is illustrated in FIGURE 7.2 (11f,22,42). "Critical psychological states" which lead to high levels of internal motivation, satisfaction and quality performance are identified. These states relate to;

i) Experienced meaningfulness referring to the individual's perception of his work as worthwhile or important by some system of values he accepts.

ii) Experienced responsibility referring to his believing that he personally is accountable for the outcome of his efforts.

iii) Knowledge of results referring to his being able to determine, on a regular basis, whether or not the outcome of his work is satisfactory.

These states are influenced by the individual's perception of his job. Five job characteristics (core job dimensions) are considered to exert the major influence on the "critical psychological states" (FIGURE 7.2). These are;

i) Skill variety is the degree to which the job presents a challenge to the individual's skill and ability.

ii) Task identity is the degree to which the job requires completion of a whole and identifiable piece of work - doing a job from beginning to end with visible outcome.

iii) Task significance is the degree to which the work impacts the lives and work of other people.

iv) Autonomy is the degree to which the individual experiences freedom and independence in exercising discretion in scheduling his work and in determining how this work is to be done.

v) Feedback from the job itself refers to the degree to which the worker, in performing the set task, is informed of the effectiveness of his efforts.

As indicated in FIGURE 7.2, "skill variety", "task identity" and "task significance" influence the "meaningfulness" which the

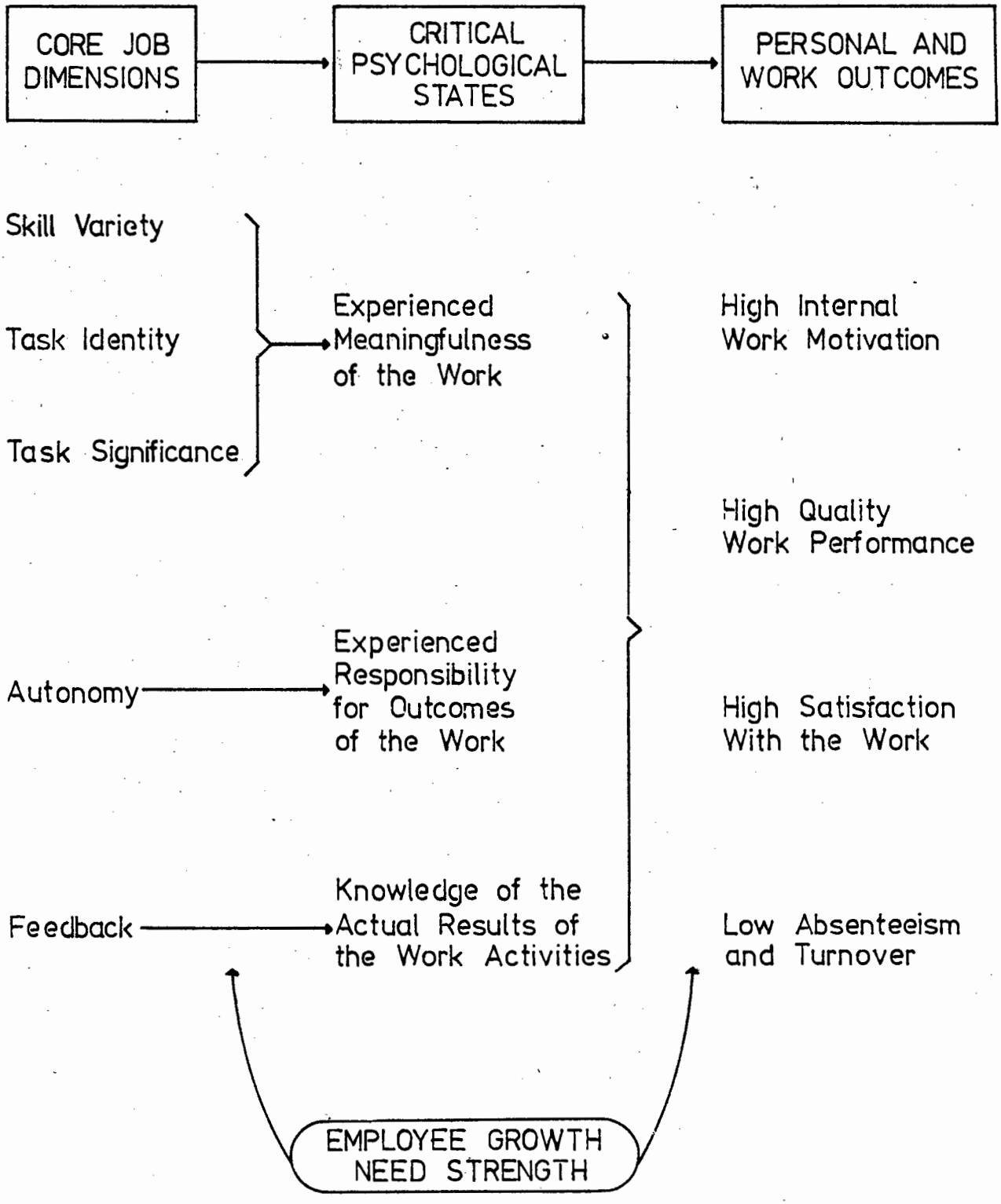


FIGURE 7.2 Influence of Core Job Dimensions on Critical Psychological States, and the Resulting Work Outcome (11f)

individual experiences in his work environement. "Autonomy" influences the degree to which he experiences responsibility for the outcome of his task, and "feedback" his knowledge of the results of his activities. When an individual's perception of his job measures highly in these "core dimensions", the job is considered to have high motivating potential. It is possible to compute a single summary index, the "motivating potential score" (MPS), for any job from measures of these "core dimensions" (22). This MPS "indicates the degree to which a particular job contains characteristics which are likely to motivate employees"(2).

Hackman et al (22), hypothesize that a job with a high MPS should be highly motivating to persons with a high GNS. GNS is considered an indicator of a person's ability to become "internally motivated" in his job, i.e. whether or not he can experience critical psychological states from interaction with his work environment. The situation is illustrated in FIGURE 7.3 (11g). A person with high GNS, will experience high internal motivation and growth satisfaction from a high MPS job. This will result in high quality performance and low absenteeism and turnover. For a person with low GNS, a high MPS job risks overstretching the individual.

Detailed analysis of the validity of each of these models of worker behaviour is beyond the scope of this study. However, it should be noted that each of the three models discussed fall into the category of "job content theory" in that they simply identify those variables (achievements, recognition, growth, etc) which influence worker performance. A major criticism of these theories is that they make no attempt to explain the underlying processes by which behaviour can be influenced. Further models, categorised as "process models", place greater emphasis on understanding the motivational processes triggered by these variables. Such models include the "Socio-Technical Systems Model", "Activation Theory", and "Expectancy theory" (42).

"Process models", while useful from an analytical standpoint, are too abstract to be useful to managers in the re-design of job

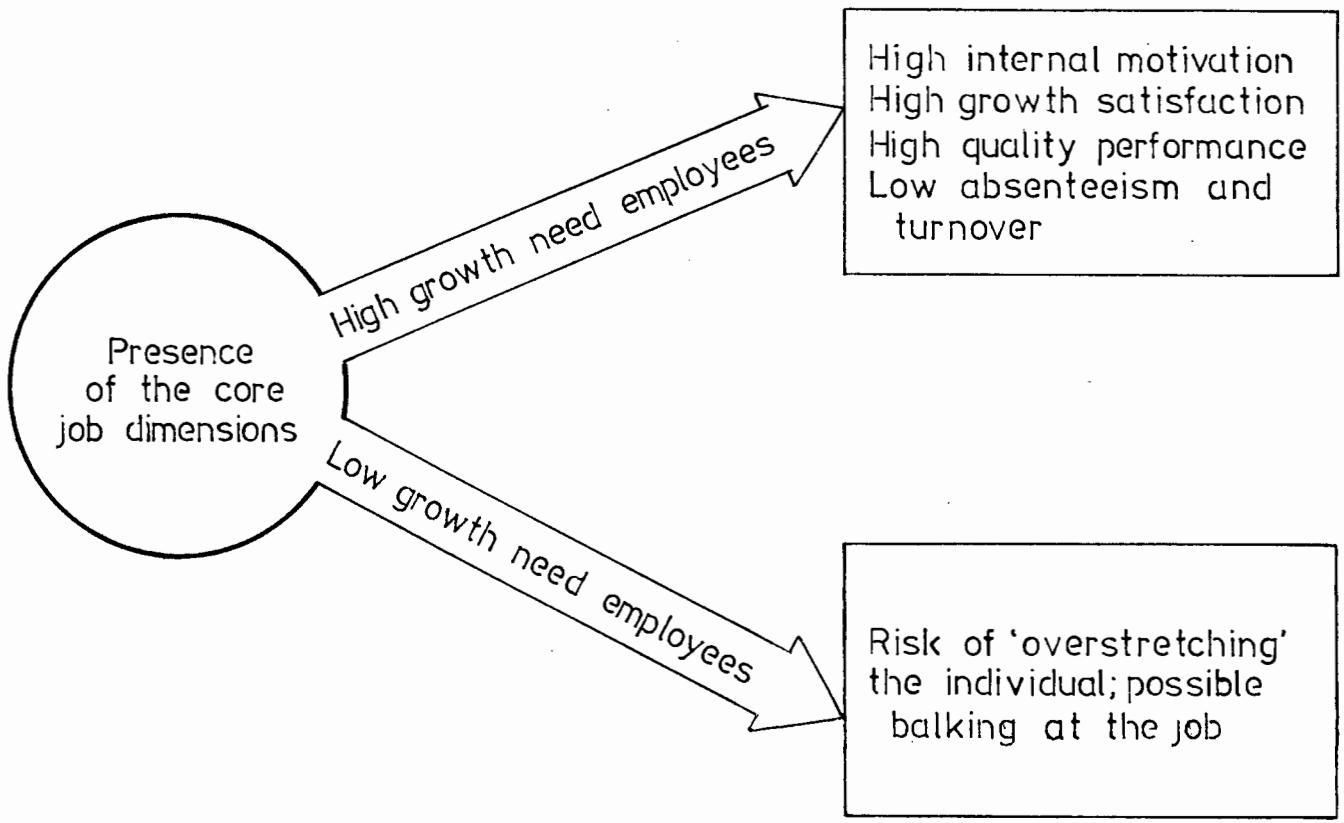


FIGURE 7.3 The Effect of a High Motivating Potential Job on Persons with Varying Growth Need Strength (11g)

structure (42). On the other hand, as indicated in the discussion above, "content theories" offer some guidance in this field, particularly when high GNS or n-Ach employees such as IS personnel are under consideration.

7.3 IS PERSONNEL PERCEPTION OF THE WORKING ENVIRONMENT

Woodruff (46) reported on the comparison of levels of job satisfaction for IS personnel with accountants and engineers. Twenty facets of the work environment were considered. IS personnel were noticeably less satisfied with their work environment than were accountants or engineers (FIGURE 7.4) Satisfaction with "advancement" and "compensation" appeared low for all categories of personnel, with IS personnel considerably below that of the others.

But not accountants see FIG 7.4

When comparing job satisfaction levels amoung four categories of IS personnel, systems analysts reported highest overall satisfaction, while production operations personnel reported the lowest. In particular, systems analyst satisfaction with "advancement" was noticeably higher than for other categories of personnel.

Cougar et al (11) studied employees' perception of "core job dimensions" and the corresponding existence of "critical psychological states" for various systems positions (Table 7.1). These results can be summarised in an MPS for each job category, which in turn can be compared with the GNS for personnel filling these positions (FIGURE 7.5).

But all except one (3rd) scores were above 4 on a scale from 1 to 7.

For DP managers and IS professionals, both showing high GNS, the motivating potential of their jobs was high, indicating a good match. However, for production operations personnel showing relatively high GNS, the motivating potential of their jobs was considered low, indicating a mismatch. Comparison of GNS and MPS for particular individuals and positions can, however, be misleading. For example, although the match appears good, turnover in the IS industry is exceptionally high. A better

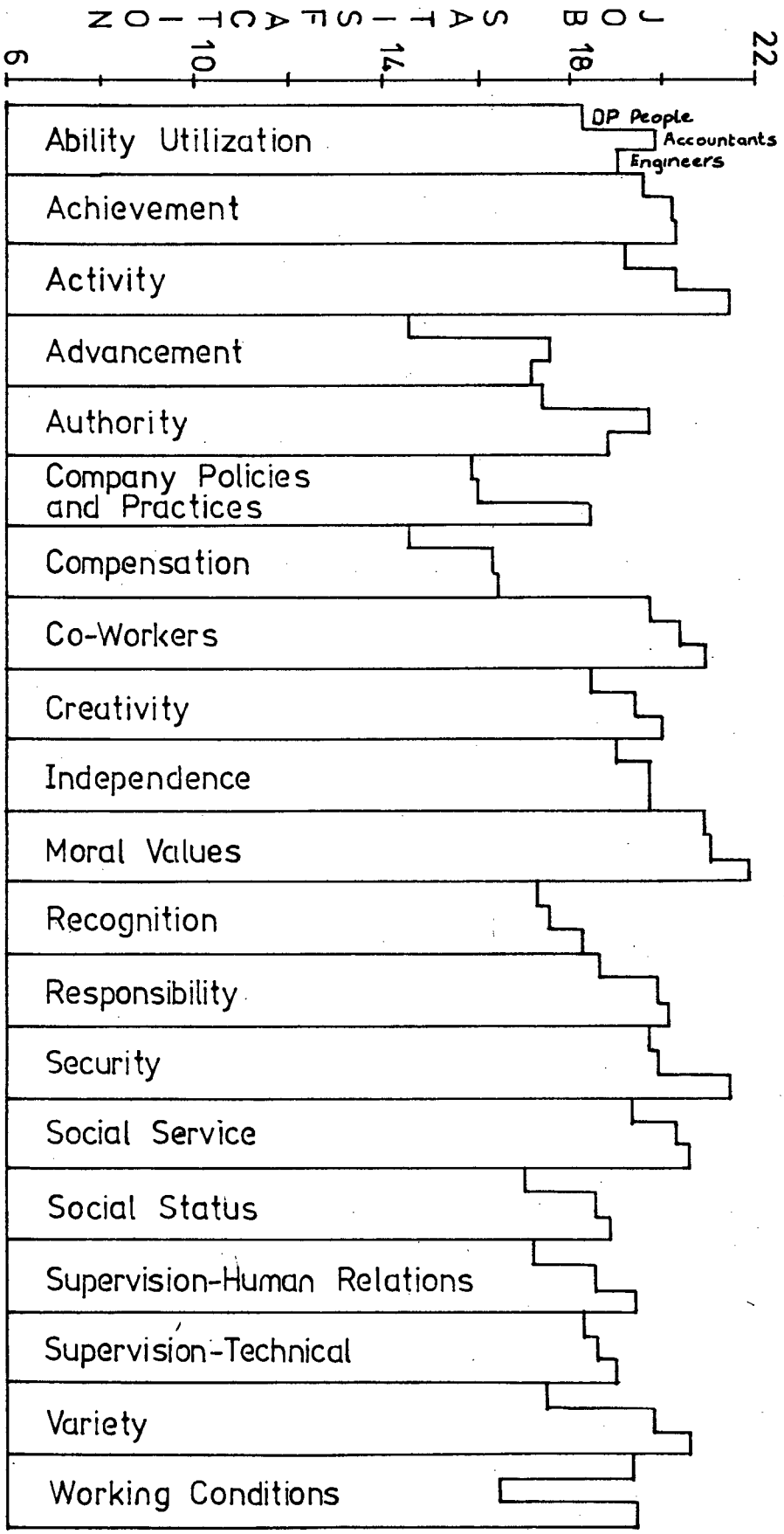


FIGURE 7.4 Job Satisfaction of Systems Personnel Compared with Accountants and Engineers (46)

Factors	Management		Professional Staff				Operational Staff		
	DP	General	DP			Other	DP	White Collar	Blue Collar
			Analysts	Pgm/Anal.	Programmers				
Core Job Dimension									
Skill Variety	6.16	5.57	5.55	5.45	5.23	5.36	3.98	4.74	4.49
Task Identity	5.80	4.72	5.37	5.29	5.00	5.06	4.53	4.76	4.60
Task Significance	6.31	5.81	5.75	5.72	5.46	5.62	5.62	5.47	5.55
Autonomy	6.10	5.37	5.31	5.48	5.13	5.35	4.08	4.85	4.83
Feedback from Job	5.25	5.15	5.20	5.05	5.10	5.08	4.62	4.88	4.76
Psychological States									
Experienced Meaningfulness	6.09	5.47	5.56	5.49	5.23	5.40	4.71	5.10	5.14
Experienced Responsibility	6.10	5.73	5.31	5.48	5.13	5.75	4.08	5.46	5.38
Knowledge of Results	4.67	4.97	4.59	4.42	4.55	5.00	4.33	4.93	5.09

(Ratings on a scale of 7, where 1 is low and 7 high)

TABLE 7.1 Comparison of Core Job Dimensions and the Resulting Psychological States between Various Work Catagories (11)

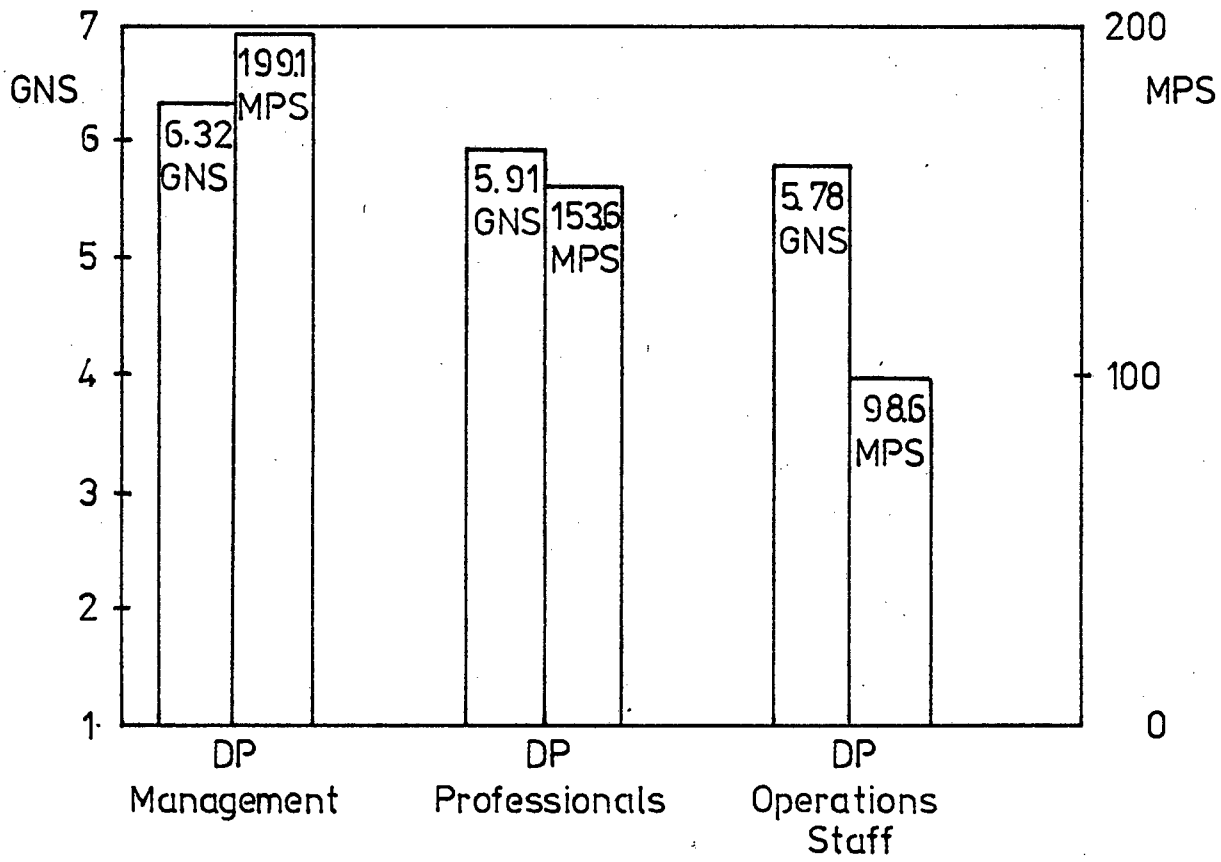


FIGURE 7.5 Comparison of GNS and MPS for Various DP Job Catagories (11)

understanding of the perception systems personnel have of their position can be obtained by examining factors effecting MPS, in more depth.

Table 7.1 reveals that for IS professionals, only "experienced meaningfulness" measured higher than other professionals. IS professionals experienced lower responsibility and knowledge of results than did other professionals. On the other hand DP management experienced greater "meaningfulness" and responsibility from their work environment than did other managers. However, they did experience significantly lower "knowledge of results" than did their peers, a "phenomenon" also affecting their subordinates.

Further analysis revealed that lack of feedback from superiors was the prime cause of low "knowledge of results" experienced by IS personnel (11). While feedback from the job itself rated higher than that experienced in other professions, feedback from superiors rated significantly lower. It was also established that IS personnel are relatively satisfied with their working environment. However, they are considerably less satisfied with their co-workers and even less with their supervisors. This tends to support the notion that relationships with supervisors, feedback, etc., significantly affect job satisfaction.

From the above, it is clear that the establishment of policy for the management and control of IS personnel, which takes cognisance of their personality characteristics and expectations from their working environment can markedly influence staff turnover, and the level of motivation and co-operation attained in supporting the objectives of the organization.

7.4 SUMMARY.

Understanding staff behaviour patterns, and an insight into the degree of satisfaction which IS personnel derive from their work can assist in the creation of a working environment and adoption of a management style suited to the development of a capable, mo-

tivated, disciplined and stable work force, considered essential to the success of any IS installation. IS personnel show the following distinct personality characteristics;

i) a strong need for personal growth and development
ii) a strong need for achievement
iii) a desire for a well ordered existence
iv) a low need for social interaction, i.e. a desire to work independently with little interference from others, but a strong desire for recognition by colleagues and other IS professionals.

Theories of worker behaviour attempt to identify aspects of the work environment or job content which affect absenteeism and turnover, or worker performance. Three theories are of particular interest;

i) Two Factor Theory distinguishes between motivating and hygiene aspects of the work environment. Improving hygiene factors will simply reduce dissatisfaction, but will not necessarily improve staff motivation. For this an improvement in motivating factors is required.

ii) Achievement Motivation Theory suggests that job enrichment, i.e. the provision of greater responsibility and challenge, will lead to increased motivation and improved performance for persons with a strong need for achievement. However, for those with a lesser need for achievement, job enrichment could lead to excessive frustration, anxiety and dissatisfaction.

iii) The Job Characteristics Model of worker behaviour suggests that "critical psychological states" are influenced by a worker's perception of his work environment. Perception of the work environment is measured in terms of specific characteristics and a

"motivating potential score" (MPS) is determined. Persons having a high need for personal growth (or high GNS) will be highly motivated by a high MPS job, while such high MPS tasks are found to be over-taxing to individuals characterised by a low need for growth (or low GNS).

Surveys reveal a good match between MPS and GNS for DP managers and professionals, however, a mismatch exists for operations staff. The high turnover of IS personnel is accounted for by examining factors influencing MPS. It is found that while satisfaction with the work itself ranked high, feedback from superiors was considered poor, adversely affecting the individuals "knowledge of results".

CHAPTER 8

ORGANIZATION FOR INFORMATION SYSTEMS MANAGEMENT

How to plan and manage, in a dynamic, demanding environment, with limited resources is possibly the most serious problem confronting the I.S. manager in a large, complex organization today. This chapter presents the author's approach to this problem*. The approach is two-fold:

NB

i) Organizational structure following a functional distribution of responsibility, with provision for I.S. activities to be guided by company policy and objectives. The extraordinary level of interaction amongst groups and individuals, demanded by the very nature of I.S. activities, requires a clear demarcation of authority and responsibility.

ii) Creation of a management climate aimed at minimising staff turn-over and maximising productivity.

8.1 ORGANIZATION

The organizational objectives of the IS department include (29):

i) Optimal DP operational or production performance.
ii) Optimal responsiveness of the systems development and implementation activity, and the orientation of this activity to the business requirements of the company.
iii) Effective management control and decision making.
iv) Long range and operational planning.

FOOTNOTE:

* This approach to managing the IS function is still under development. Initial results have been promising, although insufficient tests have been conducted to report details at this stage.

The organizational structure, or functional hierarchy, shown in FIGURE 8.1 is designed to satisfy these objectives for a medium to large organization. Different groupings of the lower level activities could be adopted. For example, for a smaller organization Software Maintenance and Development could be incorporated into Data Processing while for larger organizations, Facilities Management and D.P. Production Operations could be created as distinct functions.

Functional co-ordination within this hierarchy is afforded through policy and planning statements, and through control instructions issued by IS management, who carry responsibility for moulding IS activities to support company objectives. IS policy should be validated by an "IS Steering Committee" composed of senior company management, hosted by the IS manager. The ISSC's prime purpose is to guide IS management on matters of strategic importance and to clarify points of contention or uncertainty relating to IS requirements or architecture. In addition the ISSC should review the IS workload and major user priorities, and approve an implementation program. Communication procedures, user orientation, education and training should also be reviewed periodically.

The ISSC is intended to function only at a stragetic or corporate level, and should perform no managerial, tactical or control function. ISSC should specifically be excluded from individual project reviews, and should not become involved in detailed progress evaluation, or replanning activities, since these are, in fact, IS line responsibilities.

The frequency of ISSC meetings will obviously vary from company, to company, depending on environmental stability, the volatility of the company's strategic plan and the level of IS development or implementation activity at the time. Bi-monthly or quarterly meetings should suffice once their purpose has been established.

The IS Planning Office is responsible for ensuring continuity between ISSC decisions/agenda and internal IS departmental

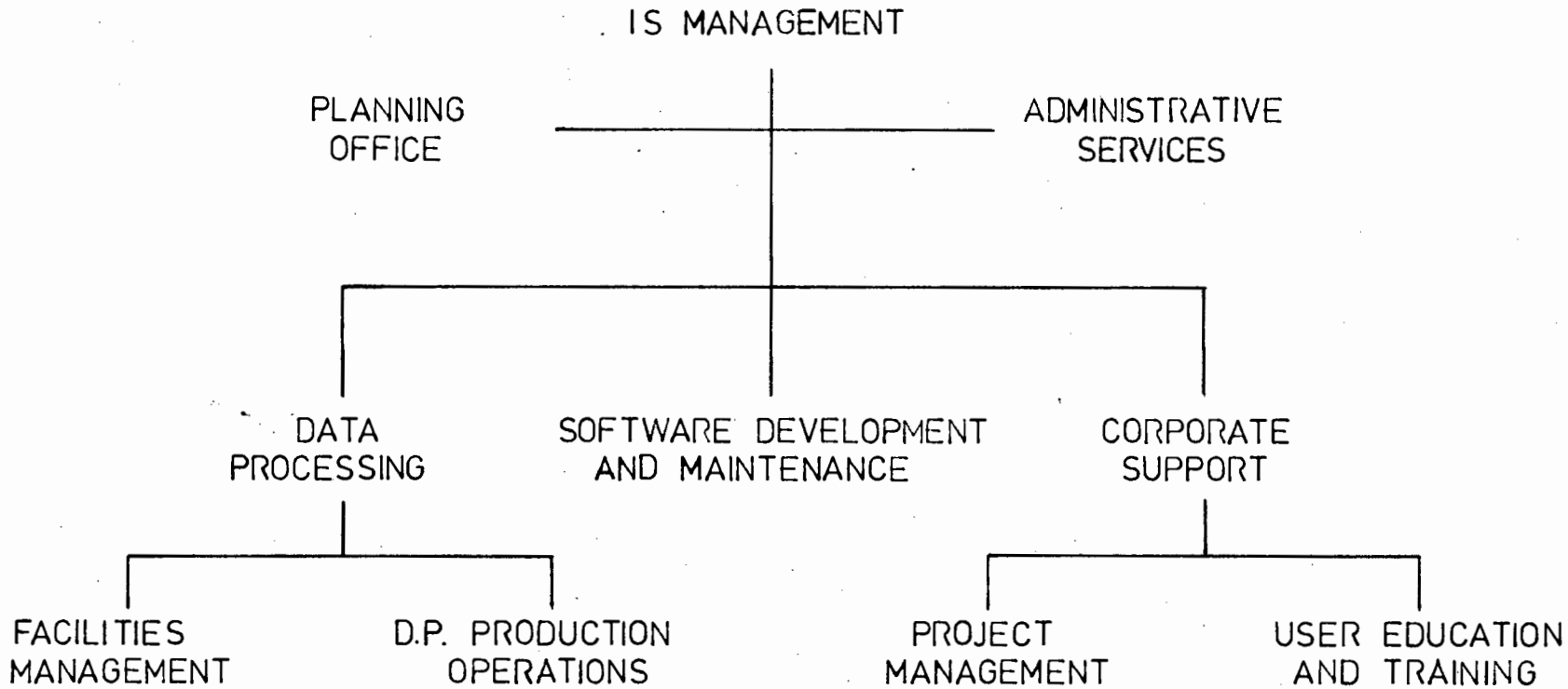


FIGURE 8.1 FUNCTIONAL HIERARCHY FOR AN IS DEPARTMENT

planning. In all but very large organizations this function is performed by the IS Manager, himself, with detailed planning being conducted by sectional heads. This is the "KEY" co-ordination and control function for IS activities. Therefore, the Internal Planning Committee composed of sectional heads and chaired by the IS manager should meet frequently (weekly, or fortnightly) to co-ordinate short-term activities. The meeting should not become involved in feedback or progress reporting and evaluation and should not allow lengthy discussions as to the merits and de-merits of a particular departments methods. Reporting and corrective action should remain a line-function, and should therefore involve manager and sub-ordinate only.

Facilities Management is intended to ensure that the computer configuration can accommodate the workload, providing a reliable service, at reasonable cost. ISSC guidelines and systems implementation plans form the major input to computer configuration reviews, which should be conducted quarterly. The purpose of these reviews is to ensure adequate capacity and to synchronise hardware and software changes with other IS activity.

DP Production Operations is responsible for the efficient operation of the computer service. Trends in computer workload influence the capacity requirements forecast of Facilities Management.

Provision of technically sound application software is the prime objective of the Systems Development and Maintenance function. Maintenance or development of software is initiated by fault reports from Production Operations, and change or development requests from Corporate Support, who form the main interface with the end user. The"Software Configuration Management" technique described in Chapter 6 forms the main tool for controlling the software product. Programming techniques and software technology will obviously depend on the relative sophistication of the particular computer installation. While technological improvement is obviously desirable, in the authors view, the technical image of the installation i.e. the technical sophistication of its

products, should not take precedence over support for the overall company objectives (refer chapter 3). The prime goal is the timeous provision of an effective and reliable product.

The Corporate Support function is intended to rationalise the interaction between users and IS. It is staffed by senior personnel, preferably having extensive experience in the company and, therefore, considerable understanding of the methods and procedures used, or alternatively, having considerable expertise in a particular user field to which IS techniques are to be introduced. These personnel provide basic user support in the design and implementation of IS systems, and in the introduction of new methods. In the author's experience, introduction of this support function, as distinct from Systems Development, has improved user confidence in the IS service, and released Systems Development staff to concentrate on providing technically sound, reliable products.

As the major interface between users and IS, Corporate Support perform the QA function of project management (refer chapter 5 and Appendix A). Being mature persons, filling senior posts in the organization they tend to be stable with respect to membership of the organization, so that the staff turnover problem (refer chapter 2) at least from the user point of view, is considerably reduced. With previously acquired expertise in the particular user field under consideration, Corporate Support staff are more readily accepted by users than was the experience of the software personnel assigned the support task previously.

8.2 MANAGEMENT CLIMATE

A successful IS installation requires a stable, motivated and capable work force, committed to contributing jointly to achieving the goals of the organization. For this, an environment must be created which is particularly suited to the type of person employed. Effort should be expended in reducing staff turn-over to a manageable level and to developing a climate in which staff feel motivated to achieving greater productivity.

8.2.1 STAFF STABILITY

Investigation into the causes of absenteeism and staff turn-over consistently points to a dissatisfaction with career posibilities within an organization. Scholl (39) found generally that career possibilities and the increase in salary associated with promotion jointly influence an employees decision to remain in an organizations employ.

More directly related to IS staff turnover, satisfaction with "advancement" and "compensation" appear particularly low for systems personnel compared with other professions (refer FIGURE 7.4 and section 7.3), possibly accounting for the alarmingly high turnover rate of IS personnel (refer section 2.3.4). It is significant that "compensation" is what Herzberg (23) terms a hygiene factor (refer section 7.2.1). When rated low, this creates dissatisfaction, thereby supporting a decision to leave a company's employ. On the other hand, "advancement" is considered a motivating factor whose presence could have a positive influence on a decision to remain with a company. However, without the hygiene factor being satisfied, "advancement" would have little influence.

Bartol (1) found an IS employee's commitment to remaining in an organizations employ to be significantly improved when professional behaviour was perceived as a source of reward within the organization. This could indicate a desire to build a professional career with such an organization. This deduction is supported by La Belle et al (26) who demonstrated a significant reduction in staff turnover, at a major IS installation, following the introduction of a human resource development program designed to satisfy the career aspirations of the IS professional.

Following the example of La Belle et al (26) the author proposes the introduction of a "Human Resources Development Program" for IS personnel. The program should include ;

i) Creation of a career path foundation, analogous to a project life cycle, carrying detailed descriptions of

each job and their interrelationships so that a career path can be plotted for each employee (Figure 8.2) Here it should be noted that the length of the career path, relative distance from the career ceiling and mobility opportunity of a position all influence a persons decision to remain in the organizations employ. This foundation should include forecast staffing requirements and, particularly, plans to create more senior positions since these influence the decision of staff members who have already attained or nearly attained their ceiling.

ii) Definition of the "skills", and the various levels of proficiency within the skill, which can be attained in a particular position i.e. preparation of a "skills glossary". These levels of proficiency in each skill should be correlated with the job descriptions in the "career path foundation" so that a standard can be set for each post and each level in the organizational hierarchy.

iii) Identification of available educational opportunities for each skill and proficiency level. Record of performance ratings for each person should be maintained and correlated with personal performance appraisals to assess career growth capabilities.

iv) Preparation of a career plan for each individual. The long-term career aspirations of the employee together with the job description and skills profile should be used to prepare an education program and to plan job experience and skill development.

v) Performance appraisal should be based on the individuals agreed career plan. Deserving employees must be seen to be given promotional opportunities and the system must be seen to favour promotion of employees rather than appointment of outsiders to more

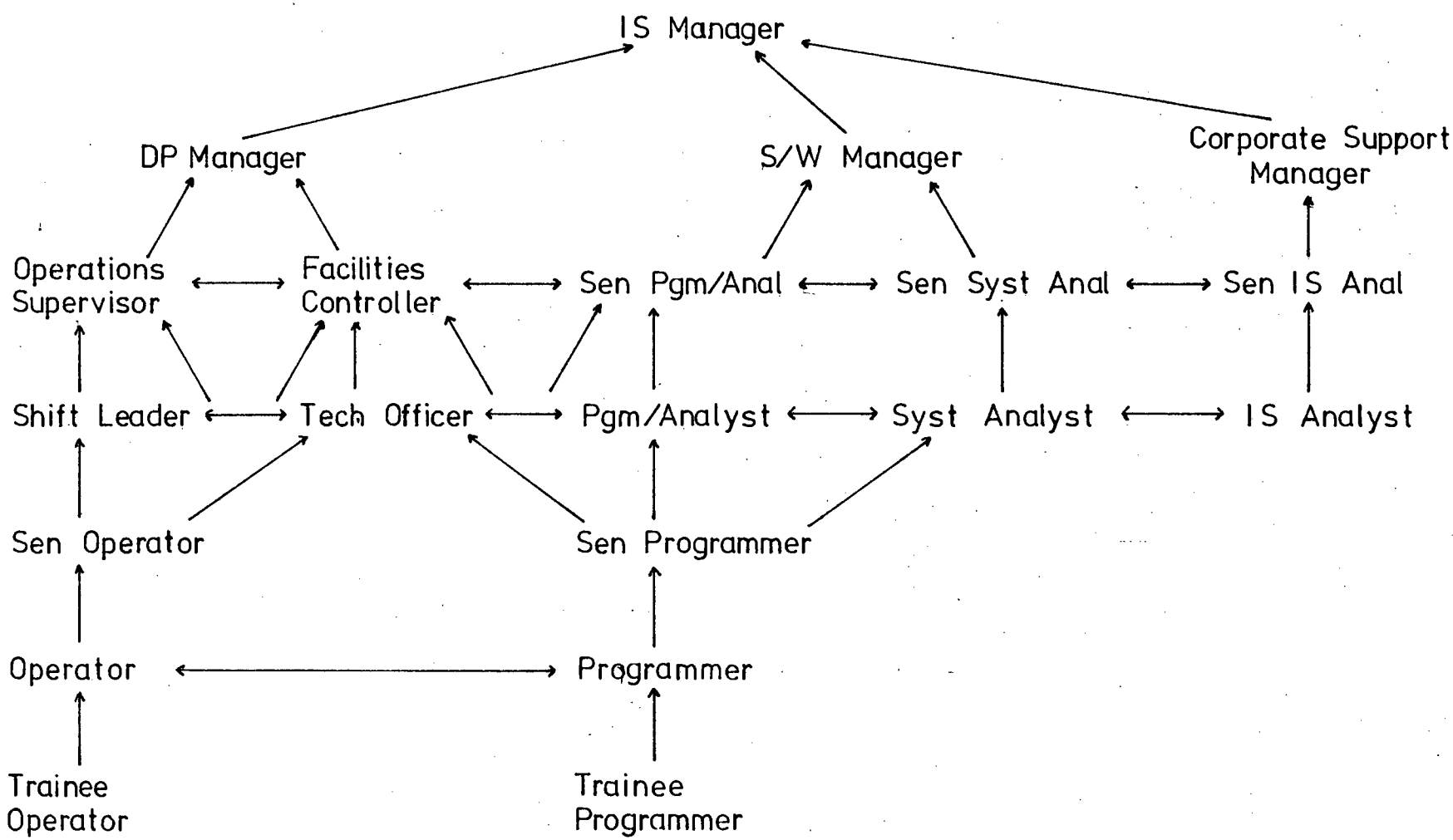


FIGURE 8.2 Typical career paths within an IS department

senior positions. The career plan is the beginning of a pact between employer and employee on detailed career development. Variance from plan should be discussed frequently (semi-annual or annual reviews will usually suffice), and corrective action agreed.

8.2.2 MOTIVATION AS A MEANS OF IMPROVING PRODUCTIVITY

The personality characteristics of IS personnel, and their perception of the working environment provides some guidance to motivating staff to greater productivity and greater co-operation in achieving the organization's goals. IS personnel display an exceptionally high "growth need strength (GNS)", "need for achievement " (n-Ach), and "need for order" in their working environment (refer 7.1). Their perception of their working environment demonstrates a strong deficiency in motivating potential, particularly in the areas of "experienced responsibility" and "knowledge of results".

With this in mind, a productivity improvement program for IS activities should include ;

i) Definition of individual responsibility and job enrichment.
ii) Improved feedback and control procedures.
iii) Development of a leadership approach suited to IS personnel.

8.2.2.1 RESPONSIBILITY AND JOB ENRICHMENT

Frequently, and particularly in developing organizations, management are found actively performing tasks which should be assigned to subordinates. The reasons for this are often historical, on promotion the manager retains the more complex tasks from his previous post. Since no one else has been trained in this area, or, alternatively, lacking confidence in the capability of his subordinates the manager undertakes the more complex tasks himself. Often, on promotion from junior ranks, the

manager is unsure of his new responsibilities and, therefore, retains junior responsibilities so as to "feel busy" and to demonstrate continued personal achievement. In such cases, performance appraisals are often disappointing, responsibilities of the post having been neglected. This often results in de-motivation.

To avoid uncertainty and ensure that all tasks be properly designated, a clear definition of individual responsibility is required. Within the framework of the established functional hierarchy (refer section 8.1) the specific responsibilities of each post must be defined. Here, distinction should be drawn between "responsibility" and "accountability" (40), where,

> Responsibility refers to tasks or activities specifically assigned to a post or person i.e. tasks which will be performed by the incumbent of that post in person, while
>
> Accountability refers to the area of concern to a particular person, and includes tasks or activities delegated to subordinates. Accountability for delegated activities, therefore, implies a responsibility for co-ordinating and controlling these activities.

Since the intention is to move responsibility down the hierarchy, the lowest level responsibilities are considered first, and the process continued upward. Senior positions are considered after all reporting to them have been completed. Analysis of the merits or de-merits of specific techniques for obtaining information required to build this responsibility structure are beyond the scope of this thesis. It is sufficient to mention that a sufficient cross-section of the organization's staff, interacting with a specific post, together with the incumbent, his superior and subordinates should contribute to its responsibility definition. Discussion of the incumbent's current activities should be avoided, a definition of the responsibilities associated with the "position" being the objective. Following this initial structuring of responsibilities, a job enrichment program can be

introduced in conjunction with the "Human Resource Development Program" discussed in section 8.2.1 above.

This involves manipulation of responsibilities, or, effectively, manipulation of the motivating factors of jobs (23). The goal is improvement in job satisfaction, motivation of staff and the effective utilisation of personnel. Detailed analysis of the job enrichment process is beyond the scope of this thesis. However, for completion, a brief introduction to the principles involved is included in Appendix B (23).

8.2.2.2 FEEDBACK AND CONTROL

Control of the activities of subordinates is a line function and therefore involves superior and subordinate only. The author has found it essential to the development of a motivated work force, that this line responsibility not be broken by higher management intervention. How this control should be exercised, and what influence senior management can, or should, exert over the activities or performance of his department frequently presents a problem. The author would suggest the following approach (40).

The type of control required depends largely on the type of activity involved. Activities fall into two basic categories, high-level and low-level.

i) Low level activities require no subjective decisions. The task and the manner in which they are to be performed is specified precisely, with instructions to be followed being clearly defined. Batch capturing of data, loading of software, running report programs, etc., fall into this category. For such activities, management by objectives produces excellent results. Here a target is agreed between superior and subordinate, the prime measure of performance being efficiency (measured in terms of time, quality and quantity). The subordinate's performance is then judged on whether this target has been achieved. In the

author's organization, this approach reduced wasted capacity in DP Production Operations from 8 % to 3 % over a 4 month period. Results are shown in FIGURE 8.3.

ii) High level activities require subjective, reasoned decisions. The prime concern here is the effective execution of a responsibility. Although the responsibility may be clearly defined (refer section 8.2.2.1) the manner in which a task should be conducted cannot be precisely specified. The task involves gauging the effect of a particular action, analysing trends and taking a subjective decision on future action. Most management or supervisory activities affecting the performance of a particular function fall into this category, as do systems analysis or corporate support activities. The prime measure of performance for such activities is the effectiveness of the action plan.

For high level activities the responsibility structure (section 8.2.2.1) should be used to identify key areas requiring specific attention (40). These "KEY performance areas" (KPA's) should be agreed between superior and subordinate and then form the basis for performance appraisal. The subordinate's analysis of KPA's and his proposed action plan form the basis for discussion between manager and subordinate.

Through these discussions the manager can guide the activities of his department, and feedback to subordinates his approval or disapproval of their performance in preparing and executing action plans. This interaction allows a pro-active approach to management. It prevents managers assuming subordinate responsibilities through reactive correction of errors. The approach requires that managers delegate responsibility while retaining accountability for the performance of their departments.

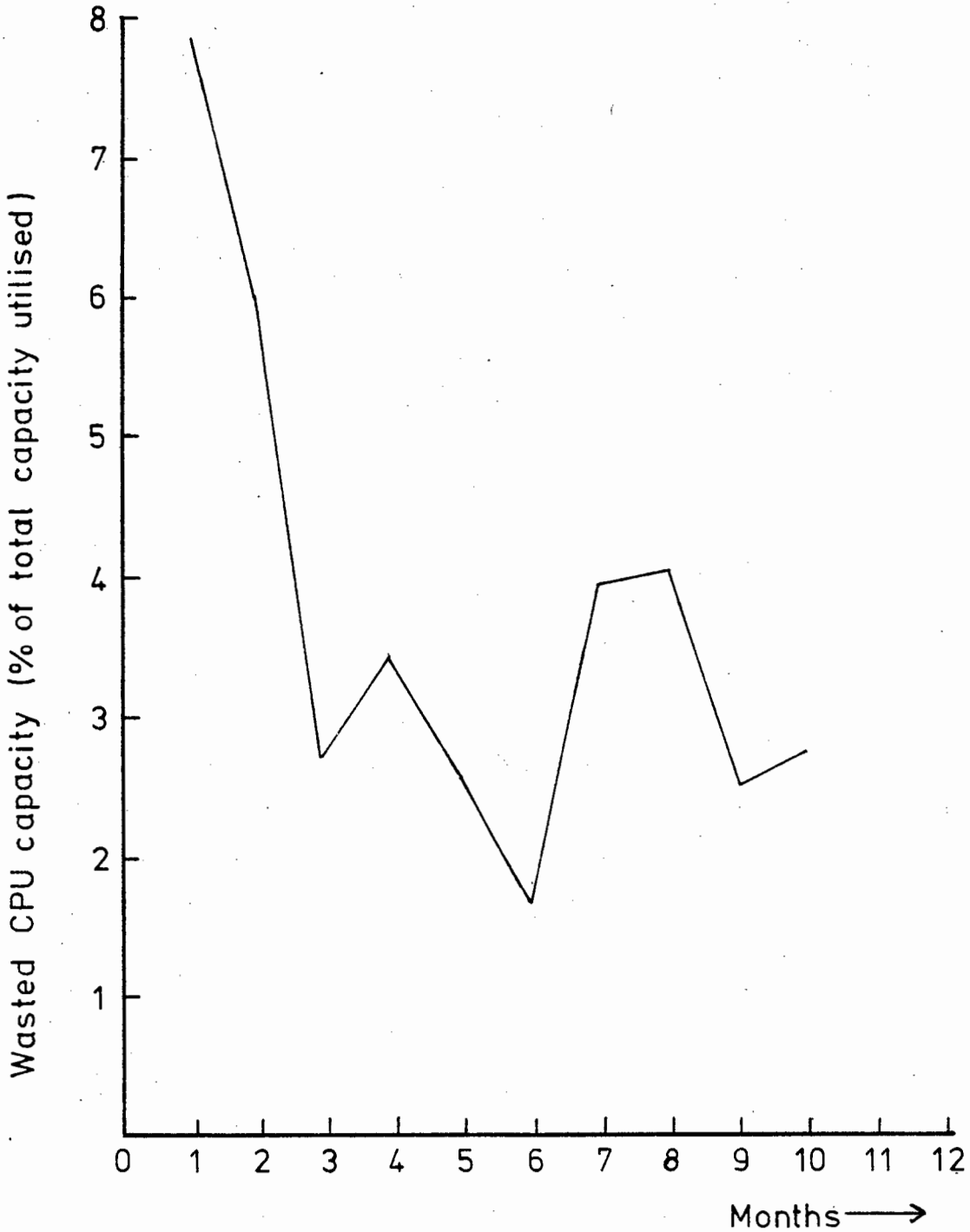


FIGURE 8.3 Reduction in wasted processing capacity

8.2.2.3 TASK ASSIGNMENT AND LEADERSHIP APPROACH

Although personality traits amongst IS personnel are more homogeneous than many other departments, individual differences are evident. Couger et al (11h) suggest that consideration of such personality differences, and of the employee's expectations from his working environment in the assignment of tasks, would influence productivity. When the scope of the task matches the individual's need for growth and for challenge in his work, high motivation can be expected. This, in turn, leads to high productivity. This influence of congruency between the scope of a task (i.e. the degree of content of "core job dimensions") and an individual's GNS, on his internal motivation is illustrated in FIGURE 8.4A (11i). The four cells in this model portray the possible combinations of GNS and scope of task assigned. Variations in leadership style can further enhance staff motivation in high congruence situations, and counter the negative effects on motivation of low congruence situations - FIGURE 8.4B (11j).

i) CELL 1. High GNS individuals desire challenging tasks for which they feel responsible. Tasks, such as new system design and development, are generally considered in this category. Such a match between GNS and task scope is found to be highly motivating, resulting in a high performance output from the individual.

Participative and achievement orientated leadership would enhance motivation in this situation. This would involve the employee in decision making and in the setting of goals relating to his work. It would enhance the meaningfulness of the task from the employee's point of view. Carrying responsibility for the achievement of these goals, and control of the task, would further enhance the high GNS individual's desire for autonomy and self control.

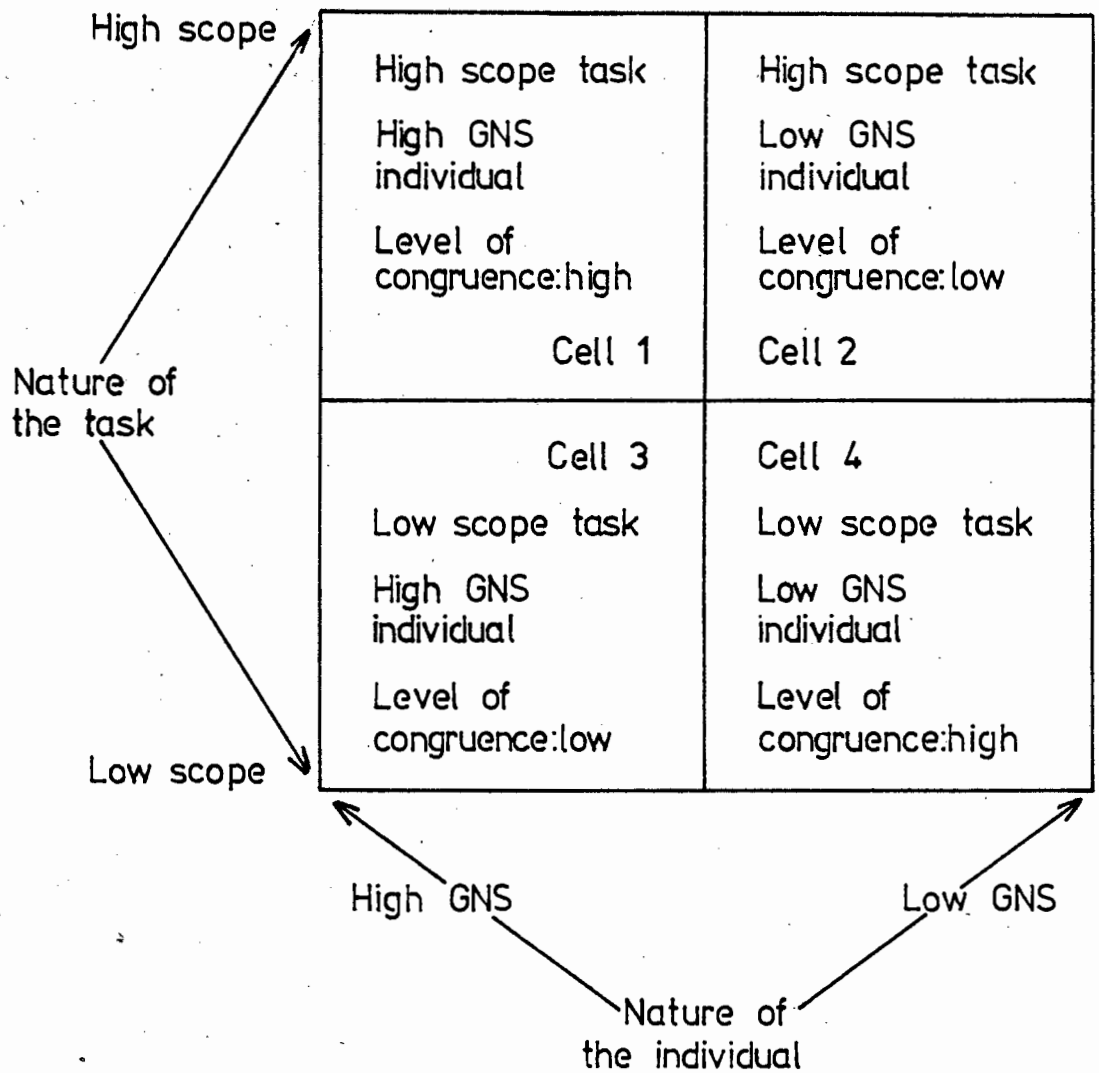


FIGURE 8.4A Level of Congruence Resulting from Degree of Match Between GNS and Task Scope (11i)

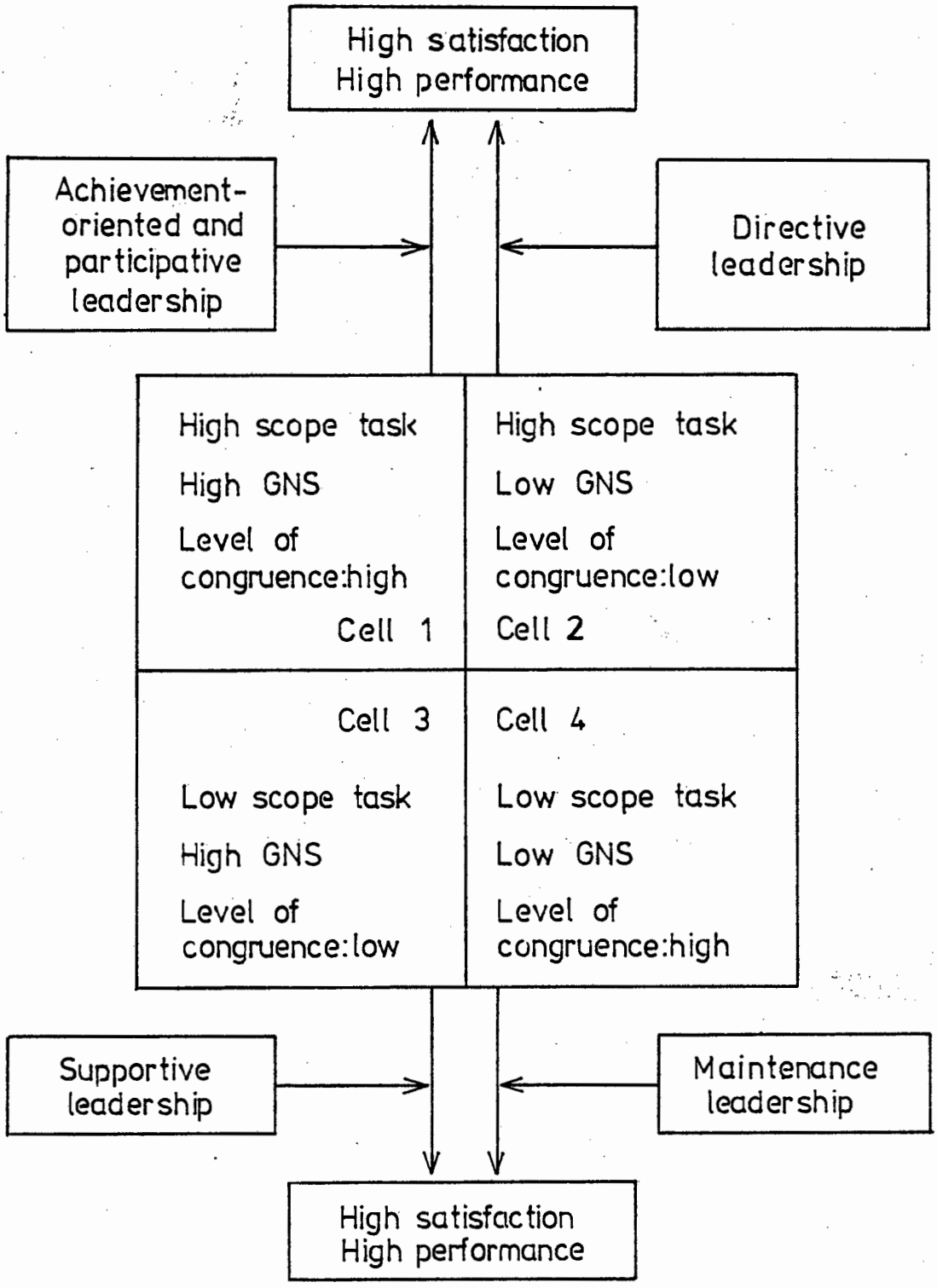


FIGURE 8.4B Leadership Style Needed to Support Varying Match on GNS and Task Scope (11j)

ii) CELL 2. Assignment of a high scope task to an employee having a low GNS results in a poor motivational environment. This may result when the employee finds himself out of his depth, e.g. when assigned tasks for which he has not been properly trained. Unable to cope with this "ambiguous situation" he experiences difficulty in structuring his own activities (11h). This situation would not produce good productivity unless "other measures (were) introduced to influence the disparity between task and individual needs" (11h).

Direct leadership, involving the structuring of the task by planning, organizing, co-ordinating and controlling the activities of subordinates, would be the most effective leadership approach in this situation.

iii) CELL 3. When an employee having high GNS is assigned a low scope task he is likely to become frustrated and dissatisfied (11h). Supportive leadership, in which the leader exhibits concern for the well being of subordinates will probably minimize the frustration and dissatisfaction generated by this situation (11h). Suggestions for improving productivity and, thereby, minimizing the duration of this frustration will be discussed below.

iv) CELL 4. A high level of congruence exists between task and individual need when a low scope task is assigned to an employee having low GNS (11h). In this situation it is expected that the individual would be highly motivated and a concommitant level of productivity could be expected.

The individual finds such a situation satisfying and thus requires little supportive leadership. In addition, he requires no job structuring or directive

leadership. However, participating in achievement oriented leadership would lead to excessive frustration. A form of maintenance leadership, or minimum-interference leadership, is therefore most appropriate in this situation (11h). The leader monitors progress, but does not interfere in day to day operations. Only when performance or progress problems arise does he intervene. Once the problem has been corrected he reverts to the maintenance leadership approach.

Obviously, tasks cannot always be assigned to suit an individual's personality, or to satisfy his expectations. The high-congruence situation depicted in Cells 1 and 4 of FIGURE 8.4B can rarely be applied continuously (11h). In IS projects, some tasks are found to be challenging, others, equally important to the efficient fulfillment of the project, mundane. All members of a project team must contribute to the more mundane tasks such as maintenance and documentation. This results in an incongruence in their personality/job relationship of the type depicted in Cell 3 of FIGURE 8.4B. From a motivational point of view it is important that the time and effort required to complete these tasks be kept to a minimum.

In the author's opinion, standardisation would minimize the effort required in the less challenging areas. Low scope tasks lend themselves to standardisation. Standard format and content for systems documentation for example, would minimize the effort required in its preparation. Where possible, low scope tasks should be included as sections of more desirable tasks. For example, rather than including all system documentation in a single task undertaken once the system is fully developed, documentation for various segments of a project should rather be prepared before the following segments are undertaken. This avoids the possibility of these (undesirable) tasks being neglected in favour of more interesting or challenging work. It also produces a sense of achievement at completing a job properly, thereby motivating the employee to even greater achievement.

Incongruence similar to that depicted in CELL 2 of FIGURE 8.4A often results from the high staff turnover experienced in the IS industry (refer SECTION 2.3.4)(31)). Staff are often assigned tasks for which they are under qualified. For example, a programmer might be assigned the task of designing a system upon the resignation of the analyst originally assigned the task. The author has found that in such situations the programmer often questions his level of appointment in the organization, reasoning that since he has been assigned analyst level tasks, he should be promoted to the analyst position. Under these circumstances it is important that management take tighter control of the project and insist on regular reviews. The analyst level responsibility is then carried by management rather than by the lower level programmer. The situation could then be turned to the organization's advantage, and the task be identified as a training project, preparing the programmer for analyst level work.

From the above, it is clear that the establishment of policy for the management and control of IS personnel, which takes cognisance of their personality characteristics and expectations from their working environment can markedly influence the level of motivation and co-operation attained in the implementation of a management methodology for IS projects.

8.3 SUMMARY.

A two fold approach to managing an IS department is proposed. Firstly, an organizational structure following a functional distribution of responsibilities.

i) An "IS Steering Committee" to guide IS management on the strategic importance of various system developments. Co-ordination and control of IS activities to support ISSC policy guidelines is the responsibility of the IS Manager.

ii) Facilities management to ensure adequate computer facilities.

Or the purchase of outside facilities

iii) DP Production Operations.

iv) Software Development and Maintenance.

v) Corporate Support as the main interface between users and the IS Department.

Secondly, creation of a management climate designed to stabilise and motivate the IS workforce.

i) A "Human Resources Development Program" to be introduced to prepare a career plan for each employee based on skills required by the organization and the long-term career aspirations of the individual.

ii) Frequent performance appraisals based on the agreed career plan.

iii) Introduction of a job enrichment program aimed at improved job satisfaction, staff motivation and effective utilisation of personnel.

iv) Establishment of a management style which takes cognisance of the personality characteristics and work expectations of IS personnel.

CHAPTER 9

CONCLUSION

The rapid growth of computerisation applications in business and industry during the past three decades, the accompanying demand for sophisticated software and the shortage of skilled software personnel has resulted in a "significant lag" (25) in the development of a management methodology for software projects. The impact of poor management of software on the cost of computerisation projects is aggravated by the inversion, in recent years, of the relative investment in hardware and software (FIGURE 2.1). Prior to the 1970's, any computerisation project required a considerably greater investment in hardware than in software. In the 1980's however, this position has been reversed. With the major investment now being allocated to software, the effect of poor management in this area is considerable and the urgency for establishing a workable management methodology, critical.

Poor management methodology for software or IS projects stems largely from an inability to define a software product in terms understandable to non-software oriented personnel. The rapid advance of computer technology has limited the dispersion of IS knowledge, and resulted in IS technology being poorly understood. There is, therefore, a tendency amongst users and management, to consider the development and implementation of software as the responsibility of the IS department. Failure in the field is then attributed to IS personnel, with user involvement, co-operation, ability and enthusiasm being considered of minimal significance. On the other hand, IS personnel often attribute the incomplete provision of facilities required to perform a particular function to the user's inability to specify his requirements fully, or to his lack of co-operation in the definition stage of the project. It is argued that users contribute little effort to the definition phase of the project, yet expect facilities, not originally specified, to be provided during implementation.

Project management in the IS industry is far from mature.

Accurate estimates of a project's cost or duration can seldom be obtained. In addition, standard practices and procedures are not well developed, so that unique procedures are followed by each project leader. Under such conditions staff performance cannot be effectively monitored, and allocation of resources must be left to the ad hoc judgement of the various project leaders. In addition, lack of standard practices and procedures accentuates the frustrations of non-systems personnel when discussing IS projects. Without standardised definitions, they have difficulty comparing project complexities, cost and duration, or resource requirements.

The principles relating to a successful management methodology for software projects were discussed in this thesis. To perform an effective service in any organization IS development must be seen in perspective. The prime objective in the development and implementation of information systems is to support the ultimate goals of the total organization. In this regard, the user's acceptance of the service offered is a prime indicator of its contribution to the achievement of the organization's goals. Obviously, this requires the provision of facilities which allow the user to perform his task more efficiently and more effectively. Accurate specification of required facilities, and of an implementation schedule providing effective support and assistance to the user must, therefore, form a major objective of IS management strategy. Verification of the technical feasibility and economic justification of a project is essential to ensuring minimum wasted effort, and senior management's commitment to the project.

The management methodology discussed in this thesis comprises four basic functions;

i) quality assurance
ii) project measurement and control
iii) product configuration management
iv) organization and control of the workforce

Quality assurance refers to activities required to provide

confidence in the capabilities of a product. This involves the establishment of acceptable technical standards for software, and the notification of all affected persons of the status of the product in time for errors and deficiencies to be reported and corrected with minimal effect on the final cost or implementation schedule. The goal is to ensure that the product has been accurately and completely specified, that it will perform as specified, and that it will be delivered on schedule.

Many of the problems experienced in the measurement and control of IS projects evolve from the abstract nature of the product. This is found to be particularly confusing to non-software oriented personnel, especially since no standard is used for describing the product. "Visibility" can be considerably improved by adopting a "standard" life-cycle for defining projects. This allows non-software oriented managers to become familiar with IS terminology. The life-cycle breakdown for products allows estimates of time and resource requirements for each phase. This can then be translated into cost and duration estimates for projects, which facilitates an adequate estimation of any risks that may be incurred when management decisions affect the progress of projects.

Configuration control allows software configuration to be accurately defined at discrete points in time. This is effected by identifying the "baseline" contents of the product as related to phases in the project life-cycle. With the initial release from a life-cycle phase as "baseline", and approved and proposed changes to this configuration, the status of the product can be accurately defined at any point in time. This in turn allows verification and validation of product configuration, i.e. verification that the contents specified at one baseline or update is satisfactorily echoed at the succeeding baseline or update, and validation of the functioning of the end product against the requirements specified. For project management, this allows precise monitoring of progress during the products development.

Understanding the behaviour patterns, expectations from the work

environment, and career aspirations of IS personnel provides guidance in the development of a capable, motivated, disciplined and stable workforce. A management climate must be created which provides challenge and responsibility, recognition and a sense of achievement, and opportunities for advancement and personal growth. The responsibilities of this workforce should be organized along functional lines, distinguishing between computer facilities management, DP production operations, systems development and corporate support functions. Making provision for user priorities and company strategy to influence IS activities through an IS Steering Committee, composed of senior company management, provides for an effective IS service supporting the company's overall objectives.

Implementation of this methodology in the author's present organization has not advanced sufficiently to report conclusive results. However, continued attention will need to be given to management principles, and to the development of techniques for managing IS projects and installations if maximum benefit is to be derived from computerisation of informations systems.

APPENDIX A

A STANDARD FOR SOFTWARE QUALITY ASSURANCE PLANS

The department or person in an organization responsible for Software Quality Assurance should prepare a Software Quality Assurance (SQA) Plan. These plans should include the sections listed below which are considered minimum acceptable requirements for SQA plans for non critical software in a medium to large manufacturing organization*.

SECTION 1 - PROJECT DEFINITION

A.1.1 SCOPE AND PURPOSE

This section shall delineate the scope and purpose of the particular SQA Plan. The problem or concept must be outlined briefly, and the project objective should be clearly defined. The names of the software product items covered by the SQA Plan shall be listed and their intended application described.

A.1.2 ORGANIZATION AND MANAGEMENT

This section shall describe the organization, tasks and responsibilities.

A.1.2.1 ORGANIZATION

This paragraph shall depict the organizational structure. In particular the following must be identified:

i) Project Leader

FOOTNOTE

* This "standard" for SQA plans includes a subset of the standard for "critical software" approved by the IEEE Computer Society Software Engineering Standard Sub-committee(8). In addition the author has included sections which his own experience have shown to be necessary.

ii) Prime representatives from each of the following*

- Corporate Support
- Software Development and Maintenance
- Data Processing
- User Departments

Delegated responsibilities for each major element of the organization must be defined.

A.1.2.2 TASKS AND RESPONSIBILITIES

This paragraph shall describe the tasks covered by the plan, and identify the specific organizational elements responsible for each task. The tasks should either form part of the project management function, or, alternatively, be associated with a specific phase of the software life-cycle. Tasks should therefore be grouped under the following categories

i) Project Management
ii) Requirements Specifications and Feasibility study
iii) Functional Design
iv) Technical Design
v) Software development and coding
vi) System Testing
vii) Implementation
viii) Maintenance

SECTION 2 - REFERENCES

This section shall provide a complete list of documents referenced elsewhere in the text of this plan.

FOOTNOTE

* Corporate Support, Software Development and Maintenance and Data Processing form the three main branches of the I.S.department's activities as described in Chapter 8, Figure 8.1

SECTION 3 - DOCUMENTATION

A.3.1 PURPOSE

This section shall identify the documentation governing the development and verification of software, and state how the documents are to be checked for adequacy.

A.3.2 MINIMUM DOCUMENTATION REQUIREMENTS

The following minimum documentation is required to ensure that the implementation of information systems satisfies requirements.

A.3.2.1 REQUIREMENTS SPECIFICATION AND FEASIBILITY STUDY (RSFS)

The RSFS shall clearly and precisely describe each of the following

i) FUNCTIONAL CHARACTERISTICS

This extends the scope and purpose of the project, as defined in A.1.1 above, to include the broad functional characteristics of the system. Qualitative and quantitative characteristics of, and the relationship between inputs and outputs must be defined. Interaction with data bases, and with other systems must be identified.

ii) PROJECT CONSTRAINTS

Project constraints must be clearly defined. The following areas should be covered;

- User environment
- Hardware limitations
- Software environment
- Quality Assurance and Auditing policy
- Budget and resource constraints
- Time constraints (urgency and priority)

iii) FEASIBILITY STUDY REPORT

The purpose is to recommend a feasible approach to the project. This recommendation should include an inventory of:

- Recommended reports and screens, identifying data element

content.

- Input/output requirements (mode, volume, etc.)
- Functional responsibilities, eg for data maintenance, report distribution etc.
- Data base and system interaction, identifying data elements and location.
- System controls.

The report should include a cost-benefit analysis and an analysis of functional benefits derived from the system. A project schedule must be included, detailing milestones and anticipated resource requirements.

A.3.2.2 SOFTWARE DESIGN DESCRIPTION (SDD)

The SDD is the prime output from the design phase of the project life-cycle. It shall describe the major components of the software design, including data basis and internal interfaces. An expansion of this description shall be included to describe each sub-component of the major component.

A.3.2.3 SOFTWARE VERIFICATION PLAN (SVP)

The SVP shall describe the methods to be used to verify that

i) the requirements in the RSFS are implemented in the SDD, and further into the software code

ii) the code, when executed, meets the requirements expressed in the RSFS.

A.3.3 OTHER

Other documentation may include the following;

i) Computer program development plan

ii) Configuration management plan

iii) Standards and procedures manual.

SECTION 4 - STANDARDS, PRACTICES AND CONVENTIONS

A.4.1 PURPOSE

This section shall identify the standards, procedures and conventions to be applied, and describe how compliance with these

items is to be monitored.

A.4.2 CONTENT

Subjects to be covered shall include the basic technical design and coding activities, such as documentation, naming and coding conventions, programming language, technicque etc. In particular, the following shall be included

i) Documentation standards
ii) Logic structure standards
iii) Coding standards
iv) Commentary standards.

SECTION 5 - REVIEW AND AUDIT

A.5.1 PURPOSE

This section shall define the technical reviews and audits to be conducted and state how these are to be accomplished.

A.5.2 MINIMUM REQUIREMENTS

As a minimum, a review or audit should be conducted after each phase of the life-cycle.

A.5.2.1 REQUIREMENTS REVIEW

This is held to ensure the adequacy of the requirements stated in the RSFR.

A.5.2.2 DESIGN REVIEW

This is held to evaluate the technical adequacy and to determine the acceptability of the SDD in satisfying the requirements of the RSFR.

A.5.2.3 FUNCTIONAL AND OPERATIONAL AUDIT

This is held prior to software delivery to verify that all RSFR requirements have been met, and that the software and its documentation are internally consistent.

SECTION 6 - CONFIGURATION MANAGEMENT

This section shall document the methods to be used for identifying the software product items, controlling and implementing changes, and recording and reporting status. This information may be provided by reference to an existing configuration management plan.

SECTION 7 - PROBLEM REPORTING AND CORRECTIVE ACTION

This section shall describe procedures to be followed in reporting, tracking and resolving problems, and identify organizational responsibility for their implementation.

Buckley (8) includes an additional four sections

i) Tools, technicques and methodologies
ii) Code control
iii) Media control
iv) Supplier control

In the author's QA plan, (i),(ii), and (iii) would be included in section 4, viz Standards, practices and conventions, while supplier control is unnecessary at this stage since no sub-contracting is conducted on IS projects.

APPENDIX B

STEPS TO JOB ENRICHMENT

The objective of a job enrichment program is to improve job satisfaction and staff motivation, the effective utilization of personnel, and to stabilize the work force. This is achieved through manipulation of responsibilities, and of the motivating factors of jobs in the organizational hierarchy.

Job enrichment is not a once-off exercise, but rather a continuous management function. However, changes should last for a long period of time, for the following reasons ;

- changes should lift the job content to provide a level of challenge commensurate with the skill of incumbent.
- employees who have still greater ability will, in time, demonstrate it and win promotion to higher-level jobs.
- motivators, by nature, have a much longer-term effect on employees' attitudes than do hygiene factors. In time, jobs may need to be enriched again, but this will not occur as frequently as the need for hygiene.

The steps to job enrichment outlined below were suggested by Herzberg (23).

i) Jobs should be selected in which a) attitudes are poor, b) hygiene is becoming costly, c) motivation will make a difference in performance, and d) the investment in equipment or procedural analyses does not make changes too costly.

ii) Managers must be convinced that these jobs can be changed. Frequently managers consider the content of jobs to be "sacrosanct" and that the only scope for action is in ways of stimulating staff.

iii) "Brainstorm" a list of changes that may enrich the jobs, without concern for their practicality.

iv) Eliminate suggestions that involve hygiene, rather than motivation.

v) Eliminate generalities, suggestions must be specific.

Suggestions to provide greater "growth", "challenge" or "responsibility" are too general for the practical enrichment of job content.

vi) Eliminate suggestions which simply enlarge a job, i.e. simply increase the work load, since these merely increase the meaninglessness of the job.

vii) Direct participation of employees whose jobs are to be enriched should be avoided. Direct participation tends to contaminate discussions with excessive hygiene. Participating in the discussion gives the employee only a temporary sense of contributing, this sense can deteriorate dramatically once the process of setting up the job has been completed. It is the content of the job, i.e. what the employee will be doing once the setting up process has been completed, which will determine the employee's continued motivation.

viii) A controlled experiment can provide useful insight into the effects of the enrichment program. At least two equivalent groups must be identified. For the first, no changes are made while for the second, motivations are systematically introduced over a period of time. Pre-and post- installation tests of performance and job attitudes allow evaluation of the effectiveness of the job-enrichment program. It should be noted that initially a temporary reduction in performance (efficiency) of the experimental groups is frequently experienced. Performance improves with experience in the new job.

Anxiety and hostility over the changes being introduced are often experienced by first line supervisors. Anxiety results from a concern that the changes will reduce performance of their unit, while hostility can arise when the subordinate is expected to assume responsibilities which the supervisor previously considered his own. However, after a successful experiment, the supervisor often finds supervisory and managerial functions which were previously neglected because his time was expended in performing subordinate duties.

REFERENCES

1. BARTOL K.M., "Professionalism as a Predictor of Organizational Commitment, Role Stress, and Turnover: A Multidimensional Approach", Academy of Management Journal, 22, 815-821 (Dec 1979)

2. BARTOL K.M. and MARTIN D.C.,"Managing Information Systems Personnel: A Review of the Literature and Managerial Implications", MIS Quarterly/Special Issue, 49-70 (1982)

3. BELADY L.A.,"Evolved Software for the 80's", Computer, IEEE, 79-82 (Feb 1979)

4. BERSOFF E.H., HENDERSON V.D. and STIEGEL S.G., "Software Configuration Management; A tutorial", Computer, IEEE, 6-14 (Jan 1979)

5. BOEHM B.W. "Software and its Impact : A Quantitative Assessment," Datamation, 19, 48-59 (May 1973)

6. BROOKS F.P., "The Mythical Man-month", Datamation, 45-52 (Dec 1974)

7. BRANDON D.H., Management Standards for Data Processing Princeton, New Jersey, Toronto, (1963)

8. BUCKLEY F. "A Standard for Software Quality Assurance Plans", Computer, IEEE, 43-49 (Aug 1979)

9. CAVE W.C. and SALISBURY A.B., "Controlling the Software life-cycle : The Project Management Task", IEEE Transaction on Software Engineering, SE 4, 326-334 (July 1978)

10. COOPER J.D. "Corporate Level Software Management," IEEE Transactions on Software Engineering, SE-4, 319-326 (July 1978)

11 COUGER J.D., and ZAWACKI R.A., Motivating and Managing Computer Personnel, John Wiley and Sons, Inc., New York, Chichester, Brisbane, Toronto,(1980)

11a ibid., 3

11b ibid., 20

11c ibid., 38

11d ibid., 5

11e ibid., 27

11f ibid., 12

11g ibid., 20

11h ibid., 113-127

11i ibid., 115

11j ibid., 119

12. DALY E.B., "Management of Software Development", IEEE Transactions on Software Engineering, SE-3, 229-242 (May 1977)

13. DE ROZE B.C. and NYMAN T.H. "The Software Life-cycle - A Management and Technological Challenge in the Department of Defense", IEEE Transactions on Software Engineering. SE-4, 309-318, (July 1978)

14. DICKSON G.W., and POWERS R.F., "MIS Project Management Myths, Opinions and Reality", in McFarlan F.W. Nolan R.L., and Norton D.P., Information Systems Administration, Holt, Rinehard and Winston, Inc., New York, (1973)

15. DRUCKER P.R., Management : Tasks, Responsibilities and Practices, Harper and Row Publishers, New York, (1974) referred to in Jensen R.W., and Tonies C.C., Software Engineering, Prentice-Hall, Inc., New Jersey, 1979)

16. DUFFY N.M. and ASSAD M.G., Information Management, An Executive Approach, Oxford University Press,Cape Town,(1980)

16a ibid., 155

17. FORRESTER M.H. "Quality Control in Software", Quality Assurance,6, 99-104, (Dec 1980)

18. FOSTER R.A., Introduction to Software Quality Assurance, 4th Ed., Richard A. Forster P.E., San Antionio, Texas, (1980)

19. GLASS R.L. and NOISEUX R.A., Software Maintenance Guidebook, Prentice-Hall Inc, Englewood Cliffs, New Jersey, (1981)

20. GOLDEN J.R., MUELLER J.R. and ANSELM B., "Software Cost Estimating : Craft or Witchcraft", Data Base, 12-15,(Spring 1981)

21. GREENE C.N., "The Satisfaction-Performance Controversy", Business Horizons,31-41, (Oct 1972)

22. HACKMAN J.R., OLDHAM G., JANSON R. and PURDY K., "A New Strategy for Job Enrichment", California Management Review, 17, 57-71, (1975)

23. HERZBERG F., "One More Time: How do You Motivate Employees", Harvard Business Review, 53-62, (Jan-Feb 1968)

24. HOOGENDOORN K., "Quality in the Software Development cycle", Systems Magazine, 30-37, (July 1983)

25. JENSEN R.W. and TONIES C.C., Software Engineering, Prentice-Hall Inc., Englewood Cliffs, New Jersey,(1979).

25a ibid., 2

25b ibid., 3

25c ibid., 4

25d ibid., 329

25e ibid., 40

25f ibid., 334

25g ibid., 335

26. LA BELLE C.D. SHAW K., and HELLENACK L.J. "Solving the Turnover Problem", Datamation, 26, 144-152, (Apr 1980)

27. LEHMAN J.H., "How Software Projects are Really Managed", Datamation, 119-129, (Jan 1979)

28. LIENTZ B. and SWANSON E.B., Software Maintenance Management, Addison - Wesley Publishing Company; Reading, Massachussets; Menlo Park, California; Don-Mills, Ontario; (1980)

28a ibid., 73

29. LOFTIN R.D. and MOOSBRUKER J.M., "Organization Development Methods in the Management of the Information System Function", MIS Quarterly, 15-28, (Sept 1982)

30. McLAUGHLIN R.A.,"That Old Bugaboo, Turnover", Datamation, 25, 97-101, (Oct 1979)

31. MORIN L.H., "Estimation of Resources for Computer Programming Projects", M.Sc. Thesis in Putnam L.H., "A General Empirical solution to the Macro Software Sizing and Estimating Problem", IEEE Transaction on Software Engineering, SE-4, 345-360, (July 1978)

32. NORDEN P.V. "Useful Tools for Project Management", in Starr, M.K., (Ed). Management of Production, Penguin, Baltimore M.D., (1970), p71-101

33. PARR F.N., "An Alternative to the Rayleigh Curve Model for Software Development Effort", IEEE Transactions on Software Engineering, SE-6, 291-296 (May 1980)

34. PUTNAM L.H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem", IEEE Transactions on Software Engineering, SE-4, 345-361 (July 1978)

35. PUTNAM L.H., and FITZSIMMONS A., "Estimating Software Costs", Datamation, 189-198 (Sept 1979)

36. PUTNAM L.H., and FITZSIMMONS A., "Estimating Software Costs", Datamation, 171-178 (Oct 1979)

37. PUTNAM L.H., and FITZSIMMONS A., "Estimating Software Costs", Datamation, 137-140 (Nov 1979)

38. ROSENGARD P., "Software Quality Assurance Standards Under Development", Computer, IEEE, 84-85 (Feb 1979)

39. SCHOLL R.W., "Career Lines and Employment Stability", Academy of Management Journal, 26, 86-103 (Mar 1983)

40. SCHUTTE F.G., Integrated Management Systems, Butterworths, Durban-Pretoria, (1981)

41. SORKOWITZ A.R., "Certification Testing: A Procedure to Improve the Quality of Software Testing", Computer, IEEE, 20-24 (Aug 1979)

42. STEERS R.M. and MOWDAY R.T., "The Motivational Properties of Tasks", Academy of Management Review, 645-658 (Oct 1977)