MEASURING THE EFFICIENCY OF SOFTWARE DEVELOPMENT IN A DATA PROCESSING ENVIRONMENT

by

KLAAS GOVERT VAN DER POEL

Thesis submitted for the degree of Master of Science at the University of Cape Town.

Supervisor: Professor S.R. Schach

June 1982

ABSTRACT

The development of software for data processing systems has, during the last 25 years, grown into a large industry. Thus the efficiency of the software development process is of major importance. It is indicative of the level of understanding of this activity that no generally accepted measure of the efficiency of software development currently exists. The purpose of this study is to derive such a measure from a set of principles, to determine criteria for the acceptability of this measure, to test it according to the criteria set, and to describe inefficiencies obtained in a number of software projects.

The definition of data processing software is based on the concepts of Management Information Systems. Flows, files and processes are identified as the main structural elements of such systems. A model of the software development life cycle describes these elements in detail and identifies the main resources required.

A review of the literature shows that lines of code per programmer man-month is commonly proposed as a measure of efficiency of software development, but this measure is generally found to be inaccurate. In defining efficiency as the ratio of the prescribed results of a process divided by the total resources absorbed, a number of desirable properties of a practical measure of efficiency of software development are then put forward. Based on these properties a specific model is proposed which consists of the sum of flows, files and processes, divided by total project costs. Various other models are also considered.

Validity and reliability are identified as the most important criteria for the acceptability of the proposed measure. Its reliability is tested in a separate

experiment and found to be adequate. A field survey is set up to collect data to test its validity. The survey design chosen is a purposive sample of twenty software development projects.

The main result of the survey is that the proposed model of efficiency is found to be valid. Other models investigated are less attractive. Efficiencies achieved in the twenty projects included in the sample are found to differ substantially from one another.

Apart from achieving its specific objectives, the study also provides a perspective on some of the problems of software development. Several subjects for related research are identified.

ACKNOWLEDGEMENTS

During the long and tortuous gestation period of this thesis, I received support and encouragement from many.

I specifically wish to thank the following:

Professor J.H.F. Meyer and Dr. M.L. Hart who advised me in the early stages.

The MBA students of the Graduate Schools of Business of the Universities of Cape Town and Stellenbosch, who helped crystalise some of the ideas.

The participants in the survey whose interest and co-operation were an important source of motivation.

I am grateful to Professor J.S. de Wet for listening to me when I was in trouble. Most of all, I am indebted to Professor S.R. Schach, who encouraged me to pick up the pieces and coached me through the difficult second start. Without him this work would not have been.

Finally, the proof-reading and typing were done professionally by Mrs. G. Wilkin and Mrs. J. Muller.

CONTENTS

LIST OF EXHIBITS

6.9 Correlation between lines of source code and functions of flows, files and processes.

6.10 Distribution of observed efficiencies.

6.11 Average efficiencies per sector of industry.

CHAPTER 1

INTRODUCTION

1.1 Background of the Study

1.2 Objectives

1.3 Scope and Limitations

1.4 Structure

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND OF THE STUDY

Ancient empires are often characterised by their achievements in communication and information systems: highly developed communication systems supported the Persian empire, the Roman empire and the Inca empire, while the ability to record and preserve information was an important feature of the Egyptian and Babylonian cultures. The introduction of more efficient ways of handling information has, on several occasions, coincided with the rise of civilisations: the use of the alphabet with the Greek civilisation, the invention of the printing press with the Renaissance, the telegraph and telephone with the Industrial Revolution. The second half of the twentieth century has seen numerous innovations in information handling. Marshall McLuhan indicated at an early stage the importance of information handling technology in modern society:

> "The mere interrelation of people by selected information is the principle source of wealth in the electric age"(1).

Information-handling capabilities, in fact, increased so dramatically during the last twenty-five years that it has become acceptable to use the term "information revolution". Auerbach(2) has shown that memory capacity, calculating speed of computers and numbers of computers installed have increased exponentially, while production costs of this equipment have decreased exponentially.

The degree to which modern society will benefit from this increase in information handling technology depends to a large extent on the efficiency with which the new technology

will be applied. Therefore, this study is specifically concerned with the efficiency of certain information handling activities.

Efficiency implies the minimal use of resources to achieve a given objective. This aspect is obviously more pressing as the level of resources involved is more significant and as these resources may be used in a greater variety of ways. These two conditions are believed to hold for information handling in general and for certain data processing activities in particular. The resources involved in all aspects of information handling are very significant indeed. Strassman[3] has estimated that about $400 billion was spent on information handling in the U.S.A. in 1973. Even small gains in efficiency in this activity will therefore have significant economic and social consequences. The same study indicates that a significant part of the above-mentioned amount, namely $26 billion, was spent on what is called electronic data processing. This term covers the use of computers and related equipment for administrative purposes. The activity will be called data processing in this study and a more precise definition is given in Chapter 2. Though data processing clearly only deals with part of all information handling, its social and economic impact is significant and likely to grow with further economic development as illustrated by the comparison between South Africa and an economically developed country in Exhibit 1.1 overleaf.

Data processing had its beginning as late as 1954 with the computerisation of a payroll system[5] and with fewer than 30 years of experience it is not surprising that practices and techniques have not yet converged to a common point. Adequate standards of quality and efficiency of data processing systems have proved elusive and, as a result, practitioners often follow their intuition in the selection

Comparison of the Size of the Data Processing Industry in South Africa and the Netherlands in 1977

	Population (million)	GNP (R million)	Employment in DP	Computers installed	Equipment value (R million)	Annual Expenditure (R million)
South Africa	24	33	17 000	1 200	500	440
Netherlands	15	95	60 000	3 000	1 480	1 380

Source: See reference (4)

EXHIBIT 1.1

of appropriate techniques. This is another reason for aiming this study at the definition and measurement of the efficiency of certain aspects of data processing.

Kanter[6] lists 26 specific problems in data processing that are of both academic and practical interest. Eleven of these refer to data processing software development, which is an indication of the scope for improvement of practices and therefore of efficiency in that area. In their book "Data Processing in 1980-1985" Dolotta et. al.[7] analyse the trends and problems expected in data processing. They conclude that

> "over the next decade, the major tasks of the data processing industry will be to improve:
> - the quality of data processing services as perceived by the end users of these services
> - the resulting productivity of these end users
> - the productivity of data processing systems and of data processing applications development efforts."(8)

The third point refers to efficiency and more specifically to the efficiency of data processing applications development. This acitivity is of particular importance since at that stage the foundations are laid for subsequent efficiency and effectiveness of applications. The focus of this study is therefore not on computer hardware or its operation but on what, in other contexts, is often called software and more particularly on the creation of software through the process of software development. A more precise definition of these terms will be given later.

Boehm has estimated[9] that in 1976 $ 20 billion was spent on software development in the U.S.A. Other investigations[10] have indicated that software development accounts for approximately 30% of all data processing expenditure. Using

the data of Exhibit 1.1, this would bring the estimated expenditure for South Africa to approximately R 150 million per annum. Yet, software development has been called "a craft or a cottage industry" and this situation is ascribed to "a lack of coupling between the research community and the practical world of applications"[11]. In particular, there appears to exist a lack of reliable and empirically tested units of measure to determine such important parameters as magnitude, quality, reliability and efficiency, in the area of software. As pointed out by Lord Kelvin

> "When you can measure what you are speaking about, and express it in numbers you know something about it; but when you cannot measure it, when you cannot express it in numbers your knowledge is of a meagre and unsatisfactory kind. It may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of science"(12)

This study will therefore attempt to make a contribution to academic knowledge as well as to the data processing industry by defining certain new units of measure in this area and evaluating their validity and applicability.

Not only have few measures been agreed upon in data processing, but elementary information about data processing activities of organisations is not readily available. Particularly in South Africa, no survey is known to exist of efficiencies achieved in software development. By reviewing the situation and pointing to certain discrepancies, this study aims to make a contribution to the insight in the software development process.

1.2 OBJECTIVES

Based on a general understanding of the problems described in the previous section, the specific objectives of this study are defined as follows:

of these kinds of software are considered to be sufficiently different to require separate treatment.

In the interest of defining a homogenous subset of comparable software three further limitations will be made:

- Only systems that are fully implemented and are working to specification will be considered. This excludes systems that are still under development, of which normally neither the extent nor the costs are completely defined.

- The study will be limited to systems produced by organisations in similar phases of their date processing life cycle. The importance of the life-cycle of data processing experience has been underlined by Nolan and Gibson[14], who distinguish four phases, namely: Initiation, Expansion, Formalisation and Maturity. These phases are characterised by the type of applications dealt with, the degree of specialisation of the data processing staff and the degree of management control and involvement. Indications are that in the Initiation phase, practices and procedures are often not well considered and concern is mainly with acceptance of the data processing effort. Consequently efficiency of development is likely to be erratic. Conversely, in the maturity phase, there is likely to be a pre-occupation with organisational issues and efficiency may be of secondary importance. The practical part of this study will therefore be limited to software developed by organisations in the phases 2 or 3 of data processing experience. This will contribute to the homogeneity of the practices to be considered.

- One of the most significant events of data processing is the emergence of the data base concept. The term "data base" describes a concept which may have different meaning to different persons:

> "Administrations and managers look upon the aggregate of data from which budgets and decisions are made and call it, nebulous as that term may be, the data base. To technicians of data base systems, a data base is a collection of physical records that are similarly defined and serve a single general application purpose".(15)

The data base concept is usually connected with certain objectives or characteristics that are specified for the data base, such as improved consistency and reduced redundancy. To achieve objectives, special structures and interlinkages are required that are normally entrusted to a software package called a Data Base Management System (DBMS). Through the use of a DBMS it becomes possible to employ quite different software development techniques and notably uncouple the data (the files) from the rest of the system (the processes). This makes it tenuous to attempt to define concepts and practices covering both conventional software development and DBMS-oriented development. It has therefore been decided to exclude fully-fledged data base systems (i.e. systems making use of a shared data base maintained by a DBMS) from the practical part of this study. This does not imply conceptual differences, but is done to ensure more compact and comprehensible definitions and to avoid confusion.

At a later stage it should be eminently feasible to extend this study to include the development of data base oriented systems. The above limitations define a subset of application software in a data processing environment. This is not considered to detract materially from the scope of the study, as the vast majority of data processing systems fall within the subset.

A further limitation needs to be made. As indicated in a recent article[16] there exists a growing tendency to acquire the software needed to support information systems

in the form of 'packages' rather than through in-house development. The implementation of packaged software is not trivial, but is clearly very different from the development of tailor-made systems.[17] Though the use of packages has obvious attractions, it is also limited by certain restrictions. It may notably lead to unacceptable risks, while the managerial style and strategy of the organisation may also require specific consideration.[18] Chandler, furthermore, has pointed to the evolving requirements as the organisation progresses through its phases of growth. Notwithstanding the advantages of packages, many organisations will therefore continue to develop special software and this is the activity on which this study will focus.

The study will furthermore be limited to South African organisations. This is not considered to be a severe limitation as it is known from personal experience and from the literature, that practices in South Africa and abroad do not differ drastically. The study will encompass all types of organisations that develop data processing software. This includes commercial and industrial organisations as well as government institutions and professional service organisations. It will be of interest to determine to what extent practices and results differ amongst these groups.

Field research to test various theories and gather data about software development results and practices will be based on a non-random but purposive sample, properly stratified to reflect relevant characteristics of the underlying population. It is realised that this method of sampling will not strictly allow extrapolation of the results to the population, but this research methodology is generally considered[20] to provide the most valuable insights in an exploratory research situation like this study.

1.4 STRUCTURE

This study is based on a field survey of a number of software development projects in a data processing environment. The results are structured as follows:

Chapter 2 deals with the basic concepts of systems and information and data processing. Important terms, notably data processing and software are defined, and flows, files and processes are introduced as the conceptual elements of a data processing system. This life cycle of the software development process is presented, and the inputs used and outputs generated by this process are defined.

Chapter 3 defines the concept of efficiency and applies this to software development. The problem of establishing effectiveness and other quality aspects of systems is recognised. A measure of efficiency of software development is proposed, based on a number of well defined principles.

Chapter 4 is concerned with ways and means of establishing the acceptability of the proposed measure of efficiency. Certain criteria are first developed in general. A test of reliability is defined and the performance and outcome of the test are described. Tests of validity are proposed and a field study to perform such tests is defined.

Chapter 5 sets out the methodology for field research. As mentioned before, a choice is made of a purposive, stratified sample. The principles underlying the choice of respondents, the construction of the questionnaire and the organisation of the survey are specified.

Chapter 6 gives the results of the field survey. The conclusion regarding the validity of the unit of measure of efficiency is discussed and other findings are analysed and presented.

Chapter 7 summarises the conclusions and provides a perspective on certain data processing problems. Topics for further research are also suggested.

REFERENCES - Chapter 1

1. McLuhan, M.: Understanding media : the extensions of man, Routledge and Kegan Paul Ltd., London, 1964, p.263.

2. Auerbach, I.: The information revolution - will it improve the quality of life, Data Processing manual - management planning, Auerbach publishers, Philadelphia, 1975.

3. Strassman, P.S.: Managing the costs of information, Harvard Business Review, Sept. - Oct. 1976, p.134.

4. For compilation of this table use was made of:

   - Finlayson, R.L. (ed) : Computer users handbook 1977/1978, Systems publishers, Johannesburg, 1978.

   - Department of Statistics : Computer survey 1977, Government printer, Pretoria, 1978.

   - Hammink, R.J.: Administrative automisering in Nederland 1976-1981, Studiecentrum Novi, Amsterdam, 1979.

   - Orpen, S.: The chip revolution, Management, December, 1979.

   - Corson, M.: A comparative analysis of computer suppliers in South Africa, Unpublished technical report, The Graduate School of Business, Cape Town, 1976.

5. WU, M.: An introduction to computer data processing, Harcourt Brace Jovanitch, New York, 1975, p.9.

6. Kanter, J.: Management oriented management information systems, Prentice Hall Inc., Englewood Cliffs, N.J., 1977, p.309.

7. Dolotta, T.A. et. al.: Data Processing in 1980-1985, John Wiley and sons, New York, 1976.

8. Ibid., p.9.

9. Boehm, B.: Software engineering, IEEE transactions on computers, December 1976, p.1226.

10. E.g.:

- Hammink, R.J. : op. cit. - p.34.

- Davis, G.B. : Management Information Systems: conceptual foundations, structure and development, McGraw Hill Book Company, New York, 1974 - p.390.

11. Dolotta, T.A. et. al.: op. cit., p.24.

12. Quoted by : Dolotta, T.A. et. al.: op. cit. - p.101.

13. Alter, G.L.: Decision support systems, current practice and continuing challenge, Addison Wesley Publishing Company, Reading, Mass, 1980.

14. Nolan, R.L., Gibson, C.F.: Managing the four stages of EDP growth, Harvard Business Review, Jan-Feb. 1974.

15. Ross, R.G.: Data base systems, Amacom, New York, 1978, p.3.

16. Van der Poel, K.G.: Programmatuur - maken of kopen, Informatie, October, 1981.

17. Van der Poel, K.G.: Ibid., p.347.

18. Antony, R.N., Dearden, J., Vancil, R.F.: Management Control Systems, Richard D. Irwin, Homewood, 1972, p.395.

19. Chandler, A.D.: Strategy and structure, The MIT press, Cambridge, Mass, 1962.

20. Selltiz, C. et al.: Research methods in social relations, Holt Rinehart Winston, New York, 1967.

CHAPTER 2

SOFTWARE DEVELOPMENT IN A DATA PROCESSING ENVIRONMENT

CHAPTER 2

SOFTWARE DEVELOPMENT IN A DATA PROCESSING ENVIRONMENT

2.1 INTRODUCTION

The objective of this chapter is to define and describe the background of software and data processing systems. This will lay the foundation for the understanding of the efficiency of software development in a data processing environment, which is the subject of this study.

Data processing will be defined as a subset of a wider field called Management Information Systems. To understand the essence of data processing, it will therefore be useful to define and study the concepts underlying Management Information Systems. This will lead to an understanding of the main functions and structural elements of Management Information Systems, which can be extrapolated to data processing systems.

Data processing systems are the result of the process called software development. This process can best be understood by means of a model. A suitable model is the life cycle model which divides software development into a number of stages. The resources absorbed and results produced by each of these stages can be enumerated, which will help in the construction of a measure of efficiency. Detailed definitions will be given of the concepts needed to define efficiency.

2.2 MANAGEMENT INFORMATION SYSTEMS

Computers and data processing have no justification in

themselves, but derive their utility from their ability to provide information for decision making. Decision making is also the essential function of management. Management in its widest sense can again be equated with the control of systems. The concepts underlying systems and their control are bundled in what is called "systems theory", which has been developed by Wiener(1), Boulding(2), Ackoff(3), Churchman(4) and others. This theory provides a coherent framework in which relevant terms are defined. It is therefore a useful tool and will be used as such without necessarily subscribing to the ambitious claims that are made for systems theory by writers such as von Bertalanffy(5) and Beer(6).

A system is defined in systems theory as: "an entity which is composed of at least two elements and a relation that holds between each of its elements and at least one other in the set".(7) The important part of this definition, the relation between the elements, is the reason why the concepts of information and communication have always been very important in systems theory. The relation between the elements of a system is maintained by means of communication. In systems terminology, communication is nothing more than the transmission of information.

In the study of systems it is important to distinguish between a system and its environment. The environment is everything that does not belong to the system, but a change in it may produce a change in the system. Determination of what belongs to the system and what belongs to the environment is a problem to which there is no universal answer. The definition of the boundaries of the system depends on the purpose of the investigation and the delineation of the system therefore depends on the point of view of the observer.

A system which interacts with its environment is called an

open system by Ackoff[8]. In such systems disturbances from the environment (D) will affect the results (R) of the system (S). The basic model of a system is represented by these three elements in Exhibit 2.1. The more complex systems which are generally of interest in the social sciences are called "goal-seeking" systems, defined by Ackoff as: "a system that can respond differently to one or more different external or internal events ... until it produces a particular state. Production of that state is its goal".[9] In other words, the system has the ability to reduce the variety of the results to a preferred set (T). This capability presupposes the presence of a control mechanism (C) which interacts with the system, the environment and the results.

The total situation is described in Exhibit 2.1 where I stands for the input and O stands for output.

A CONTROLLED SYSTEM

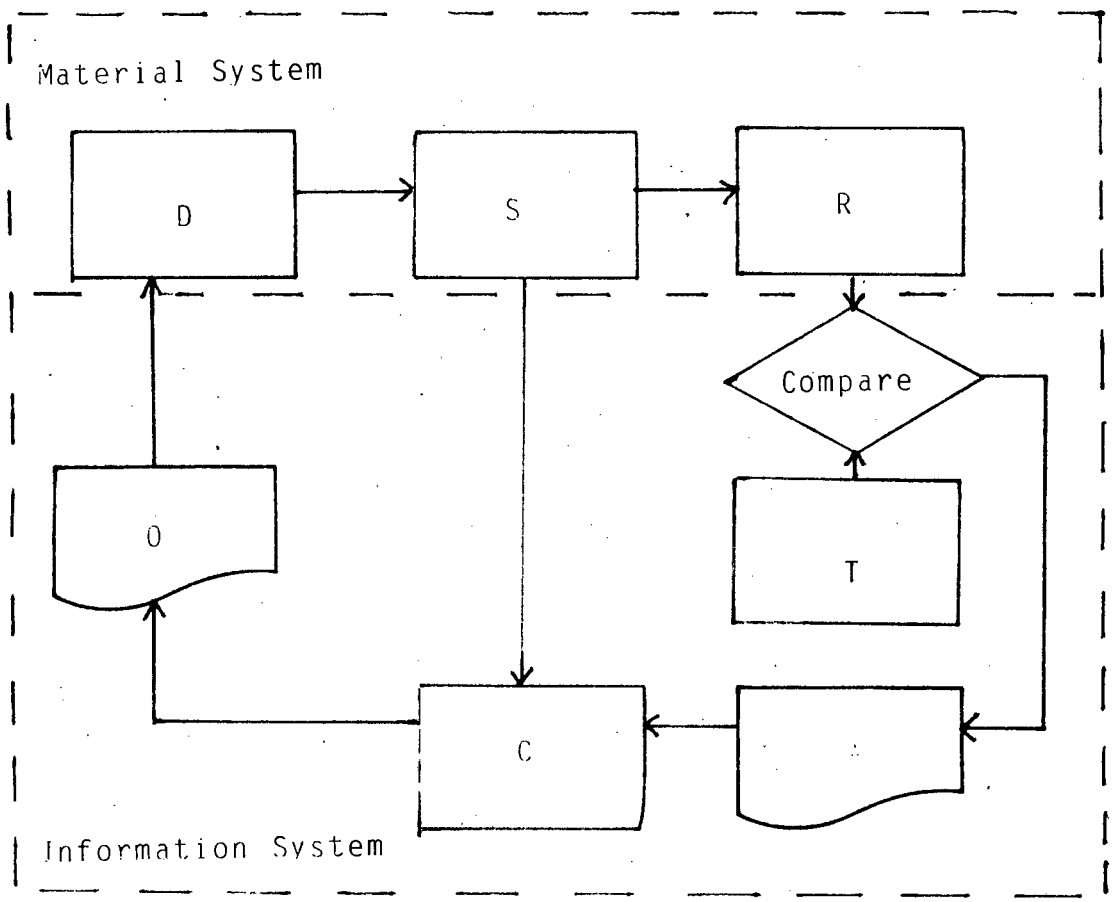

EXHIBIT 2.1

To fulfil its function, the control mechanism must be capable of performing the following tasks:

- Store information about target results;
- Transmit information about the environment and the status of the system to the control mechanism and from the control mechanism back to the system;
- Process received information and stored information to obtain control information.

It is clear from the above that all actions of the control subsystem involve information. The control subsystem may therefore be equated with an information system. The upper part of Exhibit 2.1 which has been labelled the "material system" has been more fully described by Bosch[10] as the material-energetic-financial system (MEFS). The equating of the control system with an information system is important because it makes it clear that storing, transmitting and processing, which are the main tasks of a control system, are also the main tasks of an information system. Starreveld[11] shows that his categorisation of the tasks of an information system leads logically to the recognition of three and only three structural elements of an information system. Each of these has a direct relationship, with the tasks of the system, as follows:

storing	-	files
transmitting	-	flows
processing	-	processes

Consequently, information and system theory provide a strong basis for the identification of flows, files and processes as the main structural elements of an information system. The above analysis will be used extensively in this study.

Block S was described in Exhibit 2.1 as the "system".

It will, however, be clear that from a different point of view the system should be defined to include the block C or all of the information system. Taking this point of view will lead to the consideration of more complex forms of systems which are typical of open, dynamic, purposeful systems such as business organisations. The most important characteristics of such systems are, according to Ackoff[12]:

- The ability to alter or adapt their own goals (purposeful behaviour). This may occur when the original goals cannot be obtained, or have been obtained already, or when it becomes clear that other demands are being made on the system.

- The ability to alter their own internal structure. This is a typical attribute of social systems and is called system designing behaviour.

- The ability to influence their environment. This usually involves reducing the uncertainty (or variety) of the environment.

When these capabilities are added to the control function (C) in our diagram then this function can truly be equated with what in an organisation is called management and the information system becomes a Management Information System depicted in Exhibit 2.2 overleaf.

The above analysis underlines the important link between management and information. On the one hand management is activated by the information it receives from the material system and exerts its influece by disseminating information. On the other hand the information system is completely geared to the management function and information has no value except to the extent that it serves the management function.

HIGHER LEVELS OF CONTROL IN A SYSTEM

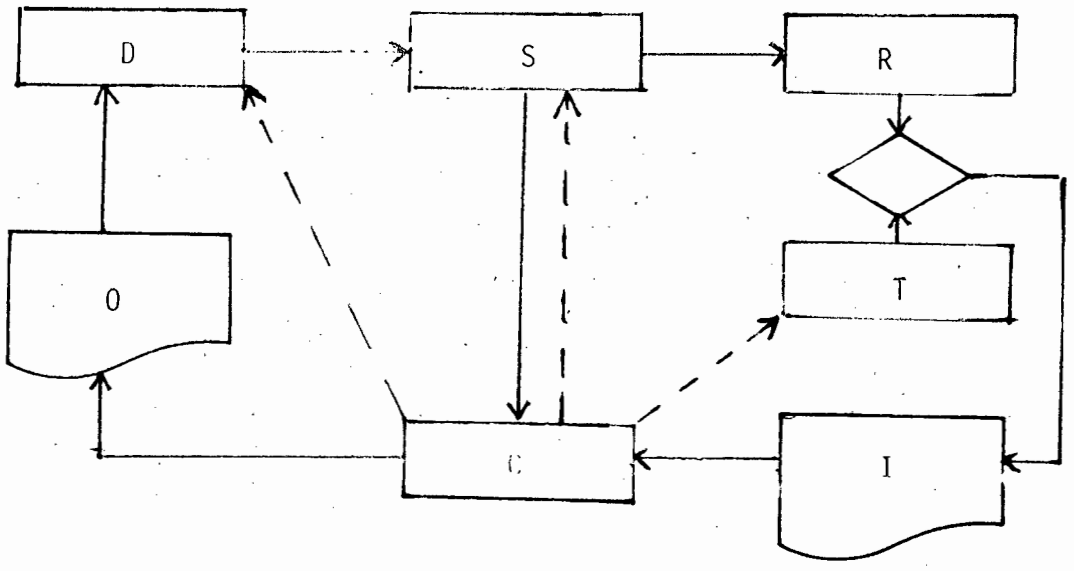

EXHIBIT 2.2

The above considerations concerning information, systems and management should contribute to a clearer perception of Management Information Systems and their constituent parts. These considerations are more or less incorporated in the following typical definitions of Management Information Systems:

> "A Management Information System is an integrated man/machine system for providing information to support the operations, management and decision making functions in an organisation".(13)

> "A Management Information System is an organised method of providing relevant data for decision making to reponsible persons in the organisation.(14)

These definitions will be used as a point of reference for this study.

TYPES OF MANAGEMENT INFORMATION SYSTEMS

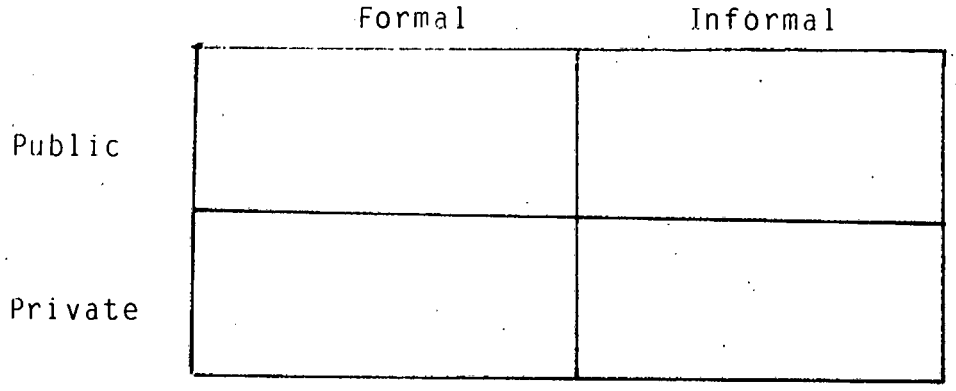

EXHIBIT 2.3

Data processing will be considered to be restricted to the formal, public part of Management Information Systems and will therefore exclude private and informal information.

Secondly, Management Information Systems have been divided into various levels. Antony[26] distinguished systems for Strategic Planning, Management Control and Operational Control. This distinction has been taken over by many authors[27] and has been used as a basis to define various type of systems. Information systems for Strategic Planning have been called Decision Support Systems and have been shown to have very specific characteristics. These systems are therefore not included in the definition of data processing. Head[28] has extended Antony's levels by introducing a fourth level - Operational Support Systems - as shown in Exhibit 2.4. Operational Support Systems include the use of data for the support of such activities as technical calculations, engineering design and process control. These applications will not be included in the definition of data processing as shown in Exhibit 2.4.

Finally data processing will be restricted to computerised activities. This does not suggest a conceptual distinction, but is useful to focus the attention on software as previously defined.

- The arithmetic unit and control unit are closely interlinked and together perform the processing function.

- Certain input and output devices are connected to media for storing data. These, together with the primary storage unit, provide a vast and readily accessible storage capacity.

- Other input and output devices such as the console particularly, enable the flow of information between the computer and its environment.

It is interesting to see the structural elements of an information system reflected in the architecture of the computer.

There is a widely held belief that computers have not yet reached their maximum usefulness in information systems. Their application is, however, not restrained by a lack of hardware capabilities, as expressed by Dolotta et. al.[18]

> "Finally, we repeat that the lack of hardware capabilities is not the major constraining influence on the growth of our industry".

This opinion in influenced by the expectation of further improvements in hardware capacity and availability. Several authors, including Duffy[19] and Dolotta et. al.[20] expect that the major constraints on the application of computers to information systems will not be hardware problems, but rather be software, management and organisational matters.

Software is a relatively new concept, introduced with the advent of computers. Its definition varies from "everything that gives life to computer hardware"[21] to "a collection of computers programs".[22] Davis[23] gives the following

definition: "Software consists of computer programs and routines which direct or facilitate the operation of the computer". In this study the term software will be given a similar, rather wide interpretation. It will include programs written in a programming language, but also the instructions on how to use these programs, the necessary data and files, as well as the essential documentation. Software development is therefore not only the coding of new programs but also the organisation, implementation, printing, binding, filing etc. which is necessary to create what will be defined in the following paragraphs as data processing systems.

2.3.2 Data Processing

Having dealt with the principle of Management Information Systems and computers, the foundation has been laid for a definition of data processing. The literature is unfortunately lacking in this respect, as several well-known textbooks[24] that have the word data processing in their title only present very loose definitions of the term. It is, however, clear that most modern writers virtually equate data processing with the use of computers in Management Information Systems. The same area of activity has also been described as Automated Data Processing (ADP), Electronic Data Processing (EDP) of Computer Data Processing, but there appears to be a convergence towards the use of the term data processing or dp.

For the purpose of this study data processing will be defined as a specific subset of Management Information Systems. Therefore several distinctions are made. Firstly, Management Information Systems as previously defined, can be divided into four parts[25] as shown in Exhibit 2.3.

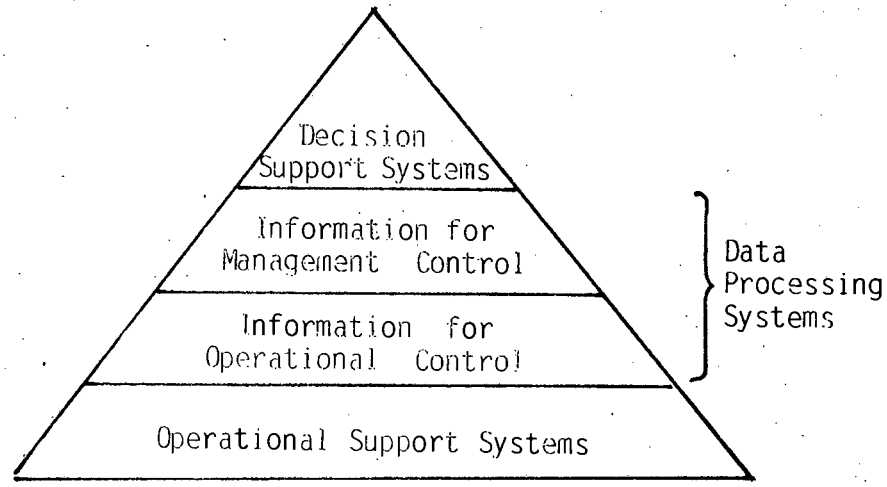

EXHIBIT 2.4

The above considerations lead to the adoption, for the purpose of this study, of the following definition of data processing: Data processing is the application of systematic methods and computer technology to perform the storage, transmission and processing functions in those parts of a Management Information System that deal with public, formal information and are directed at management control and operational control.

2.3.3 Identifying Data Processing Systems

From an overall management point of view, the data processing effort of an organisation may be viewed as a whole, but on closer inspection it is possible to distinguish the sub-systems of which it may be assumed to be built up. Such subsystems will be called data processing systems. The reasons why and ways in which a total information system can be subdivided into more manageable subsystems have been extensively discussed by Blumenthal,[29] based on an analysis by Forrester.[30]

Accordingly, data processing systems are defined as follows: A data processing system is a section of the data processing

software of an organisation which is directed at a significant, but reasonably self-contained, functional part of the activities. Examples of typical data processing systems in a business organisation are: the accounts receivable system, payroll system. inventory control system, production scheduling system.

2.3.4 The Structural Elements of Data Processing Systems

In paragraph 2.2.1 the structural elements of information systems were defined as flows, files and processes. As data processing systems form a subset of information systems with essentially the same goals and functions, they are also built up of the same elements. Starreveld[31] has corroborated this by pointing out that the operations that can be performed on data are essentially the same as those that can be performed on physical goods: they can be moved in time (stored), moved in space (transmitted) or reshaped (processed). A slightly more elaborate but similar analysis is presented by Davis[32] who described the functions of a data processing system as: processing transactions, maintaining history files, producing reports and interacting with the user. It is clear that this describes essentially the same functions of storing, transmitting and processing. The distinction of flows, files and processes is, however, preferred because of its theoretical foundation as well as its simplicity.

The establishment of the fact that a data processing system may be considered as a structure of flows, files and processes has two important consequences. In the first place, it dispels the commonly held notion that the understanding of such systems requires knowledge of electronic technology and should be left to computer experts. Ackoff brands this notion as a dangerous fallacy:

> "This leaves managers unable to evaluate the MIS as a whole. In failing to evaluate their MIS, managers delegate much of the control of the organisation to the system designer". (33)

Evaluation and control of an information system is made possible by understanding it and this, in turn, is facilitated by reducing it to its most essential elements - notably flows, files and processes. Data processing systems reduced to these elements can be readily understood by the interested layman.

Secondly, the realisation that the system consists of flows, files and process, eliminates the common preoccupation with only one of these elements, namely the processes or programs. Too often, systems are implicitly equated with programs, which in turn leads to an over-emphasis of the programming task, to the detriment of the other tasks involved in the development of systems.

Having developed the conceptual foundations of flows, files and processes as the structural elements of data processing systems, it remains to produce operational definitions of these concepts. This requires a good understanding of the practicalities of the data processing systems and their development. These practicalities will be discussed in the following paragraphs and operatioanl definitions are therefore given only in paragraph 2.4.3.

2.4 A MODEL OF SOFTWARE DEVELOPMENT

To describe the software development process, a model is commonly used which divides the total process into a number of stages. It is a source of confusion that a similar model is also often used to describe the total existence of a data processing system, including its development, operational

life and maintenance. The life cycle model to be presented in the following paragraphs will only relate to the development process and therefore specifically excludes the operational stage and any activities that take place during that stage. Also excluded from the life cycle model will be the general planning and analysis activities that precede the actual initiation of a specific development project.

This planning is obviously of crucial importance to the success of software development, but for details and opinions on the subject, reference is made to the extensive literature.[34]

Several alternative procedures for information systems planning are shown by Canning[35] but there is obviously a growing consensus that planning and control of information systems is a continuous function of general management, and that only limited inputs to this task are expected from data processing specialists. As a consequence, it is normally not possible to allocate the effort spent on systems planning to a particular system. Therefore the overall planning and initiation of systems (sometimes described as preliminary analysis) is considered in this study to be separate from and preceding the software development effort. It will therefore be excluded from further analysis.

2.4.1 The Life Cycle Model

The model most commonly used in the literature to describe software development is the life cycle model. In its most basic form, this model divides the software development process into a number of roughly consecutive stages, starting with the inception of the project and ending with its completion. In a graphical presentation, the various stages are typically shown on the horizontal axis and the expected level of resources required is shown on the vertical

axis, so that a picture results as in Exhibit 2.5 below.

The purpose of the life cycle model is to emphasise the logical progression of activities common to all software development projects and to provide a number of checkpoints

BASIC DIAGRAM OF THE SOFTWARE DEVELOPMENT LIFE CYCLE

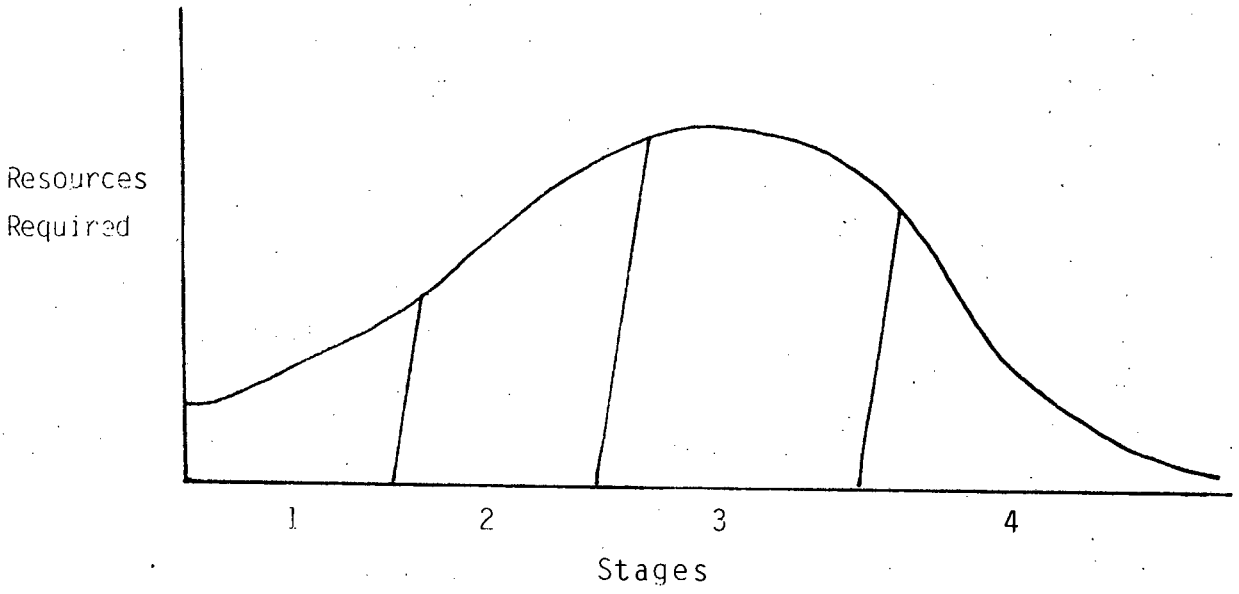

EXHIBIT 2.5

in the form of stage completions during the development of the system. This model is therefore useful for the management and control of software development activities. According to Benjamin

> "the definition of a useful number of well-defined workphases for the development cycle ... is actually the development manager's principal aid".(36)

Many authors commented on, or made contributions to the life cycle model. An early contribution was made by

Brandon,[37] while Hartman et. al.[38] provide a very comprehensive structure. Benjamin[39] has produced a survey of life cycle models proposed and terminology used by six other authors, while Murdick[40] presents a comparison of 17 authors. These comparisons show a high degree of confusion in terminology and a great deal of uncertainty as to which activities at the beginning and the end of the software development process should still considered part of it.

While some authors define planning as the first stage of the software development process, it has been argued before that overall systems planning is the responsibility of general management and is inseparably linked to the process of strategic business planning. Planning for a particular data processing system is not a separate stage, but an activity which takes place as long as the development process continues.

Bearing in mind that the purpose of an information system (and therefore of a data processing system) is to support decision making, a logical first step in software development is the definition of precisely what decisions need to be supported and what information is needed to do this. This is normally called requirements definition and a software development project is therefore deemed to have started when a decision is made to embark on this activity.

The development of a system is considered to be complete when the system is fully implemented, which is defined as the moment when the system satisfies the explicit and implicit user requirements. At that moment the system may already have been operated for a while and it is also likely that further alterations and maintenance will be required in future. Maintenance is generally recognised as an important (and cumbersome) stage in the total life of a

system but, as it occurs after implementation, it is not included in the development life cycle model. Some authors specify an evaluation and audit stage at the end of the life cycle. Such an activity is, however, not always performed, or only performed much later. Sulcas[41] has shown that evaluation and audit should normally be a task external to the software development effort and should be performed by a party such as the management accountant or a consultant or a multi-disciplinary team of users and management. This activity is therefore not included in the life cycle model.

Following the above considerations and insights gained from the literature as well as from personal experience, a life cycle model of software development is proposed, consisting of the following four stages:

- Requirements definition
- System design
- System construction and testing
- Documentation and implementation

Essentially similar models are described by Metzger,[42] Enger[43] and Davis.[44] Descriptions of these stages are given in the following paragraphs and are closely related to the definitions given by the above authors.

A. Requirements Definition

This stage may be equated with what other authors have called the analysis or systems analysis stage. The objective of this stage is to produce a complete and explicit specification of all aspects of the system that affect the prospective users. This includes the information to be produced by the system as well as the

inputs to be provided to the system. This stage begins with the initiation of the project and ends with the acceptance of the functional specification. Some of the tasks to be performed in the requirement definition stage are listed below (not necessarily in sequence):

- Analysis of the decisions to be supported
- Study of the existing system
- Definition of alternative ways to provide information to the decision makers
- Determination of the technical, economic and operational feasibility of the alternatives
- Preparation of a functional specification document.

The requirement definition stage culminates in the production of a document containing the functional specification. Guidelines for the contents of such a document are provided by Enger.[45]

B. System Design

While the requirement definition stage defines what the system will produce and require, the system design stage defines how it will work. This involves determining the structure of the system, notably defining the main structural elements: flows, files and process. This stage begins with the acceptance of the functional specification and ends with presentation of the technical specification. The design process has been extensively investigated by Freeman who describes it as:

> "one of those interesting human activities that is all around us, in which everyone engages in one form or another and that eludes precise characterisation because of its many forms and complexity".(46)

Various methods have been proposed to rationalise the design process[47] but according to Peters and Tripp[48] design, and specifically software design, may still be classified as a "wicked" problem because "it is not always clear how to proceed and the problem seems to change - often for the worse - during resolution". The main tasks to be performed in the system design stage are:

- Definition of file content and organisation
- Drawing up of system flow charts
- Specification of programs and procedures
- Design of control and security procedures
- Determination of procedures for testing and implementation
- Preparation of a technical specification document

The system design stage terminates with the production of the technical specification. Suggestions for the content of this document are listed by Enger.[49]

C. System Construction and Testing

System construction and testing begins after the acceptance of technical specification, and ends when all processes and files have been created and found to be correct. Testing of processes (programs) typically requires the availability of masterfiles and other permanent data sets. The construction of these elements often requires a substantial amount of effort

which is considered to be part of this stage.

The main tasks in this stage are:

- Specification and preparation of test data
- Coding, compiling and testing of programs
- Conversion of existing files and creation of new files
- Production of procedures and control statements.

These construction activities and particularly the coding and compilation task, form the nucleus of software development, but they are clearly not the only activities of importance. Results delivered in this stage are programs (both source and object codes), control procedures and permanent files.

D. Documentation and Implementation

In this stage the individual structural elements of the system are put together and the interfaces between the system and its users and operators are completed. This stage is conceived to begin when all structural elements are completed, and to end when the system has been shown to satisfy all the explicit and implicit user requirements and has been accepted as such. It may therefore be a protracted stage, involving extensive care and corrections. Activities in this stage are:

- Preparation of operator documentation
- Preparation of user documentation
- Training of users
- Performance of acceptance tests
- Fault correction and modification.

It is unlikely that there will be a sharp division between this stage and the preceding one and a

substantial overlap is more normal. For this reason, several authors (including Davis)[50] do not specify this stage. Another reason for not specifying this stage is put very strongly by Gildersleeve[51] who feels that documentation is an intergral part of the progamming activity. In this study, documentation and implementation are considered to be a separate stage because:

- the activities are crucial to the success of the system
- they are likely to be rather time-consuming
- they are often performed by specialists.

Deliverables in this stage are the external documentation, namely user manual and operations manual, as well as a more ethereal result: acceptance of the system by the users.

2.4.2 Resources Absorbed and Results Produced

In the previous paragraphs various resources absorbed and results produced by the software development process were mentioned. These will be examined more closely in this section, as they will form the ingredients for the determination of efficiency which will be defined in the next chapter as the ratio of results produced to resources absorbed.

A. Results Produced

This study is only concerned with the physical results of software development. Effects such as job satisfaction or a higher professional standing of the participants are not considered.

Progams are commonly considered to be the main or only product of software development. However, programs and procedures together constitute only the processes of a system and it is obvious that files and flows which are equally essential structural elements should also be taken into account. Flows and files are of course referenced in the programs, but they are conceptually different entities.

Apart from the structural elements of the system, documentation may also be considered to be a product of software development. It is, however, not clear whether this should be regarded as an independent entity or as dependent on the structural elements. Four different types of documentation have been mentioned before, which can be combined into two groups:

- Functional and technical specifications are necessary aids in the construction of the system. They may be regarded as scaffolding, essential for the construction of the system, but there is a great deal of merit in retaining the scaffolding for maintenance purposes after the system has been completed.
- User and operator documentation are part of the interface of the system with the environment. They fulfil an important support function.

The results of software development are summarised in Exhibit 2.6, overleaf.

B. Resources Absorbed

The most important resource required for software development consists of people who can be distinguished according

SUMMARY OF INPUTS AND OUTPUTS OF SOFTWARE DEVELOPMENT

INPUT	OUTPUT
Manpower Primary users system analyst system designer programmer technical writer librarian Secondary system programmer data base adminstration computer operator data preparation Tertiary auditor legal advisor other experts Software Standard software Special development tools Hardware Computers Other Materials Infrastructure	Structural elements flows files processes source programs object programs procedures Documentation functional specifications technical specifications user manual operator manual

EXHIBIT 2.6

to the degree of involvement. Amongst the people responsible for and therefore (normally) primarily involved in the process, are the users who will ultimately utilise the sytem to support their activities. Also involved at a primary level are dp professionals. In the early days of data processing, the latter group consisted essentially of programmers, and the tendency was to ignore the contributions of all other parties. This tendency is still noticeable in the preoccupation with programmer productivity, which will be dealt with in the next chapter.

Recent literature (e.g. Davis[52]) identifies the following dp professionals as typically directly involved in software development:

- System analyst: produces the functional specification and ensures correct working of the system.
- System designer: produces the technical specification and ensures the structural integrity of the system.
- Programmer: Codes and tests programs and procedures
- Librarian: maintains programs, modules and test data and supervises testing
- Technical writer: prepares documentation.

An issue in the organisation of software development is the extent to which some or all of these functions should be combined in one person. Many organisations indicate the type of integration by using job titles such as business analyst, analyst/designer or anlyst/programmer.

Several dp professionals are likely to give support to software development at a secondary level, notably:

- systems programmer
- data base administrator
- computer operator
- data preparation personnel

Certain persons may render assistance at a tertiary level, in a consulting capacity:

- auditors
- legal advisors
- industrial relations experts and other specialists.

Apart from the various types of manpower, other resources required for software development are:

- Hardware, notably computers. This is essential during the system construction and testing stages. Use may also be made of computer resources for other purposes, e.g. to produce sample reports or simulate system performance

- Software products. Some standard software products such as the operating system are essential to operate the computer. Other products may be specifically acquired for software development, including compilers and possibly special packages such as test data generators or source library handlers.

- Materials. This includes stationary and storage media such as discs and tapes.

- Infrastructure. A minimum of office space etc. is usually required.

A summary of the resources required for software development is shown in Exhibit 2.6.

2.4.3 Operational Definitions of Flows, Files and Processes

The concepts of flows, files and processes were identified in the previous paragraphs as structural and crucial elements of data processing software. To be able to quantify these concepts, it is necessary to establish operational definitions.

The foundations of these definitions have been laid in the previous discussion of the software development life cycle. Reference is furthermore made to the previously quoted literature as well as to accepted dp practice and private experience in data processing. Based on these foundations the following operational definitions of the concepts of flows, files and process are proposed:

A. Files

Files generally are collections of records that are both logically and physically related but, for the purpose of this definition, are restricted to master-files, i.e. files that are permanently maintained either by the system under consideration or by another system. This definition excludes transaction files, workfiles or report files which are considered to be features of the implementation, rather than essential elements of the system. Only the number of masterfiles is considered and not their content or structure. Different generations or different sort sequences of a file are not counted as separate files. The number of files is expected to be readily obtainable from the technical specifications of a system.

B. Flows

Flows are means of exchanging information. For the purpose of this definition they are restricted to external flows, i.e. exchanges of information between

the system and the environment. Excluded are therefore internal flows within the system as these are considered to be features of the implementation. Only the number of flows is considered, without regard to their content or structure. As flows are counted:

- Each different type of input record that may be presented either through conventional input records or through input files from other systems.

- Each unique type of report that can be produced by the system

- Each screen format in an interactive system which is available for display or input of information.

It is assumed that the number of flows can be readily determined from the functional specifications, specifically from a detailed system flow-chart.

C. Processes

Processes are logical or arithmetic manipulations of data. They are to be understood at a functional level, i.e. the description of a process should define what manipulation is performed on the data, not how it is done. Typical examples of functional processes are sorting, validating, updating and report presentation. Processes may coincide with programs but it is quite conceivable that a physical program contains two or more processes. Determining the number of processes in a system will require a measure of understanding of the system, which should be obtainable from the functional and technical specifications.

It is accepted that the application of the above definitions of flows, files and processes is not automatic, but requires a degree of interpretation and comprehension of the system. It is, however, contended that this should normally be well within the capabilities of an experienced dp professional. Two important issues remain, concerning these definitions: Firstly, the extent to which uniform interpretations are made by different people in different situations will determine the reliability of subsequent measures. Secondly, the ease with which the required data can be collected will determine the practicality of the measures. These aspects have been assessed in practice and are reported in subsequent chapters.

REFERENCES - Chapter 2

1. Wiener, N.: Cybernetics, John Wiley and Sons, New York, 1948.

2. Boulding, K.: General Systems Theory - the skeleton of science, Management Science, April 1956.

3. Ackoff, R.L.: Towards a system of system concepts, Management Science, July 1971.

4. Churchman, C.W.: The systems approach, Dell Publishing Company, New York, 1968.

5. Von Bertalanffy, L.: The history and status of general systems theory, in: Couger, J.D. and Knapp, R.W. (eds) Systems Analysis Techniques, John Wiley and Sons, New York, 1971.

6. Beer, S.: Cybernetics and Management, The English University Press, London, 1959.

7. Ackoff, R.L.: op.cit., p.622.

8. Ackoff, R.L.: ibid., p.663.

9. Ackoff, R.L.: ibid., p.665.

10. Bosch, P.G.: Een raamwerk voor de analyse van informatie behoeften, Informatie, September 1972.

11. Starreveld, R.W.: Bestuurlyke informatie technologie Samson Uitgeveny, Alphen a.d. Ryn, 1971.

12. Ackoff, R.L.: op.cit., p.666.

13. Davis, G.B.: Management information systems - conceptual foundations, structure and development, McGraw Hill, New York, 1976, p.5.

14. Van der Poel, K.G.: Outline of MIS course, unpublished course material, the Graduate School of Business, Cape Town, 1978.

15. Dekerf, J.: De geschiedenis van de automatische digitale rekenmachine, Informatie, October 1977.

16. Ganshorn, K., Walter, W.: Die geschichtliche entwicklung der datenverarbeitung, IBM Deutschland, Angsburg, 1975.

17. Busch, J.G., Strater, F.R.: Information systems - theory and practice, Hamilton Publishing Co., Sta. Barbara, 1974.

18. Dolotta, T.A. et al.: Data processing in 1980-1985, John Wiley and Sons, New York, 1976.

19. Duffy, N.M.: Towards more effective information systems, Systems/Stelsels, August, 1976, p.37.

20. Dolotta, T.A.: Op.cit., p.76.

21. O'Brien, J.A.: Computers in business management, Richard D. Irwin Inc., Homewood, 1979.

22. Parker, D.B.: Crime by computer, Charles Scriber's Sons, New York, 1976.

23. Davis, G.B.: Op.cit., p.237.

24. E.G.: Martin, E.W.: Electronic data processing, Richard D. Irwin Inc., Homewood, 1961.

- Wu, M.: An introduction to computer data processing, Harcourt Brace Govanovitch, New York, 1975.

25. Davis, G.B.: Op.cit., p.198.

26. Antony, R.N.: Planning and control systems - a framework for analysis, Harvard Graduate School of Business Administration, Boston, 1965.

27. E.g.: Keen, P.G.W., Scott Morton, M.G.: Decision Support Systems, Addison Wesley Publishing Co., Reading Mass., 1978.

28. Head, R.V.: Management information systems - a critical appraisal, Datamation, May 1967.

29. Blumenthal, S.C.: Management information systems - a framework for planning and development, Prentice Hall Inc., Englewood Cliffs, 1969.

30. Forrester, J.W.: Industrial Dynamics, The MIT Press, Cambridge Mass., 1969.

31. Starreveld, R.W. Op.cit.

32. Davis, G.B.: Op.cit., p.194.

33. Ackoff, R.L.: Management mis-information systems, Management Science, December 1967, p.32.

34. E.g.: Duffy, N.M., Assad, M.G.: Information management - an executive approach, Oxford University Press, Cape Town, 1980.

35. Canning, R.G.: Are we doing the right things, EDP Analyser, May 1975, p.2.

36. Benjamin, R.: Control of the information systems development cycle, John Wiley and Sons, New York, 1971, p.27.

37. Brandon, D.H.: Management standards for data processing, D. van Nostrand Co., Princeton N.J., 1963, p.7.

38. Hartman, A., Matthes, H., Proeme, A.: Management Information Systems Handbook, McGraw Hill, New York, 1968.

39. Benjamin, R.: Op.cit., p.111.

40. Murdick, R.G.: MIS development procedures, in: Systems Analysis Techniques, Couger, D.J., Knapp, R.W. (eds.), John Wiley and Sons, New York, 1974, p.88.

41. Sulcas, P.: Management accounting for in-house computers, Unpublished doctoral thesis, The University of Stellenbosch, Stellenbosch, 1978, p.66.

42. Metzger, P.W.: Managing a programming project, Prentice Hall Inc., Englewood Cliffs, 1973.

43. Enger, N.L.: Management standards for developing information systems, Amacom, New York, 1976.

44. Davis, G.B.: Op.cit.

45. Enger, N.L.: Op.cit., p.72.

46. Freeman, P.: Software reliability and design: a survey, Proceedings of the 13th design automation conference, IEEE, 1976.

47. Geenen, J.: Inleiding tot methodisch ontwerpen, Informatie, March 1977.

48. Peters, L.J., Tripp, L.L.: Is software design wicked, Datamation, May 1976, p.131.

49. Enger, N.L.: Op.cit., p.104.

50. Davis, G.B.: Op.cit.

51. Gildersleeve, T.R.: Successful data processing systems analysis, Prentice Hall Inc., Englewood Cliffs, 1978, p.13.

52. Davis, G.B.: Op.cit., p.370.

CHAPTER 3

DEFINING THE EFFICIENCY OF SOFTWARE DEVELOPMENT

CHAPTER 3

DEFINING THE EFFICIENCY OF SOFTWARE DEVELOPMENT

3.1 INTRODUCTION

> "However you define the scientific activity, measurement pervades most of the enterprise"(1)

The above statement by Stevens seems particularly relevant to the understanding of software development today. Though terminology such as "software engineering" and "software technology" is used, most of the activity is performed in a traditional rather than a scientific manner, but practitioners are becoming aware of the need for measurement and quantification before real progress can be expected. Kirkley sums up the situation as follows:

> "Today we have the intuitive feelings of legions of burned users whose request for systems resulted in jobs that took twice as long and cost three times as much as the original estimate. We have generalities and vague approximations but we do not have precise measurements. After all these years we cannot answer the question: 'software is expensive relative to what?" (2)

Jones explicitly refers to a lack of adequate measures:

> "In the field of computer programming, the lack of precise and unambiguous units of measure for quality and productivity has been a source of considerable concern to programming managers"(3)

Implicit in the above quotations is an indication that there are two areas in which the lack of quantitative measurement is particularly felt - namely estimation of

expected effort before a project is begun, and measurement as a condition for the effective promotion of productivity of software development. Estimation of expected effort is often poor guesswork as Kirkley has attested, and the literature offers extremely little support to the would-be estimator. Several well-known textbooks[4] do not discuss the subject at all. Part of the problem in estimation lies in the difficulty of defining the scope of development projects at an early stage. An equally important problem is the lack of a general yardstick and therefore the inability to control the efficiency of software development by manipulating the factors that influence it. It is to these problems that this study addresses itself.

A scientific approach to the promotion of productivity or efficiency of software development is obviously dependent on the proper definition and measurement of this concept. In the general literature, no distinction is made between productivity and efficiency, e.g. Faraday[5] states in a study entitled "The management of productivity" published by the British Institute of Managment: "We are equating efficiency with productivity". An authoritative study sponsored by the OECD (Organisation for Economic Cooperation and Development) states[6] that, although efficiency goes slightly beyond productivity, "it may be taken in its widest sense as practically synonymous with productivity". The same study defines productivity as the relationship between the volume of production and the sum of the production factors included in the production cycle. Faraday[7] introduces a refinement which is important in the context of software development: "productivity is the ratio of a prescribed output to the required inputs". It is to be noted that the terms "input" and "output" in this definition are to be interpreted in the economic sense, i.e. in our case, "inputs" are the resources absorbed by the software development process, and "outputs" are the results of that process. The following definition of

efficiency will therefore be adopted: Efficiency in software development is the ratio of a well defined result to the resources absorbed by the process. This definition will be the basis for a scientific approach to the creation and testing of a measure of efficiency of software development, which is the subject of this chapter.

3.2 REPORTED STUDIES

Several studies that deal with the efficiency of software development, or at least aspects thereof, have been reported. Remarkably, all these studies concentrate almost exclusively on the programming task which, as explained previously, is only a part of the total system development effort. The main thrust and conclusions of these studies are reviewed below

- Sackman et. al.[8] studied the effect of the use of on-line facilities on programmer performance. "Performance" was taken as the amount of programmer and CPU time required to program a simple, well defined problem. It was found that the use of on-line facilities provided significant improvements but the difference in performance of individual programmers was an order of magnitude greater. No attempt was made to refine the performance measure used.

- Weinberg[9] introduced the notion of measuring programming performance according to five criteria namely: core storage used, output clarity, program clarity, number of statements, and time used. He showed that programmers who were given the same problem were capable of optimising on any of these criteria, at will.

- Johnson[10] proposed the use of the number of lines of source code written per man day as a measure of

programmer productivity. He warned however that: "inconsistent definitions can account for dramatic differences"[11] and proposed the use of different standards for (subjectively determined) levels of difficulty as well as other distinctions.

- Walston and Felix[12] in an extensive study of 60 projects that involved different computers and programming languages, used delivered source lines per man month (DSL/MM) as a unit of measure. They reported reasonable correlation between DSL and documentation produced and only weak influences from factors such as the use of RJE facilities and the geographical proximity of developers to users. However, significant difference in productivity was reported between projects with high and low customer interface complexity.

- Jones[13] specifically addressed the problem of measuring both the quantity and the quality of the output of the programming effort. As a qualitative measure he focussed on the number of defects found in the original program, the effectiveness of subsequent defect removal efforts and the defects remaining after all reviews, inspections, tests and so on, which he termed the "maintenance potential". He stated that "it is well to observe that the term quality has been used here to mean absence of defects. Of course there are many other attributes associated with quality than defect rate" but he introduced no other measures to take care of this problem. Another problem raised in the same study, is the inadequacy of using measures that are geared essentially to the programming task for measuring the efficiency of the total software development effort: "The complete job of developing a computer program requires more than coding activities, and these activities must also be measured. Therefore,

when lines of code per programmer month is used on the non-coding task, the results are apt to be questionable"[14] Several other components of total effort and cost were also examined by the same author but the efficiency measure finally retained was lines of code per dollar expended or the inverse of that measure, dollars per line of code. Several conditions were however discussed that would complicate the use of these measures.

- Crossman[15] related his experience with a method of measuring programming productivity in which the unit of output is not lines of code, but number of functions. A function is defined as a section of a program which performs a specific activity, has one entry and one exit, conforms to the rules of structured programming and has between five and fifty source statements. In a study of 14 programs, a consistent relationship was found between functions produced and man-hours spent (the indication is that only programmer hours were counted). It was also reported that the only factor found to influence this relationship significantly was the use of new tools on a system. It should, however, be noted that this study was confined to a very stable situation where strict and consistent quality was enforced.

- Parikh[16] reviewed the merits of lines of code (LOC) per man-day as a measure of productivity. He pointed to a number of problems and stated that, when not defined precisely, LOC could in fact be misleading. It was advocated that, as a measure, it should be supported by a number of additional "metrics", designed to suit the situation. Examples of such metrics are: defects found during the first month of operation, execution speed and core storage used and deviation from the original development time schedule. Attention should also be paid to the design and analysis stages

and to the benefits derived from the system by the users.

- Halstead[17] initiated an approach to software and software development which he labelled "Software Science" It is interesting to note that Software Science approaches its subject with the same intentions as this study, namely to apply the scientific method, specifically measurement and quantification to software and software development. Software Science is concerned with programs and algorithms that are usually small and technically oriented. The main measures proposed (e.g. program volume) are intuitively acceptable, but some (e.g. program level) remain rather speculative. These measures finally lead to an estimate of the effort required to produce or understand a particular program and thereby to a norm of efficiency. However, only programming time is included in this effort. All measures are derived from the completed program, so that they have no predictive value concerning the development effort required, unlike the measures proposed in this study. The differences between Software Science and this study, basically results from an orientation towards technical programming on the one hand and data processing software development on the other.

While admiring the pioneering effort and the valuable contributions made by these studies, it appears that the following factors mitigate against the acceptance of lines of source code per man-hour in particular as an adequate measure of software development efficiency:

1. Programming is only a part of the total software development effort. In terms of the total effort expended it is often a relatively minor task: e.g. Boehm[18] reports the effort spent on programming as 20% of the total software development effort, while Jones[18] puts

this figure at 30%. While these percentages are rather speculative, it is interesting to note from the 1977 survey by the SA Bureau of Statistics[20] that the ratio of analysts to programmers was about 1 to 3. It appears therefore unjustified to use the quantity of programming work alone as a basis to calculate the efficiency of the total software development effort.

2. Trade-offs between system design, programming and implementation are usually possible. Concentration on the efficiency of programming alone may therefore well lead to overall inefficiency.

3. As discussed in the previous chapter, recent developments are likely to diminish the role of conventional programming languages and thereby make the counting of lines of code obsolete.

4. There is a high degree of uncertainty about the most suitable definition of lines of code and no proof that the number of functions can be measured reliably in different circumstances.

5. Programming hours should obviously not be taken as the only input into the software development process. Contribution from other parties such as analysts and users as well as resources such as computer time and the use of software tools must be taken into account.

6. An elementary assessment of validity, reliability and sensitivity was not provided for any of the measures proposed thus far.

3.3 PRINCIPLES FOR THE DEFINITION OF A SUITABLE MEASURE OF EFFICIENCY OF SOFTWARE DEVELOPMENT

As is clear from the previous section, the definition of an adequate measure of efficiency of software development is a task which poses many theoretical and practical problems. It is therefore felt to be useful and important to begin by establishing a number of principles which may serve as guidelines to overcome these problems. Listed below are six principles which are considered to be appropriate foundations for the construction of a suitable measure of efficiency of software development, and a justification for each of these principles.

3.3.1 The Complete Software Development Life Cycle should be taken into account, not just one phase

As argued in the previous chapter, the phases of the software development life cycle are strongly interrelated. The effort required in each phase is dependent on the standard of the work performed in the previous phase and, in turn, determines the complexity of the tasks in the following phases. For example, the time spent on system design and therefore implicitly the quality of the design, is an important determinant of the programming effort, while a high quality of programming facilitates the implementation task. It is therefore considered fruitless to attempt to define a measure of efficiency for only one phase, as much as such a measure would require a very precise definition of the quality of the design and the quality of the programming. Such precise definitions are considered to be totally unobtainable in data processing practice.

3.3.2 All Resources used in the Software Development Process must be taken into Account

The resources normally used in software development were

enumerated in the previous chapter. The ratio in which these resources are to be used is, however, not fixed and substantial variations have occurred in the history of software development. In the early days, programmers did most of the work, while currently, analysts and other specialist staff play an increasingly important role. When computer hardware was expensive it had to be used sparingly, but at present the tendency is to use on-line computer power extensively during development. Preoccupation by management with one particular resource such as manpower could easily lead to wasteful use of other resources and therefore detract from overall efficiency. To be practically useful a measure of efficiency should therefore take all resources into account.

3.3.3 Consideration must be given to all Attribute of the Result of the Software Development Process

The scientific method of establishing a measure of a phenomenon requires[21] that all measurable attributes of that phenomenon are considered. From this domain the most significant attributes must then be selected to constitute a multidimensional or unidimensional measure. This procedure focusses the attention on the inherent multidimentionality of software. Properties to be considered are not only its volume (as in lines of code) but also other aspects such as effectiveness or flexibility. Only when it has been demonstrated that these attributes do not play a significant role or are correlated with some other attributes, can they be excluded from the measure.

3.3.4 The Attributes Constituting the Measure should be Quantifiable, not highly Correlated, and Discriminating

Quantification is the objective of the measure of efficiency. Therefore non-quantifiable, judgemental attributes of the

resources used or results produced should be avoided when possible. Where two attributes are correlated, one can be implicitly represented by the other and it is therefore not necessary to include both attributes. Attributes that are essentially the same for all software to be considered are not discriminating and can therefore safely be omitted from the measure.

3.3.5 The Attributes Defining the Results of the Software Development Process should be known before most of the Resources are spent

The purpose of this principle is that when an experience value for efficiency would be available, determination or estimation of the results of the software development process would allow calculation of the necessary resources. In this way the measure can help to overcome one of the practical problems mentioned at the beginning of this chapter, namely the inability to estimate software development costs.

3.3.6 The Function Defining the Measure of Efficiency should be as simple as possible

This is the principle of parsimony which is well known in the history of science.[22] It leads to preference for simple mathematical and logical relationships.

3.4 PROBLEMS IN MEASURING RESOURCES USED BY AND RESULTS OF THE SOFTWARE DEVELOPMENT PROCESS

The above principles should help to arrive at an adequate measure of the efficiency of software development, but a number of practical and conceptual problems must be anticipated. Practical problems appear to prevail in the definition of the

resources absorbed by the process while conceptual problems predominate in the definition of the results produced.

3.4.1 Problems in the Definition of Resources Absorbed

The resources required for software development were defined in the previous chapter and summarised in Exhibit 2. They are manpower, hardware, materials and infrastructure. These resources can easily be integrated by multiplying each with its relevant unit cost and adding the resulting costs, to arrive at the total development cost. The following objections may, however, be raised against this procedure:

- Some of the cost elements may, in practice, be correlated, thus making it possible to omit these from the measure. Though correlations may well exist in the short run, it is not certain that in the long run such correlations will continue to hold. Also, including all cost elements does not appear to be unduly complicated while it ensures that the total development cost appears in the formula, which has the advantage that the measure is simple and intuitively understood.

- For some of the cost elements the unit cost may be difficult to obtain. This is a practical objection which may pose a problem indeed, particularly as far as the collection of survey data is concerned. As discussed by Sulcas[23], accounting procedures do exist to cope with this problem, and when management is seriously concerned about the improvement of efficiency it should be feasible to obtain relevant cost figures. Such figures will probably also prove useful for other purposes, such as the charging out of services.

- Some organisations do not record resources used for software development as a matter of course. This is probably

the exception as far as man-power is concerned, but is more common for other resources. This is again a practical problem which will make the collection of survey data difficult, but interested managers should not have undue trouble in recording the data and would find the effort well worthwhile. The practical extent of this problem and the previous one may well be established through a special survey, the details of which will be discussed in Chapter 5.

- Inflation may make the aggregation and comparison of costs relating to different periods invalid. This problem does not exist when the data relate to a short enough period of time. Over longer periods, the problem can be eliminated by the use of an index to convert all costs to a base period.

In summary, there appear to be no conceptual problems with the definition of resources used for software development. The obvious practical problems are such that concerned managers should be able to control them.

3.4.2 Problems in the Definition of the Results Produced

As suggested in section 3.3 the selection of attributes of software to be included in the measure of efficiency should start from the definition of the domain of all relevant attributes. This is a challenging task, as most practitioners are not used to analysing the properties of software in any detail and attention to these questions in the literature is limited. Interesting indications of relevant attributes of software are given by Weinberg,[24] who identifies 4 potential objectives namely: output clarity (effectiveness), program clarity (flexibility), speed of execution (device efficiency) and program size (physical dimension). A major study of the characteristics of software quality by Boehm

et. al.[25] mentions the same attributes, though using a slightly different structure. Yourdon[26] lists as the main aspects of program quality: effectiveness, flexibility, maintainability and the resources required during execution. On the basis of these analyses a list of attributes of the result of the software development process has been assembled of which the main categories are: effectiveness, flexibility, device efficiency and physical dimensions. A further breakdown of these categories is shown in Exhibit 3.1. This list of attributes will be used as a basis to select the most appropriate attributes according to the criteria established in section 3.3.4. In the following paragraphs, each of the attributes will therefore be evaluated against these criteria.

A. Effectiveness

This is a measure of how well a data processing system performs its ultimate function - namely to support decision making in an organisation. To be effective, the information produced must have a number of characteristics which are defined by King and Epstein[27] as: timeliness, significance, understandability, precision and efficiency. It is clear that ineffectiveness of a system may totally negate any efficiency achieved in the development of the system.

The effectiveness of a data processing system is, however, notoriously difficult to establish and particularly hard to quantify. Under the title "Why information systems fail" Lucas[28] has reported the result of six studies involving the effectiveness of systems. In only one of these an attempt was made to measure effectiveness directly by asking users to rate various system aspects. The author introduced the discussion of this procedure by stating: "It is difficult to rate the quality of a system because there are no accepted standards".[29]

ATTRIBUTES OF SOFTWARE

Main Categories	Attributes
A. Effectiveness are the decision making requirements adequately supported?	- adequacy - timeliness - correctness - suitability - reliability
B. Flexibility can corrections and changes be made with relative ease?	- readability - maintainability - modifiability
C. Resources required for operation what complementary resources are needed to operate the system?	- processing requirement - storage requirement - communication requirement - personnel
D. Physical Dimensions what physical entities have been produced by the software development process?	- flows - files - processes - documentation - number of functions - lines of source code - size of the object code

EXHIBIT 3.1

In a study on the quantification of financial benefits of information Kleynen[30] describes a series of experiments with computerised management games designed to measure the effect of information. His conclusion is that "without at least some rudimentary theory, we just simulate our own ignorance". At a congress held under the auspices of the Intergovernmental Bureau of Informatics in 1974[31] several speakers addressed the problems of measurement of effectiveness of systems but none of these offered a practical procedure for its measurement and Banbury[32] concluded: "With current knowledge, the technique would be immensely difficult to apply, so many of the issues are only partly understood and so few of the variables are quantifiable".

Furthermore, experience appears to indicate that similar organisations working with similar technology on fairly similar problems, tend to come up with systems that exhibit similar characteristics. One such example occurred when all major airlines found themselves compelled to install similar flight reservation systems within a fairly brief span of time. Another example was when the major building societies implemented real-time teller systems in quick succession.[33] A reason for the emergence of systems with similar characteristics may be that within a well-defined framework of space, time, type of application and data processing maturity, a certain level of effectiveness appears to be obtainable and is therefore aimed for. For the present study such a framework has been very carefully defined. It seems therefore reasonable to assume a fairly standard level of effectiveness for the subset of data processing systems defined in the previous chapter.

The probability of convergence to this common standard of effectiveness is further increased as this study is limited to data processing sytems which are operating

successfully and may be assumed to be "shaken down".

In summary, as effectiveness is not readily quantifiable and is unlikely to discriminate substantially, it is not practical to include this attribute in the measure of efficiency. It is ackowledged that, if quantification ever appeared feasible, the attribute would definitely be of potential interest.

B. Flexibility

The importance of flexibility and its constituents, namely readability, maintainability and modifiability has been studied extensively by Duffy.[34] An important finding of his research is that organisations subject to similar incidence of change, tend to use similar techniques to enhance flexibility. Between similar organizations differences in flexibility are therefore likely to be small. Furthermore, it is important that data processing systems in the present study are restricted to systems that are fully implemented and accepted by the users. It seems reasonable to assume that at that stage due attention must have been paid to the maintenance problem and that the system and its supporting documentation must have been brought up to a reasonable standard.

Quantification of flexibility is, once more, an extremely difficult task. Duffy[35] proposes to equate it to a rating by users, whereas Boehm et al. consider it to be an inherent property of software which can be deduced from the code.[36]

Flexibility, therefore, also does not seem to be an attribute that should be included in the measure of efficiency. Like effectiveness, it is not readily

quantifiable and does not discriminate adequately.

C. Device Efficiency

Low usage of resources such as disk space or CPU capacity during program execution is, in principle, a desirable feature and the software which achieves this is therefore (certeris paribus) more desirable. Recent technological progress has, however, made this attribute increasingly insignificant and it is expected that this trend will continue. Also, data processing systems as defined for this study are typically much smaller users of hardware resources than other types of systems such as computer aided design or decision support systems. Many organisations, as a consequence, tend to be rather unconcerned about efficiency of data processing systems.

This attribute therefore does not appear to be a good, discriminating measure of the result of the software development process.

D. Physical Dimensions

In the previous chapter the essential elements of data processing systems were shown to be flows, files and processes and their definitions were given in paragraph 2.4.3. It was stated that these elements can be considered to encompass the primary results of the software development process. They are expected to be readily quantifiable and reasonably discriminating, while there is no reason to believe that they are highly correlated. These attributes, particularly when taken together, therefore appear to be well-suited to constitute part of the measure of efficiency.

Documentation, consisting of functional specifications, user manual and operations manual, was identified as a secondary result of software development. There is, however, reason to believe that the volume of documentation is dependent on other attributes of the system. Walston and Felix[37] have shown a significant correlation between lines of source code and volume of documentation. It is furthermore dubious whether the volume of documentation is readily quantifiable in all situations. Documentation standards are likely to differ substantially among organisations, while no accepted definition exists of what constitutes, for example, a page of documentation. It also remains to be established to what extent functional and technical specifications exist and are updated.

The volume of documentation is therefore not considered to be a very useful attribute but it will be interesting to investigate its quantifiability and correlation with other attributes such as flows and files and processes.

There are a number of other measures describing the physical dimension of software, notably lines of source code, number of functions, or size of the object code. These are at least quantifiable and are likely to be discriminating. They do, unfortunately, only relate to a particular part of data processing systems, namely the programs, and would therefore be alternatives to the concept of processes introduced previously.

Their drawbacks can be summarised as follows:

- They are tied to a particular programming language or technique which itself may become outdated.
- Different interpretations of lines of code and of functions may make these measures highly unreliable.

- The size of the object code is heavily dependent on the level of the compiler used.

- These measures will not be available before virtually all the work on the project is done.

These attributes therefore seem less attractive than a more simple definition of processes, in conjunction with measures of flows and files. It will, however, be interesting to see to what extent they will correlate with other attributes.

In summmary it is concluded that, having considered a wide domain of attributes, most of these are found to be unsuitable to measure the result of the software development process. Straightforward definitions of flows, files and processes appear to be preferable to more detailed specifications such as lines of source code. In the following paragraph a measure of efficiency based on these attributes will therefore be proposed. Tests of the appropriateness of this measure will be worked out in subsequent chapters.

3.5 FORMULATION OF A MEASURE OF EFFICIENCY OF SOFTWARE DEVELOPMENT

Having selected the most appropriate attributes to be included in the desired measure of efficiency, it remains to define a formula to tie these attributes together. The formula is to be of the nature: results produced divided by resources absorbed. As reasoned in previous sections, resources absorbed should be a simple summation of all cost elements, while results should be some aggregation of the number of flows, files and processes. Bearing in mind the principle of simplicity expressed in section 3.3.6, a simple additive model appears most appropriate for this part of the model but other functions may also be considered. The following abbreviations will be used:

E is efficiency
Pr is the number of processes
Fl is the number of flows
Fi is the number of files
C is total development cost
x, y, z are weighting coefficients
w is a constant

and the general model will be of the form:

$$E = \frac{f\ (Fi,\ Fl,\ Pr)}{C}$$

where f is an arbitrary function.

3.5.1 The Simple Linear Model

The simplest form of the linear model is represented by the formula:

$$E = \frac{Fi + Fl + Pr}{C} \qquad (I)$$

According to the principle of parsimony or simplicity mentioned before, this a very attractive model. No 'artificial' parameters are introduced and the model has intuitive appeal: efficiency is equal to the sum of flows, files and processes divided by the total development costs. Also, all the variables are readily quantifiable and the number of files, flows and processes ought to be known at an early stage in the development of the project. Futhermore, the variables in the numerator are probably not highly correlated, but this remains to be tested empirically. If this model can be shown to be empirically acceptable, there are, therefore, strong arguments to accept it at least as working model, possibly to be refined in specific circumstances.

3.5.2 Other Models

Model I really is a special case of the general linear model:

$$E = \frac{w + x.Fi + y.Fl + z.Pr}{C} \qquad (II)$$

The weighting factors x,y and z indicate the different relative importance of the factors flows, files and processes, while the constant w represents a 'fixed cost'. The definition of efficiency given in paragraph 3.1 does, strictly speaking not provide for a 'fixed cost' element but it might be of some interest to see what empirical figures may be obtained.

Unfortunately, there is no theoretical basis for determining the parameters w, x, y and z. If this model is to be used, the weights will therefore have to be estimated through regression analysis, making use of the survey data to be defined in Chapter 5.

Another model which may be considered is a multiplicative, or log-linear model of the form:

$$E = \frac{Fi^x . Fl^y . Pr^z}{C} \qquad (III)$$

This model achieves linearity when the log of cost is regressed against the log of flows, files and processes. It has the theoretical attraction that it avoids the addition of dimensionally different quantities. Practically, this model is however problematic, as it implies that an increase in e.g. the number of files from 2 to 4 or from 4 to 8 would have the same effect on the work content. This appears unrealistic but empirical analysis might still be of some interest.

Another possible formulation is the following model:

$$E = \frac{(Fi + Fl + Pr)^x}{C} \qquad (IV)$$

If x is greater than one this model implies that, if the size of a project increases by a certain factor, the work content increases by more than that factor. This sounds attractive, as big projects are often disproportionally complicated, particularly in the analysis phase. On the other hand, large projects could be favoured by economies of scale, which would be indicated by a value of x less than one. This model may therefore offer a interesting alternative.

Many other model formulations might be considered, but it is felt that the above models cover the most likely hypotheses.

On the whole, the theoretical simplicity and intuitive attraction of the simple linear model (model I) are considered to weigh heavily in its favour. Tests to determine whether this model is indeed acceptable will be proposed in the next chapter, while survey procedures to collect the necessary data will be described in Chapter 5. If these tests are not successful, other models will be investigated and tested in detail. If, however, the tests indicate that model I is acceptable, other models will only be cursorily inspected.

3.6 CONCLUSION

With the formulation of the above model of efficiency of software development and the selection of Model I as the preferred model, the first two objectives of this study as defined in Chapter 1 have been attained. The necessary concepts were defined in Chapter 2 and the model was built

up step by step in Chapter 3. In terms of the objectives, it remains to set criteria for the acceptability of the proposed model, test it and, if found to be acceptable, use it for further analysis. This will be undertaken in the next chapters.

REFERENCES - Chapter 3

1. Stevens, S.S.: Measurement and Man, Science, 21 February, 1950, p.385.

2. Kirkley, J.L.: Programmer Productivity, Datamation, May 1977, p.385.

3. Jones, T.C.: Measuring Programming Quality and Productivity, IBM Systems Journal, nr. 1 1978, p.40.

4. E.g. Davis, G.B.: Management Information Systems, Conceptual Foundations, Structure and Development, McGraw Hill, New York, 1976.

5. Faraday, J.E.: The Management of Productivity, British Institute of Management, Tonbridge, 1971, p.7.

6. Deurnink, G.: (ed), Productivity Measurement, Volume I: Concepts, Organisation for Economic Cooperation and Development, Paris, 1955, p.26.

7. Faraday, J.E.: op.cit., p.13.

8. Sackman, H., Erikson, W.J., Grant, E.E.: Exploratory Studies comparing On-line and Off-line Programming Performance, Communications of the ACM, January 1968.

9. Weinberg, G.M.: The Psychology of Improved Programming Performance, Datamation, November 1972.

10. Johnson, J.R.: A Working Measure of Productivity, Datamation, February 1977.

11. Johnson, J.R.: ibid, p.107.

12. Walston, C.E., Felix, C.P.: A Method of Programming Measurement and Estimation, IBM Systems Journal, nr. 1 1977.

13. Jones, T.C.: op.cit.

14. Jones, T.C.: ibid., p.52.

15. Crossman, T.D.: Taking the Measure of Programmer Productivity, Datamation, May 1979.

16. Parikh, G.: How to Measure Programmer Productivity, SA Computing, April 1981.

17. Halstead, M.H.: Elements of Software Science, Elsevier North Holland, New York, 1977.

18. Boehm, B.: Software and its Impact: A Quantitative Assessment, Datamation, May 1973, p.52.

19. Jones, T.C.: op.cit, p.56.

20. South African Bureau of Statistics: Computer Survey 1977, Government Printer, Pretoria, 1978, P.63.

21. Thorndike, R.L. and Hagen, E.: Measurement and Evaluation in Psychology and Education, 3rd ed. John Wiley and Sons, New York, 1969, p.14.

22. The New Encyclopaedia Britannica, Encyclopaedia Britanica Inc., London, 1979, p.475.

23. Sulcas, P.: Management Accounting for In-House Comuters, Unpublished Doctoral Thesis, The University of Stellenbosch, 1978.

24. Weinberg, G.M.: op.cit., p.83.

25. Boehm, B. et al.: Characteristics of Software Quality, North Holland Publishing Company, Amsterdam, 1978, p.xiii.

26. Yourdon, E.: Techniques of program Structure and Design, Prentice Hall Inc., Englewood Cliffs N.J., 1975, p.7.

27. King, W.R. and Epstein, B.J.: Assessing the Value of Information, Management Datamatics, nr.4, 1976.

28. Lucas, H.C.: Why Information Systems Fail, Columbia University Press, New York, 1975.

29. Lucas, H.C.: ibid, p.55.

30. Kleynen, J.P.: Computers and Profit: Quantifying Financial benefits of information, Unpublished Manuscript, 1979, p.308.

31. Frielink, A.B.: (ed), Economics of Informatics, North Holland Publishing Co., Amsterdam, 1975.

32. Banbury, J.: The Concept of Information Systems Effectiveness, in: Frielink, A.B.: (ed), Economics of Information, North Holland Publishing Co., Amsterdam, 1975, p.34.

33. Botha, F.G.: An Anlysis of the Role of Software Languages in the Development of the Computer Information Systems of the Major Financial Organisations in South Africa Unpublished MBA thesis, The Graduate School of Business, Cape Town, 1977.

34. Duffy, N.M.: The Design of Flexible Computer-based Information Systems, Unpublished Doctoral Thesis, The University of South Africa, Pretoria, 1977, p.84.

35. Duffy, N.M.: op.cit., p.25.

36. Boehm, B.W.: et.al.: op.cit., p.xiii.

37. Walston, C.E., Felix, C.P.: op.cit., p.68.

CHAPTER 4

ASSESSING THE APPROPRIATENESS OF THE PROPOSED MEASURE OF EFFICIENCY

CHAPTER 4

ASSESSING THE APPROPRIATENESS OF THE PROPOSED UNIT OF MEASURE OF EFFICIENCY

4.1 INTRODUCTION

> "More stupefying than the sheer number of our measures is the ease with which they are proposed and the uncritical manner in which they are accepted. In point of fact, most of our measures only measure because someone says they are, not because they have been shown to satisfy standard measurement criteria"(1)

In the previous chapter a unit of measure of the efficiency of software development was built up from first principles and a choice was made of what appeared to be the most suitable quantitative formulation. Standard practice, detailed in the literature on measurement,(2) requires that a formal assessment is now made of the appropriateness of the measure. The criteria to be applied are well documented(3) and are, with slight variations, identified as validity, reliability and sensitivity. These concepts will be defined more precisely later, but are roughly to be understood as follows: validity means that a measure makes sense, reliability assures that it is a stable and uniform yardstick, while sensitivity requires a scale which allows sufficient distinction.

To be acceptable, a measure must obviously meet these three criteria. It is, however, remarkable to note that none of the studies reviewed in section 3.2 discusses any of these aspects explicitly. Yet, the unit of measure used most frequently (lines-of-code per unit-of-time) is not intuitively

acceptable as argued in the previous chapter. Where some authors[4] imply concern about the appropriateness of the measure, it is in attempting to show that there is a high correlation between lines-of-code and unit-of-time. Assuming circumstances do not vary too much, such a correlation can be taken as an indication of the validity of the measure, as will be explained later. By itself, a high correlation will however not be convincing evidence of the validity of a measure of efficiency.

In the following sections, the measure of efficiency proposed in the previous chapter will be tested against the criteria of validity, reliability and sensitivity. This will not only provide an assessment of the appropriateness of this particular measure, but also point to procedures that may be used to test other similar measures.

Validity will be dealt with first, but its formal assessment will require an extensive survey which will described in the next chapter. Reliability will be tested by means of a specially developed experiment discussed in this chapter, while sensitivity will be found not to require any special experimentation.

4.2 THE VALIDITY OF THE MEASURE

The validity of a measure is defined by Tull and Albaum[5] as "the extent to which measured values truly reflect the property of interest", or stated more simply, whether what one really measures is indeed the same as what one intends to measure. Validity as thus defined, can sometimes be assessed. For example, a test which purports to predict academic achievement may be validated by correlation of the test results with subsequent actual achievements. This example also makes it clear that there is typically a degree

of validity rather than absolute presence or absence of it.

The validity of units of measure and more specifically of test results has been of great concern in psychometrics and specifically in educational research. Meyer[6] has summed up the various facets of validity discussed in the literature and distinguishes the following aspects of validity:

- Face validity which is defined by Osgood et. al.[7] as the extent to which "distinctions it provides coincide with those which would be made by observers without the aid of the instrument".

- Content validity, defined by Shaw and Wright[8] as "the degree to which the items on the scale sample the content of the .... domain".

- Construct validity. This assumes that it is the intention to measure one or more traits (or contructs) which are implicit in the measurement results. The degree to which isolation of these contructs is successful can be measured statistically.

- Predictive validity. This is defined as the correlation between a predicitive measurement and the ultimate result. Its assessment may be complicated by difficulty in defining the result itself.

- Concurrent validity. This consists of relating (usually by means of correlation) two measurements of essentially the same property.

4.2.1 Practical Assessment of Validity

The proposed measure of efficiency consists of two parts,

namely a measure of the resources required and a measure of the results produced by the software development process. The measure of resources absorbed (total project cost) has an overwhelming amount of "face validity" as defined above and is therefore considered not to require any further assesment of its validity.

The measure of results produced (flows, files and processes) is novel and more debatable. Explicit procedures to validate this part of the measure will therefore be pursued. Tull and Albaum[9] have given a number of practical suggestions for the assessment of the validity of a unit of measure in the general context of survey research. Of those, the following are to a greater or lesser extent applicable:

- Logical validation: This requires that the measure chosen is "intuitively obvious" or theoretically attractive. The effect is closely related to face validity. The definition of the results of software development is firmly rooted in systems theory and careful observation of the software development process. A certain degree of validity may be derived from this. In cases such as this it is, however, in the words of Tull and Albaum[10] "not wise to rely on logical validation alone".

- Validation by expert opinion: This requires the consensus of a group of recognised experts on the acceptability of the measure. This assessment can certainly be made, but requires a carefully planned survey. The design of a survey which will serve this purpose (as well as others) will be described in the next chapter.

- Validation by corroborative design: This is performed by measuring the same characteristic twice, using different units ofmeasure and calculating the correlation between the two measures. The result of

this procedure clearly amounts to concurrent validity, as defined before. This form of validation can, indeed, be performed for the proposed unit of measure, since an alternative measure exists, namely lines of source code. As explained previously, this latter measure is, however, itself of questionable validity so that, assuming positive correlation, at best a small contribution to validity will be derived from this approach. To perform this validation, careful survey research is again necessary, which will be described in the next chapter.

Another opportunity for validation through corroborative design is offered by testing the correlation between the resources used and results produced by software development. When this is done for a substantial number of projects, a positive correlation is to be expected but the correlation coefficient may not be very high, as a large number of factors is likely to influence the ratio. It is, in fact, believed that, though a trendline should be visible, both positive and negative outliers are likely to be encountered. The specific result of this correlation therefore also forms only a small contribution to the assessment of validity of the measure and will require careful survey research.

- A specially designed test of validity: As the above approaches only provide a weak form of validation, a further procedure has been specially designed to test the validity of the proposed measure of the result of the software development process. This procedure will provide an amalgamation of face validity and concurrent validity and also needs to be established through survey research. Consider the following experiment: A number of data processing experts (managers) are asked to identify three data processing systems with which they are

familiar and rank these in order of size (e.g. small, medium, large), using their own intuitive assessment. Application of the proposed measure of the result of software development should then at least preserve the original rankings. This test will again not provide a definite proof of validity but, if successful and combined with other suggested tests of validity, should provide substantial support. Further details of the survey procedure to perform this test will be discussed in the next chapter.

4.3 THE RELIABILITY OF THE MEASURE

According to Tull and Albaum[11] a measure is reliable: "when it will consistently produce about the same results when applied to the same sample or to different samples of the same size drawn from the same population". The concern is clearly that, when the measure is not reliable, assessment of essentially the same phenomenon could yield different values in different circumstances. Regarding the proposed unit of efficiency of software development, potential sources of variation (or inconsistency, or instability) are:

- Inaccessibility of the necessary data. This may lead to approximation rather than enumeration of certain data. This is unlikely to be a problem in the determination of flows, files and processes but may pose difficulties in the assessment of costs. These difficulties are, however, not inherent in the nature of the measure and can be minimised when organisations pay sufficient attention to this aspect.

- Deviations in successive measurements by the same person due to environmental influences. This is stressed by several authors, notably Churchill,[12] as an important source of variation in market research.

It is, however, considered to be a minor importance in the context of the present study, as the efficiency of the same project will not be measured repeatedly by different persons.

- Differences in the interpretation of the definition of the various elements of the measure. Difficulties in the interpretation and collection of the cost data were discussed in section 3.4.1 and were found to be definitely surmountable, given the right managerial climate. The same remains to be demonstrated for the definitions of the elements of the physical dimension of software, namely flows, files and processes. Unreliability may be introduced by the state of the availability documentation and the views of the person performing the assessment. This could be a potentially important source of a variation and cause of unreliability. To establish the necessary confidence in the measure, a small scale test has therefore been designed to determine this order of magnitude of this variation. The design and results of this test are discussed in the next paragraphs.

4.3.1 An Experiment to Determine Reliability

The objective of a test of reliability is, as stated by Meyer "to obtain ultimately an estimate of error variance in (his) measurements".[13] In many measurement or testing situations, it is extremely difficult to obtain such estimates of error variance directly and an indication of reliability is therefore derived from the correlation between either consecutive applications of the same test or concurrent applications of equivalent test. Specific procedures mentioned in the literature[14] are test - re-test, split - halves and parallel forms. These methods rely essentially on the repetition of the test by one subject. In the present problem, namely the determination of the reliability of the measurement of software size, such

methods are impractical since it is unrealistc to expect a respondent to assess the size of the same system twice while it is also not possible to present a respondent with two different descriptions of the same system.

It was therefore decided to focus attention exclusively on the variation in measurement between different respondents. The quantification of this variation was defined as the coefficient of variation, i.e. the standard deviation of the measurement as a fraction of the mean. No absolute criteria for this parameter exist, but any value below 0,5 may generally be taken as small.

As a basis for evaluation, a proposal containing functional and technical specifications of a small, typical data processing system was obtained. It would obviously have been preferable to use the specifications of a fully implemented system, but this proved impossible to obtain, due to the natural reluctance of organisations to divulge details of implemented systems. Participants in the experiment were five d.p. experts, each with ample experience in software development. The participants were given a copy of the system specifications and asked to count and report the number of flows, files and processes, using the definitions of these attributes as shown in Chapter 2. The results of this experiment are shown in Exhibit 4.1 overleaf.

4.3.2 Conclusions from the Experiment

From Exhibit 4.1 it can be seen that there was no unanimity on any of the attributes, but that a fair level of agreement was nevertheless achieved.

In discussions with the participants it was found that the

RESULTS OF RELIABILITY EXPERIMENT

Participant	Files	Flows	Processes	Total
1	1	12	11	24
2	1	11	10	22
3	2	12	13	27
4	1	12	10	23
5	1	10	12	23
	Mean			23,0
	Standard deviation			2,1
	Coefficient of variation			0,1

Exhibit 4.1

following circumstances made precise measurement difficult in this case:

- Inevitable ambiguities in the specifications could not be resolved by reference to the implemented system.

- Improvements to the design appeared possible, but it was not certain whether these would be implemented.

- Most of the participants were not previously familiar with the system.

In judging the results of the experiment it has therefore been accepted that this situation represented a "worst case", with more ambiguities than in a normal situation. Bearing this in mind the following conclusions were drawn:

- The definitions of files, flows and processes given in Chapter 2 are sufficiently clear to allow quantification of these attributes in an operational situation.

- Using these definitions, the physical dimension of data processing software can be determined with a reasonable degree of reliability. This provides strong support for (though clearly no rigorous proof of) the reliability of the proposed measure of efficiency.

- In a well defined situation (e.g. within a particular company) the reliability of the measure would probably be much better, as uniformity of definitions and practices and full understanding of the system could be assured.

- Total reliability of the measurement of any attributes of software is illusory. Any measure of efficiency of software development will therefore suffer from a degree of unreliability and this must be borne in mind when conclusions and inferences are drawn. The user of a specific measure of efficiency should at all times be made aware of a degree of (un)reliability of the measure.

- More insight into the reliability of the proposed measure of efficiency might be obtained by repeating the above experiment under different circumstances, with more observers and different systems. This would be both interesting and desirable, but is considered to be beyond the scope of this study.

4.4 THE SENSITIVITY OF THE MEASURE

Sensitivity indicates the ability of a measure to register small differences. This is generally desirable, provided that validity and reliability are assured. As long as validity and reliability are dubious, there is obviously little merit in sensitivity. Also, great sensitivity and the resultant precision may often not be required for decision-making.

The sensitivity of the proposed measure of efficiency of software development does not appear to pose a problem. The resources used are to be expressed in Rands (or, when this is a problem, in thousands of Rands) which should provide a sufficiently fine scale. The results of the development process are to be expressed in a function of flows, files and processes, which will yield numbers expected to range between, say, 5 and 100 for typical data processing systems. This should provide perfectly adequate distinctions. Admittedly, the sensitivity of a measure based on lines of code would probably be greater.

Greater validity and reliability of the proposed measure is however, considered to be more important than a loss in sensitivity. Moreover, in the current state of development, the data processing profession is not in a position to take advantage of a very sensitive measure of efficiency, but will be better served by a measure which is valid, reliable and satisfies the principles laid down in section 3.3.

4.5 CONCLUSION

In the previous chapter a measure of efficiency of software development was proposed and in this chapter validity, reliability and sensitivity were identified as important criteria for the acceptability. It was established that special survey research is required to test the validity of the measure. An experimental design to perform these tests and at the same time gather other worthwhile information is proposed in the next chapter. A separate experiment indicated a sufficient degree of reliability, while it was shown that the sensitivity should be adequate.

REFERENCES - Chapter 4

1. Jacoby, J.: Consumer research: A State of the Art Review, Journal of Marketing, April 1978, P.91.

2. Stevens, S.S.: Measurement and Man, Science, 21 February 1958, p

3. Thorndike, R.L. and Hagen, E.: Measurement and Evaluation in Psyschology and Education, 3rd edition, John Wiley and Sons, New York, 1969, p

4. e.g. Walston, C.E. and Felix, C.P.: A method of Programming Measurement and Evaluation, IBM Systems Journal, nr 1 1977.

5. Tull, D. and Albaum, G.: Survey Research: a Decisional Approach, International Textbook Company, Aylesbury, England, 1973, p.91.

6. Meyer, J.H.F.: The Application of Educational Technology to Selected Areas and Disciplinesin University Training, Doctoral Thesis, The University of the Witwatersrand, Johannesburg, 1974.

7. Osgood, C.E., Suci, G.J., Tannenbaum, P.H.: Measuring Meaning, University of Illinois Press, Urbana, 1957, p.141.

8. Shaw, M.E., Wright, J.M.: Scales for the Measurement of Attitudes, McGraw Hill, New York, 1967, p.18.

9. Tull, D., Albaum, G.: op.cit., p.92.

10. Tull, D., Ablaum, G.: ibid., p.92.

11. Tull, D., Ablaum, G.: ibid., p.94.

12. Churchill, G.A.: A Paradigm for Developing better Measures for Marketing Constructs, Journal of Marketing Research, February, 1979.

13. Meyer, J.H.F.: op.cit., p.96.

14. Tull, D., Ablaum, G.: op.cit., p.94.

CHAPTER 5

RESEARCH METHODOLOGY

CHAPTER 5

RESEARCH METHODOLOGY

5.1 INTRODUCTION

In previous chapters the foundations were laid for the analysis of the software development process and based on these, as well as on practical considerations, a unit of measure for the efficiency of software development was proposed. This measure has been specifically constructed to cover the totality of software development rather than one of its phases, such as programming. It is felt that this outlook is important, as it will focus the attention of management on the total process, rather than on a particular aspect which might eventually turn out to be expendable or easy to automate cheaply. Another important aspect of the measure is that it is firmly rooted in systems theory, which affords a degree of validation, as explained in paragraph 4.2.

The validity of the proposed measure remains, however, to be established empirically by testing various constructs. The specific formulation of the model and the value of some of its parameters also need to be assessed. It will furthermore be interesting to obtain an indication of current efficiencies, practices and techniques in software development. Proper indications can only be obtained by careful research and are likely to be of great value for the management of software development, as is underscored by by Dolotta et al.

> "Although there is an intuitive feeling that the data processing industry's growth will be limited by the inability to produce the necessary software, unless there is nearly an order of magnitude

> improvement in productivity, there exists no way to either substantiate or refute that feeling today. Then years from now, if nothing more than intuition is available to guide the industry's choices, the growth of the industry will be stunted". (1)

As the above quotation indicates, research into the efficiency of software development is sorely needed. This research has, however, been hampered by several circumstances. Firstly, the state of our knowledge of software development is still rather limited and few scientifically tested concepts are available. Secondly, the data processing profession has developed very rapidly and new practices have been introduced before the previous ones have been properly tested. Finally, software development in a data processing context is very difficult to analyse in laboratory-type experiments because it involves the capabilities of, and interaction between, a variety of people in practical work situations. The effort involved in a controlled experiment can be appreciated by the $444 000 budget of a research project proposed elsewhere.(2)

It is therefore clearly pertinent to attempt to verify the theoretical concepts advanced in the previous chapters through practical research. This must be field research, directed at the data processing work situation, but it must be carefully designed to ensure the reliability of the results. In this chapter plans for a survey of resources used for, and results obtained by software development projects are therefore methodically developed.

Selltiz et al.(3) distinguish two major types of research, notably exploratory or descriptive studies and studies testing causal hypotheses. The purpose of the first type of study is described as:

- to gain familiarity with a phenomenon or to achieve new insights into it, often in order to formulate more

precise research problems or to develop hypotheses.

- to portray accurately the characteristics of a particular individual, situation or group (with or without specific initial hypotheses about the nature of these characteristics)

- to determine the frequency with which something occurs or with which it is associated with something else.

This type of study requires an exploratory research design with emphasis on measurement and identification and attention to flexibility in the data collection.

The purpose of the second type of study is described as:

- to test a hypothesis of causal relationship between variables.

This type of study requires a careful design and completely controlled conditions in order to allow inferences about causality.

It is clear that the present study will be of the first type as it is intended to corroborate rather than to prove the acceptability of the proposed measure and to demonstrate the plausibility of the model, rather than prove its uniqueness. Furthermore, the difficulty in controlling the environment would make studies of the second type extremely difficult. The emphasis in the following research will therefore be on the promotion of insight through identification, classification and measurement.

5.2 OBJECTIVES, SCOPE AND LIMITATIONS OF THE FIELD RESEARCH

The purpose of the field research is to provide insight in the efficiency of software development in a data

processing environment through the collection of data concerning resources absorbed and results produced by software development projects. This research will further be designated as the survey. The outcome of the survey, together with theoretical arguments advanced in the previous chapters and with the test of reliability described in paragraph 4.3.1 should provide a fair assessment of the acceptability of the proposed measure of efficiency. The specific objectives of the survey may be related to the objectives 4 and 5 of the study as a whole and to problems and requirements identified in previous paragraphs. These objectives are:

1. To test the validity of Model I in Chapter 3 as a measure of efficiency by means of various constructs detailed in paragraph 4.2.1 and verify whether any other models give significantly better results.

2. To determine the availability of software development costs and the relative importance of the various cost elements in present practice, as suggested in paragraph 3.4.1.

3. To use the proposed measure of efficiency and other characteristics to describe and compare the efficiency of a number of typical data processing software development projects.

The survey will be confined to software development projects of a data processing nature, as defined in Chapter 2. For obvious practical reasons, it will be limited to South African organisations. These organisations should have reached a roughly similar level of data processing maturity, as defined in Chapter 2. The projects to be included must be fully implemented and working to satisfaction. Finally, the organisations and projects must, as a group, be reasonably representative of well-considered

practice in data processing software development.

An investigation of the total population under consideration i.e. all organisations performing software development at the intended levels would be impossible for various reasons, namely:

- The total population can only be roughly identified: though lists of computer users are available, these are not necessarily complete, while it is also not certain to what extent all computer users are involved in data processing software development.

- Non-response would make coverage of the total population illusory.

- Time required would make the approach of the total population unrealistic.

Random sampling of the population must be rejected for essentially the same reasons and it has therefore been decided that the survey should be based on a sample which may be expected to represent normal, well-considered practice amongst a typical group of target organisations.

Selltiz et al.[4] have described this type of sample as a purposive sample which is defined as a sample in which good judgement and an appropriate strategy are used to hand-pick cases to be included and thus develop samples that are satisfactory in relation to one's need.

Zetterberg[5] comments on this procedure:

> "from the summit of theory, a one-storey building with data does not look very different from a fifteen-storey building with data"

It must be stressed that because of this construction of

the sample there is no scientific basis for extending the conclusions reached beyond the limitations of the sample.

5.3 SAMPLE DESIGN

The two important aspects of the design of a sample are the selection of the sample members and the collection of the data. Problems concerning the collection of the data will be dealt with in paragraph 5.5.

Concerning the selection of the sample members, Tull and Albaum[6] list the following questions:

- What type of sample is to be used
- What is a practical size for the sample
- What is the appropriate sampling unit
- What frame is available
- How are refusals to be handled.

These questions will be answered in the following paragraphs.

5.3.1 Type of sample

The purposive sample has been chosen before as the most suitable, bearing in mind the exploratory and descriptive nature of the survey, the lack of generally accepted definitions and the geographical dispersion of the population.

Amongst the potential factors that may have a bearing on efficiency there are two that can easily be assessed for all organisations in the population. These factors, namely

the make of computer used and the sector of industry of the organisation, are therefore used to stratify the sample proportionally to the population. A basis for this stratification was provided by the Computer Users Handbook[7] which lists the users of various types of computers, and the Computer Survey of the Bureau of Statistics[8] which gives a breakdown of computer users per sector of industry. The stratification of the sample and the population according to these two criteria is shown in Exhibit 5.1. The 10% of the organisations classified as "other" in the sector breakdown was assigned to service bureaux, as these handle a proportionally large (but not precisely known) volume of data processing work for a variety of customers. The underrepresentation of users of computers classified as "others" results from the fact that these are generally mini-computers. In many cases data processing, as defined for this study, is not the main purpose of these installations and consequently such organisations would not be appropriate respondents.

STRATIFICATION OF POPULATION AND SAMPLE

	Estimated Population %	Sample %
SECTOR		
manufacturing	37	35
commerce	18	20
finance	22	25
public administration	10	10
others	13	10
COMPUTERS USED		
IBM	27	30
ICL	25	30
Burroughs	10	15
NCR	10	10
others	28	15

EXHIBIT 5.1

Further stratification of the sample was not deemed necessary. It has notably not been considered important to provide for proportional representation of geographical areas, as geographical location is not expected to have a large effect on the efficiency of software development. Practical considerations, therefore, mainly determined the area from which the sample members were selected.

5.3.2 Size of the sample

As it is not the intention to draw statistical inferences about the population from the sample, the sample size could largely be determined by practical considerations. Bearing in mind the principles of a purposive sample, a sample size of twenty was deemed sufficient to build up a reasonably representative cross-section of organisations to be interviewed. A much smaller number would not allow the desired stratification, while a much larger sample would be outside the limits of time and resources available.

5.3.3 Sampling unit, sampling frame and handling of refusals

The appropriate sampling unit was defined to be all organisations that have reached maturity level two or three in the use of data processing (as defined by Nolan[9]). A workable sampling frame was found in the Computer Users Handbook[10]. As a guide in choosing organisations of the correct maturity level, it was verified by means of a list of computer users in 1975[11] that all organisations selected had at least five years dp experience. On selection of the sample, it appeared that in most strata the choice of suitable organisations was fairly limited. Where a choice was possible, it was ensured that both large and small organisations were covered. In each organisation selected, the data processing manager was

chosen as the principal respondent, but it was accepted that most of the details to be requested would be supplied by another person, e.g. a project leader. To handle refusals, a back-up organisation was chosen in each stratum.

The sample was checked with a senior data processing consultant who considered it to be reasonably representative for the target population described above.

5.4 ADDITIONAL DATA TO BE COLLECTED

Once it was decided that a substantial effort would be made to approach a number of organisations and investigate aspects of their data processing practices, it appeared attractive to use this opportunity to collect some additional data. Notably, data concerning the factors likely to influence the efficiency of software development were considered to be of great interest. Data about the incidence of such factors, i.e. about practices and procedures actually used in software development are not readily available. Better definitions of such factors and insight into their real incidence would, in conjunction with a good measure of efficiency, probably lay the foundation for improvements in the efficiency of software development. In addition, it was therefore decided to use the survey to collect some exploratory data concerning the factors influencing efficiency. Substantial effort was devoted to the analysis of these factors and the main categories were defined as tools, techniques, personnel, organisational, and extraneous factors. However, as the analysis of these factors is not part of this study, it is not further discussed.

The results of the section of the survey devoted to the factors influencing efficiency will also not be analysed in detail in this study. Some elementary tabulations will

be presented in Chapter 6 but it is intended to use these data later as a basis for further studies.

5.5 DATA COLLECTION

A great variety of techniques exists for securing information in a survey. Parten[11] lists the following:

- personal interview
- observational methods and recording devices
- telephone interview
- mail questionnaire
- radio appeal
- panel technique.

Certain of these techniques, notably telephone interviews and observational methods are clearly unsuitable, as the situation is too complex to explain over the telephone and does not lend itself to observation. Under certain circumstances, panel techniques and radio appeal could, in a modified form, be of some interest. It would notably be conceivable that a central body (e.g. the CSSA) might organise a panel of system developers and monitor their progress. No such panel exists at the moment and the initiative would be outside the scope of this study.

Instead of a radio appeal, an appeal could possibly be made to a wide audience through a popular journal. This approach would, however, introduce insurmountable problems of validity and reliability.

The realistic choices for this research were therefore personal interviews or mail questionnaires. The following

discussion is based mainly on Selltiz et al.[13] who analyse the details of the various instruments for data collection.

The main difference between mail questionnaires and personal interviews is that in the latter case there is a greater opportunity for flexibility since questions can be rephrased and additional explanation provided. Also, the observer has the opportunity to ask for, or look for, proof of the answers supplied. This improves the reliability of the results.

The main advantages of mail questionnaires are that they are quicker, easier and less costly to administer. In simple cases, the standardised working may also contribute to uniformity of measurement. In more complicated cases like the current survey, this may prove illusory as interpretations of the printed text will differ.

An important disadvantage of mail questionnaires is the possibility of non-response, whereas a personal visit is more likely to elicit the co-operation of all selected individuals.

The above reasons were considered to weigh heavily in favour of personal interviews for the present study. In personal interviews, it is, however, of utmost importance to ensure that all respondents are answering essentially the same questions. The categorisation of the replies must also follow the same pattern in each case. In a complex and many-sided survey like the present one it is therefore imperative that the interviewer works from a standard questionnaire.

5.6 QUESTIONNAIRE DESIGN

The questionnaire is intended to support the interviewing

process by giving structure to the interviews, ensuring the use of identical terminology and serving as a means of recording answers.

Guidelines for the detailed construction of the questionnaire were taken from the Chapters 5 and 6 of Clover and Balsey[14] and Appendix C of Selltiz et al.[15]

The following principles were followed:

- confidentiality was stressed in advance
- details of the interview situation were recorded
- for the main questions pre-coded answers were provided, followed by open-ended probes where applicable
- explanations of crucial terms were supplied in writing.

A copy of the research questionnaire is shown in Appendix I to this Chapter.

The first page of the questionnaire introduces the purpose of the survey and defines the type of data processing systems to be selected. The body of the questionnaire is divided into 3 sections:

Section A - collects resource and result details about a particular software development project.

Section B - requests information about the factors influencing the efficiency of the project described in Section A.

Section C - collects only result details about two other projects. which will be used for validation purposes.

The terminology used is explained on the last page of the questionnaire.

5.6.1 Details of the Content of the Questionnaire

Section A: Quantification of the Size of a System and the Resources used to Develop it

This section is intended to gather information on a data processing system as defined on page 1 of the questionnaire. The main purpose of this information is to allow calculation of the efficiency of the development of this system, using the measure proposed in paragraph 3.5.1. This will be used to test the validity of the measure and to describe practices.

Questions A1 and A2 serve to determine whether the system selected by the respondent is in fact in agreement with the limitations of the research. Question A2 is meant particularly to eliminate any systems that may not have achieved the common level of effectiveness postulated in paragraph 3.4.2.

Question A3 is straightforward and is intended to determine the duration of the development project as a descriptive characteristic.

Question A4 supplies the crucial data for measurement of the result of the development process. The data on lines of source code and the various categories of documentation are requested to describe practices and to serve as a basis for validation of the proposed measure.

Question A5 provides the quantification of the resources absorbed by the software development process. Where pertinent data were not available, estimates were accepted, but these were recorded as such. For question A5 as well as A4 all information was not always readily available during the interview. In those cases it was ascertained

that the definition of the information required was well understood and the respondent was asked to submit the information later.

Section B: Measurement of Factors Influencing the Efficiency of the Development of the System described in Section A

The purpose of this Section is to describe and quantify the circumstances which are likely to have played a role during the development of the system described in Section A. These data fall outside the scope of the present study but, as explained in paragraph 5.4, were collected to be used in subsequent studies.

The questions in this Section are grouped in categories, with the intention of providing a frame for discussion of related problems. Of necessity, many of the questions ask for a subjective assessment and the questions have therefore been carefully worded and the answers pre-coded to prevent interviewer bias. Many questions include both a rating and a probing part to allow quantification as well as searching for explanation and checking for correctness. At the end of each category of factors, further factors not included in the questionnaire but belonging to the same category are solicited.

Section C: Quantification of the Size of two other Systems

The purpose of this section is to collect data for one of the constructs proposed in paragraph 4.2.1 for the assessment of the validity of the proposed measure of efficiency. The procedure to be used has been defined as an amalgamation of face validity and concurrent validity.

Questions C1 to C3 request the definition of a system which is subjectively rated as larger and one which is subjectively rated as smaller than the system described in Section A of the questionnaire. The limitations of the type of system to be selected as specified on page 1 of the questionnaire also apply to these systems.

Question C4 quantifies flows, files and processes. These quantifications are used to calculate the size of these systems and compare them with those of the system described in Section A. The data on lines of source code and documentation are again used to describe practices and other tests of the validity of the measure of efficiency. To ensure homogeneity of interpretation and understanding, the lay-out of this Section is identical to the questions 1 to 4 of Section A and it is important that the respondent for this Section is the same as for Section A.

5.6.2 Reliability of the Data

The reliability of the proposed measure of efficiency and its constituent parts was explicitly discussed in paragraph 4.3 and a special test was described in paragraph 4.3.1. Further testing of this aspect is therefore not incorporated in this survey.

The reliability of the other data to be collected in this survey is also of importance and of some concern. Measuring lines of code and pages of documentation is not straightforward but subject to problems of interpretation and accessibility of the data. These problems have been discussed extensively by Jones[16] and Johnson[17]. It is, however, believed that within the confines of data processing systems as defined, and by the use of the personal interview technique, a reasonable degree of reliability can be obtained. Explicit verification of this

reliability is outside the scope of this study.

The determination of the incidence of factors influencing efficiency in Section B of the questionnaire might also suffer from some loss of reliability, due to incorrect recall or different interpretation. Where possible, the answers were therefore checked by inspection and by probing questions. The interview also gave latitude for explaining terminology where there was reason to suspect wrong interpretation. Reliability may therefore be assumed to be reasonably assured. Explicit testing of the reliability of the answers was again not deemed feasible in the context of this survey.

5.7 PLANNING OF THE DATA ANALYSIS

The analysis of the survey results is largely determined by its main objective, namely the testing of the validity of the measure of efficiency, as described in paragraph 4.2.1. Apart from this, the research is exploratory and the results sought are therefore insight, classification and association, rather than proven causal relationships.

It is also to be borne in mind that the measurements of the factors in Section B of the questionnaire will mostly result in values on ordinal or nominal scales only. As explained by Stevens[18] such data do not allow calculation of measures such as arithmetic mean or standard deviation. In this survey, only simple tabulations of these data will therefore be presented.

5,8 SURVEY PROCEDURE

An initial draft of the questionnaire was used in a pilot test. For this pilot test an international oil company

with offices in Cape Town was chosen. As a result of the pilot test several alterations were made to the questionnaire and to the interview procedure. The results of this pilot test were not included in the results of the survey.

To underline the importance of the survey and obtain maximum co-operation a personal letter was sent to the data processing manager of the twenty selected companies. This letter was followed up by a telephone call to make an appointment for a personal interview. At this stage, two of the originally selected companies had to be withdrawn from the survey. The reason given by the first company was that their systems had gradually grown into a totally inter-woven entity, and in recent years development work had consisted of consolidation and integration rather than development of well-defined systems. In the second organisation there had been a complete change of data processing staff. The systems developed by the old staff were largely abandoned and the new staff was busy developing new applications, but no systems had yet been completed. These two organisations were replaced by others from the same strata. The resulting list of participating organisations is shown in Appendix 11.

REFERENCES - Chapter 5

1. Dolotta, T.A. et al.: Data Processing 1980-1985, SILT Report, John Wiley and Sons, New York, 1976, p.101.

2. George, D. et al.: Predicting cost of change from design structure metrics, Software Engineering Notes, Jan. 1982, p.30.

3. Selltiz, C., Jahoda, M., Deutsch, M., Cook, S.W.: Research Methods in Social Relations, Holt Rinehart Winston, New York, 1967, p.50.

4. Selltiz, C. et al.: Ibid., p.520.

5. Zetterberg, H.L.: On Theory and Verification in Sociology, The Bedminster Press, New York, 1965, p.154.

6. Tull, D. and Albaum, G.: Survey research: A decisional approach, International Textbook Co., Aylesbury, England, 1973, p.45.

7. Finlayson, R.L. (ed): Computer users handbook 1977/1978, Systems Publishers, Johannesburg, 1978.

8. Department of Statistics: Computer Survey, Government Printer, Pretoria, 1978.

9. Nolan, R.L., Gibson, C.F.: Managing the four stages of EDP growth, Harvard Business Review, Jan/Feb 1974.

10. Finlayson, R.L. (ed): op cit.

11. Editorial: All SA computer users, Management, December 1975.

12. Parten, M.: Surveys, Polls and Samples: practical procedures, Harper and Row, New York, 1965, p.71.

13. Selltiz, C. et al.: op cit., p.238.

14. Clover, V. and Balsey, H.: Business research methods, Grid Inc., Columbus Ohio, 1974.

15. Selltiz, C. et al.: op cit.

16. Jones, T.C.: Measuring programming quality and productivity, IBM Systems Journal, No.1, 1978, p.40.

17. Johnson, J.R.: A working measure of productivity, Datamation, Feb., 1977, p.107

18. Stevens, S.S.: Measurement and man, Science, Feb.1958.

APPENDIX I

RESEARCH QUESTIONNAIRE

The objective of this research is to collect data concerning:

- the efficiency of the development of data processing systems (part A)
- the incidence of factors thought to influence this efficiency (part B)
- the validity of the unit of measure of efficiency (part C)

For the purpose of this research, data processing systems have been fairly narrowly defined. An abbreviated definition is as follows:

A data processing system is a section of a computerised system for managerial and operational control. It is reasonably self-contained and serves a well-defined function.

This definition typically includes various systems for financial control such as debtors, creditors and wages and various systems for logistics management such as inventory control, order processing and production progress control.

Specifically excluded are:

- Computerised systems for technical support e.g. Computer Assisted Design
- Computerised systems for decision support e.g. Financial modelling
- Very small systems e.g. one input, one output and one program
- Systems making use of a Data Base Management System.

A. QUANTIFICATION OF THE SIZE OF A SYSTEM AND THE RESOURCES USED TO DEVELOP IT

This part concerns a system which has been completed (ie implementation stage has been finalised) not more than 3 years ago. The choice is open to the respondent, within the bounds described on the front page of this questionnaire.

1. Brief description of the system

.....

.....

2. Is this system being used as intended

Yes	No

3. a) Date completed .....

b) Date started .....

elapsed time [ ] months

4. Quantification of system size. For definition of terminology - see Appendix

a) Number of permanent files [ ]

b) Number of flows [ ]

c) Number of processes [ ]

d) Estimated number of lines of program source code [ ]

e) Documentation in pages

- functional spec [ ]
- technical spec [ ]
- user guide [ ]
- operator guide [ ]

5. Resources Used for Development

a) manpower

	manmonth	cost
users		
analysts		
designers		
programmers		
d p support personnel		
other support personnel		

b) computer power

identification	quantity	unit	cost *
.....			
.....			
.....			

c) software

identification	quantity	unit	cost *
.....			
.....			
.....			

d) other

identification	quantity	unit	cost *
.....			
.....			
.....			

* as far as not included in other charges

B. MEASUREMENT OF FACTORS INFLUENCING THE EFFICIENCY OF THE DEVELOPMENT OF THE SYSTEM DESCRIBED IN SECTION A

Tools

1. Was an important hardware or software tool new to the development team

Y	N

Specify .....

2. What make of computer hardware was used for system development .....

3. What was the availability of the required computer capacity

on line	better than once a day	less than once a day

4. Was use made of special utilities or systems software e.g. for screen formatting or test data generation

Y	N

Which .....

5. Which programming language was used for development of the system

Cobol	RPG	Other	None

6. Was use made of special system generation software or a non procedural language

Y	N

Which .....

7. Any other tools .....

Techniques

8. Was use made of specific methods or analytical techniques for requirement specification and systems design

Y	N

Which .....

9. Was structured programming used

no	some form	strictly

Explain.....

10. Were standards covering most of the system development process enforced

Y	N

How .....

11. Any other techniques.....

Personnel Related Factors

12. Rate the familiarity of the development team with the subject matter of the system being developed

low				high
1	2	3	4	5

13. Average number of years of experience of development staff in their particular function ..... Years

14. Rate the intellectual ability of the development team

low				high
1	2	3	4	5

15. Rate the level of motivation of the development team

low				high
1	2	3	4	5

16. How many alterations occurred in the development team, i.e. how many of the team were replaced or withdrawn before their task was completed ☐ out of ☐

17. Any other personnel related factors .....
.....

Organisation, Communication

18. Rate the degree of user participation in the development of the system

low				high
1	2	3	4	5

Special mechanisms used.....
.....
.....

19. Was a chief programmer team used

no	some form	strictly

20. Was progress formally reported to an authority higher than the project leader

Y	N

Format .....
Frequency .....
Recipient .....

21. Rate top management support for this system

low				high
1	2	3	4	5

How was it expressed .....

22. Any other organisational factors .....

Extraneous Factors

23. Is a significant amount of development work done for third parties

Y	N

If so, what percentage ..... %

24. Are d p practices significantly influenced by a parent organisation

Y	N

25. For how many years has the organisation actively performed data processing system development ..... years

26. How many people are full time involved with data processing ..... persons

27. How many people devote virtually all their time to data processing system development

	Job Title	No. of Persons
in d p department		
in other departments		

28. Any other .....

C. QUANTIFICATION OF THE SIZE OF TWO OTHER SYSTEMS

This part concerns two systems with which the respondent is familiar. The systems must satisfy the criteria specified on the front page of the questionnaire, but do not need to be recently developed.

	A smaller System	A bigger System
1. Brief description	.....	.....
	.....	.....
	.....	.....
2. a) data completed	.....	.....
b) date started	.....	.....
elapsed time	☐	☐ months

3. Is this system being used as intended

	A smaller System	A bigger System
	Yes / No	Yes / No

4. Quantification of system size

	A smaller System	A bigger System
a) number of files		
b) number of flows		
c) number of processes		
d) estimated number of lines of program source		
e) documentation in pages		
- functional spec		
- technical spec		
- user guide		
- operator guide		

APPENDIX

Definition of Terminology for Quantification of System Size.

A. FILES

Files generally are collections of records, that are both logically and physically related but, for the purpose of this definition, are restricted to masterfiles i.e. files that are permanently maintained either by the system under consideration or by another system. This excludes transaction files, workfiles and report files which are considered to be features of the implementation rather than essential elements of the system. Only the number of masterfiles is considered and not their content or structure. Different generations or different sequences of a file are not counted as separate files.

The number of files is expected to be readily obtainable from the technical specifications of the system.

B. FLOWS

Flows are means of exchanging information. For the purpose of this definition they are restricted to external flows i.e. exchanges of information between the system and its environment. Excluded are therefore internal flows within the system as these are considered to be features of the implementation. Only the number of flows is considered, without regard to their content or structure. As flows are counted:

- each different type of input record that may be presented either through conventional unit record or through input files from other systems.
- each unique type of report that can be produced by the system.
- each screen format in an interactive system which is available for the presentation or acceptance of information.

It is assumed that the number of flows can be readily determined from the functional and technical specifications, specifically from a detailed system flowchart.

C. PROCESSES

Processes are logical or arithmetic manipulations of data. They are defined at a functional level, which means that the description of a process should define what manipulation is performed on the data, not how it is done. Typical examples of functional processes are: sorting, validating, report presentation, updating. Processes may coincide with programs, but it is quite conceivable that a physical program contains two or more processes.

Determining the number of processes in a system will require a degree of understanding of the system which should be obtainable from the functional and technical specifications.

APPENDIX II

LIST OF PARTICIPATING ORGANISATIONS

1. AECI
2. Argus
3. Cape Provincial Administration
4. Dorman, Long, Vanderbyl Corp. (Johannesburg)
5. Garlicks
6. ICL Service Bureau
7. Irvin and Johnson
8. John Thompson Africa
9. Management Computer Services
10. Mobil SA
11. Nedacom
12. Old Mutual
13. Parow Municipality
14. Printpak
15. Reckitt and Colman
16. Rex Trueform
17. Sanlam
18. Shell South Africa
19. Southern Life
20. Standard Bank - Corporate Systems Department (Johannesburg)

CHAPTER 6

RESULTS OF THE SURVEY

CHAPTER 6

RESULTS OF THE SURVEY

6.1 INTRODUCTION

During March 1980, interviews to collect the required information were arranged through the data processing managers of the 20 participating organisations. After a suitable system had been selected, the majority of the questions were normally answered by the project leader responsible for that project, often with the data processing manager or system development manager attending the discussion. In most organisations, some data collection regarding essential systems' characteristics had been done beforehand, which substantially reduced the time required for the interview.

A total of 36 hours was spent on the interviews, to which 33 people actively contributed. The interviews often led to a discussion of background issues which contributed to an understanding of the particular situation and ensured that the questions were understood and answered correctly.

In some instances, the data requested were not readily available. Where this involved crucial data concerning the determination of efficiency or its influencing factors, respondents supplied the data later by telephone. Where it involved supporting data such as lines of source code, the data were recorded as not available.

All data collection was completed by the middle of April 1980. In general it is felt that the survey was successful. It yielded not only the data required to achieve the objectives specified in paragraph 5.2, but also provided

valuable insights into prevailing software development practices and problems. The success of the survey may be attributed partly to the carefully chosen sample and partly to the personal approach, but mostly to the high level of interest in the subject of the study amongst data processing professionals.

In this chapter, the primary data resulting from the survey, namely the resources absorbed and results produced by a number of software development projects will be described. These data will be used to test the validity of the proposed measure of efficiency of software development and to determine the value of the parameters of the suggested model. Furthermore, a number of characteristics of the projects will be categorised and summarised.

Throughout the analysis of the results, reference will be made only to systems and never to the individual organisations supplying the data. This is in keeping with the guarantee of anonymity given to the respondents, without which the same results could not have been obtained.

6.2 SUMMARY OF BASIC RESULTS

6.2.1 All systems investigated

Data characterising the results of software development projects were requested in Section A, questions 1-4 and in Section C of the questionnaire. Each organisation was asked to supply data about three systems: full details about one system and minimal data about two others. In several cases the data requested in Section C were not, or only partly, available which resulted in a reduction of the sample size for some of the analyses. Table 6A (at the end of this Chapter) records the following data about the 53

remaining systems: purpose and identification, elapsed development time, numbers of flows, files and processes, lines of source code, pages of documentation (four categories and a total). The column labelled 'size' contains a simple addition of flows, files and processes, which may serve as a preliminary indication of the magnitude of the systems. A frequency distribution of this variable is shown in Exhibit 6.1. It shows a very skew distribution, with many relatively small and a few relatively large systems. This distribution is believed to be in keeping with South African data processing practice.

In paragraph 3.3.4 it was suggested that attributes forming part of the measure of results produced by software development should not be highly correlated. The observed correlation between the main candidates (flows, files and processes) has been tabulated in Exhibit 6.2. It is difficult

DISTRIBUTION OF THE SIZE OF ALL SYSTEMS

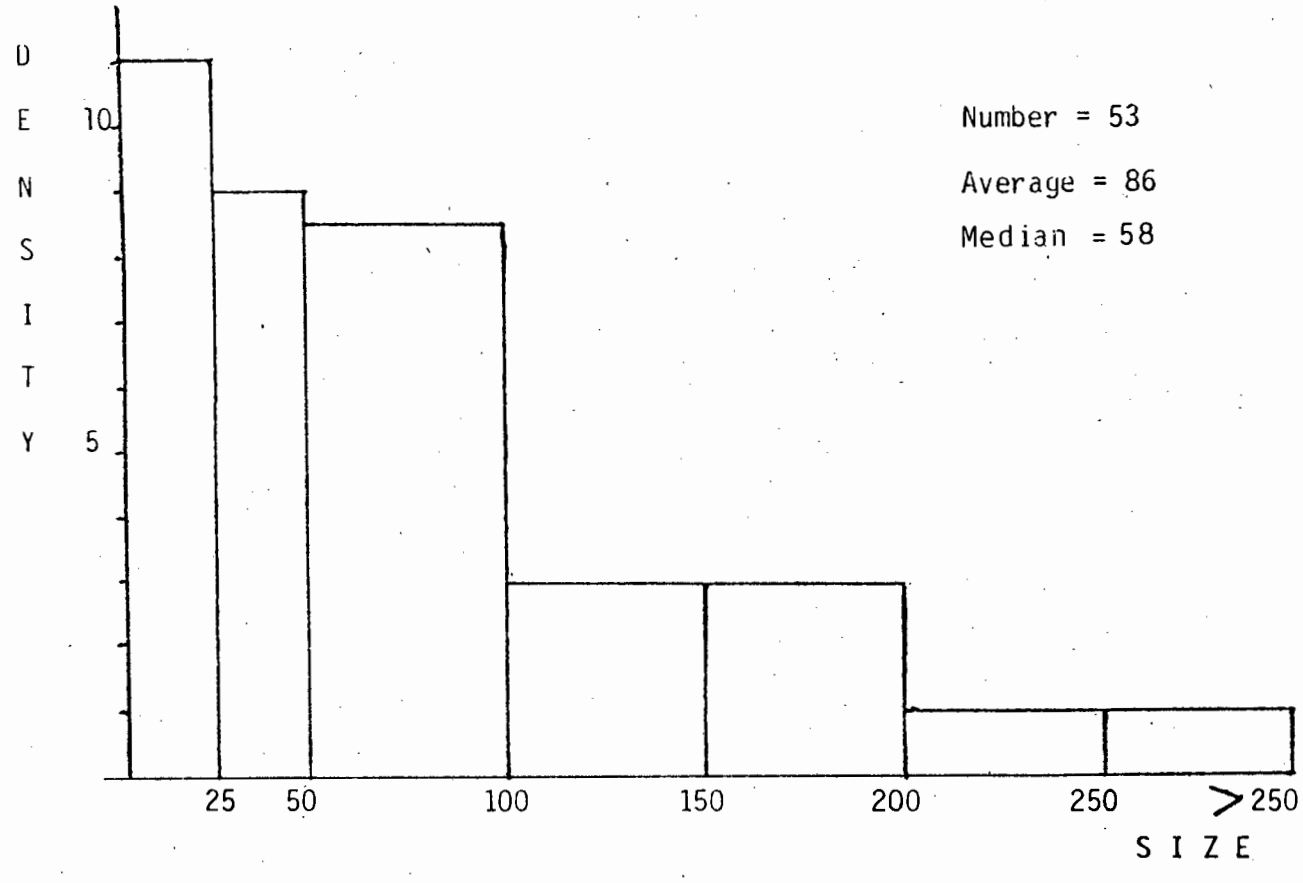

EXHIBIT 6.1

to determine what precisely constitutes high correlation and some correlation must obviously be expected. The observed correlation coefficients appear, however, sufficiently low to accept that none of these characteristics is completely determined by the others.

INTERCORRELATIONS BETWEEN STRUCTURAL ELEMENTS

	flows	files	processes
flows	-		
files	0,33	-	
processes	0,49	0,53	-

EXHIBIT 6.2

Lines of source code (source in Table 6A) were available for most, but not all, of the systems. Correlations between this characteristic and others will be discussed later.

Documentation was found to be a problematic characteristic. Most of the respondents were familiar with the four categories of documentation investigated, but in 17 cases the functional specifications were not produced during development of the system. The reason given was mostly lack of time or user co-operation. For more than half of the systems some of the documentation was either not produced (NP), or the data were not available (NA). The number of pages of documentation was therefore not considered further as a viable measure of the results produced by software development.

Furthermore, the following facts may be noted from Table 6A:

- The variety of purposes served by the data processing

systems surveyed, underlines the diversity of applications of data processing systems and reinforces the representativeness of the sample.

- The majority of systems were developed in a limited elapsed time. Only six out of 49 took more than a year to develop.

- In two cases the number of lines of source code was listed as not produced (NP). In both cases the reason was that a significant amount of development was done using a non-procedural language where no source code is written.

6.2.2 The 20 systems in Section A and B of the Questionnaire

For 20 systems (one in each organisation surveyed) data were collected concerning both results produced and resources absorbed by software development. The information was recorded in Section A of the questionnaire and is summarised in Table 6B. The distributions of flows, files, processes, lines of source and pages of documentation for these 20 systems do not differ much from the corresponding figures for all 60 systems presented in Table 6A, as is demonstrated by the parameters at the bottom of both tables.

Data concerning resources used were collected in question A5. Table 6B shows that none of the organisations specifically recorded the cost of software products used in software development, while only two reported some costs under "other". In two organisations total project costs were available but no breakdown in d.p. manpower and computer utilisation could be given. In two other organisations the manpower cost was available, but there appeared to be no basis for the estimation of computer utilisation costs. In these cases the average ratio of manpower costs to computer costs was used to calculate the

total project cost. The average percentages of the various categories of costs reported in the sample are shown in Exhibit 6.3 below.

The development cost figures required were not always readily available but in many cases ad-hoc procedures allowed rapid calculation. The degree of availability of the various cost figures is summarised in Table 6C. This Table shows that in most cases at least the number of man-months was recorded, but for computer utilisation ad-hoc calculations often had to be made. During the interviews these calculations were, however, extensively questioned and cross-checked and they are therefore deemed to be sufficiently reliable.

PERCENTAGES OF TOTAL SOFTWARE DEVELOPMENT COSTS

User manpower	10%
DP Manpower	68%
Computer power	22%
Software costs	-
Other	< 0,1%

EXHIBIT 6.3

6.3 VALIDATION OF THE PROPOSED MEASURE OF EFFICIENCY

It was found in Chapter 3 that a simple linear model of the form

$$E = \frac{Fl + Fi + Pr}{C} \qquad (I)$$

provides the most attractive representation of the model of efficiency sought. It remains to establish the validity and acceptability of this model. As stated in paragraph 3.5.2,

if this model is found to be valid and acceptable, other models will only be briefly considered.

6.3.1 The Validity of Model I

Tests to assess the validity of the various models proposed in Chapter 3 were suggested paragraph 4.2.1, and the survey was organised to produce data for each of these tests. The following results were obtained:

- Validation by comparison of Resources Absorbed and Results Produced:

It was argued in paragraph 4.2.1 that a positive, but not very high correlation should be expected between resources absorbed and results produced by software development. The relationship observed in the survey data is shown in Exhibit 6.4 and 6.5.

REGRESSION OF COSTS AND RESULTS

(Model 1)

Model	Regression function	Sum of squared deviations
1	C = 380 (Fl+Fi+Pr)	$2,92 * 10^9$

EXHIBIT 6.4

It is to be noted that the regression function described in Exhibit 6.4 was "forced through the origin" i.e. does not include a constant term, in accordance with model (1). The correlation coefficient as normally defined (i.e. the variation around the line divided by the total variation of

SCATTER DIAGRAM OF COST VERSUS RESULTS

(Model 1)

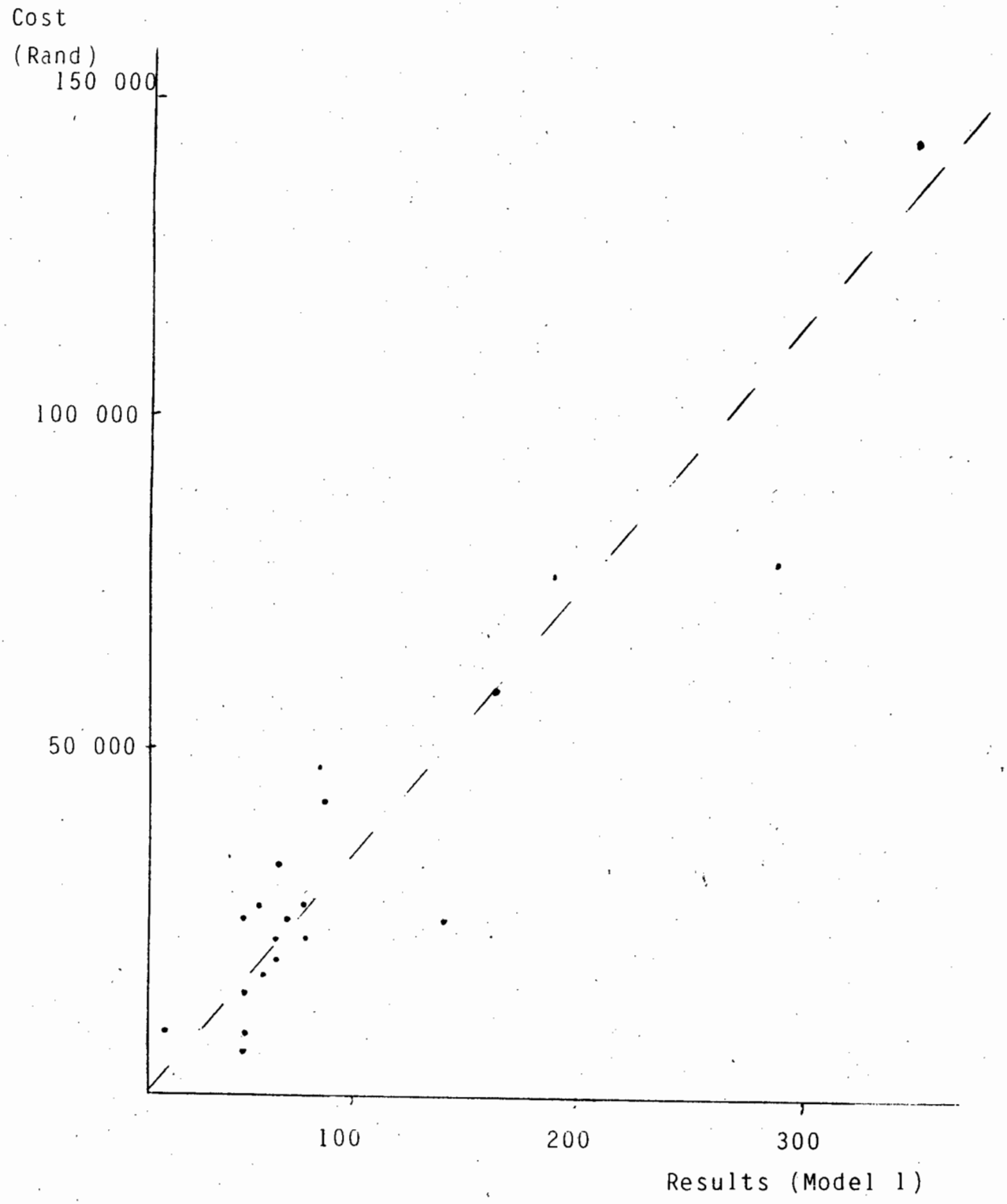

EXHIBIT 6.5

function of flows, files and processes.

This provides another indication of the validity of the proposed measure.

CORRELATION BETWEEN LINES OF SOURCE CODE AND THE SUM OF FLOWS, FILES AND PROCESSES

model	no. of cases	correlation coeff.	F-Ratio	level of significance
1	42	0,83	88,6	0,99

EXHIBIT 6.6

- Validation by expert opinion:

The survey was constructed to be reasonably representative for well-considered data processing practice, and in each of the organisations the data processing manager or software development manager was involved in answering the questionnaire. These people clearly represent a body of substantial expertise on software development. To each of these, the purpose of the survey and the concept of expressing efficiency as a function of flows, files and processes and total development costs was explained. In most cases an enthusiastic reaction was received and in all cases further co-operation with the survey (often involving substantial effort) was obtained. It was not attempted, and is not considered feasible, to quantify the level of support for the proposed measure of efficiency. The positive attitude of 20 experts and their willingness to co-operate are, however, considered to constitute a substantial validation by expert opinion.

- Validation by means of a specially designed test:

It was suggested in paragraph 4.2.1 that the validity of the measure could be tested by comparing the intuitive ranking of a number of systems with their ranking according to the proposed quantitative measure. In Section C of the survey, respondents were asked to identify systems considered to be either larger or smaller than the system described in Section A and to supply the necessary characteristics to measure these systems. As not all respondents were able to describe both a smaller and a larger system within the constraints imposed by the survey, this resulted in 34 pairs of (relatively) small and (relatively) large systems. The complete measurement data for these 34 pairs are shown in Table 6D.

The information in this table is summarised in Exhibit 6.7 below. Exhibit 6.7 shows that in all 34 cases data were available to compare the results of the application of the measure of flows, files and processes with the subjective ranking according to size. In all cases the calculated values preserved the subjective ranking. For lines of source code and documentation, fewer comparisons were possible but the subjective ranking was not always preserved. This provides a substantial degree of validation for the sum of flows, files and processes as a measure of system size.

In summary, it has been shown that four tests designed to assess the validity of the proposed measure all resulted in strong support for the measure as expressed by model (1). It is therefore concluded that model (1) may be accepted as a valid measure of efficiency. As a result, the other models proposed in paragraph 3.5.2 will only be cursorily inspected below.

RESULTS OF A TEST OF CONCURRENT VALIDITY PERFORMED ON THREE POSSIBLE MEASURES OF SYSTEM SIZE

Measure	Cases Available For Comparison	Subjective Ranking Preserved	Subjective Ranking Not Preserved
Flows, files, processes	34	34	0
Lines of source code	25	24	1
Documentation	22	21	1

EXHIBIT 6.7

6.4 ALTERNATIVE MODELS

In paragraph 3.5.2, three other models were discussed. All three models contain parameters which need to be estimated empirically. This estimation process is rather problematic as there is no obvious method to calculate the required parameters. Also, limited data (20 cases) are available, while from this amalgam of cases at best only "average" values of the parameters can be obtained. Accepting these limitations, the best way to estimate the parameters appears to be to assume a fairly stable relationship between development costs and the size of the resulting software, and determine this relationship by means of regression analysis. In the case of models II and III this requires multiple linear regression, while for models III and IV log conversion to achieve linearisation is necessary. The results of these analyses are shown in Exhibit 6.8 (which incorporates Exhibit 6.4 for the sake of comparison).

It is to be noted that model II results in a negative constant term, implying a negative "cost" for very small projects. This makes this model very unattractive. Model III also leads to unacceptable results, notably when either

flows, files or processes are zero, as is the case with some projects. The exponent in model IV is so close to one that it seems hardly worthwhile to consider this model separately.

REGRESSION OF COST AND RESULTS

Model	Regression function	Sum of squared deviations
I	$C = 380(Fi+Fl+Pr)$	$2,92 * 10^9$
II	$C = -584+150Fi+26Fl+596Pr$	$2,33 * 10^9$
III	$C = 1077Fi^{0,7} * Fl^{0,41} * Pr^{0,48}$	$1,14 * 10^8$
IV	$C = 372(Fi+Fl+Pr)^{0,99}$	$3,51 * 10^9$

EXHIBIT 6.8

To assess the validity of these models more formally, they may be subjected to the same tests as model I above:

- As in the case of model I, it would for the models II to IV also not be correct to use the correlation coefficient to express the goodness of fit. Simple (as opposed to multiple) correlation is inapplicable to model II and for models III and IV this would give the correlation of the linearised models, rather than of the original models. However, bearing in mind that the dependent variable (cost) is the same for all models, an appropriate unit for comparison is again the sum of squared deviations which is tabled in Exhibit 6.8. It can be seen from this Exhibit that the more complex models do not yield much improvement according to this measure. Although the models II and III apparently fit the cost data more closely than model I, this is almost certainly due to

the fact that these models have respectively 4 and 3 parameters, while model I has only one parameter that may be varied.

- The regression functions for models II to IV shown in Exhibit 6.8 imply an expression for the results of software development for each of these models. The relationships can be calculated between these expressions and the alternative measure, i.e. lines of source code. As this is a straight-forward relationship between two variables (lines of source code versus the result of the particular model) simple linear correlation indicates the closeness of the relationship. As argued before, not too much faith should be put in this correlation, as lines of source code is itself a suspect measure, but at least some positive correlation is expected. The result of these correlations for models II to IV (using the model parameters defined in Exhibit 6.8) is shown in Exhibit 6.9 (which incorporates Exhibit 6.6). As can be seen, significant correlations were obtained in all cases, and the correlation coefficients were of the same order of magnitude as that for model I shown in Exhibit 6.6.

CORRELATION BETWEEN LINES OF SOURCE CODE AND FUNCTIONS OF FLOWS, FILES AND PROCESSES

model	no. of cases	correlation coeff.	F-Ratio	level of significance
I	42	0,83	88,6	0,99
II	42	0,87	124,5	0,999
III	42	0,86	199	0,99
IV	42	0,83	88,6	0,99

EXHIBIT 6.9

- The test of validity based on expert opinion applies as much to the other models as to model I, since no specific formulation was suggested to the respondents.

- The special test (see Exhibit 6.7) provides the same results for the models II to IV as for model I.

It can be concluded that from the above that formally the models II to IV might also be considered to be valid expressions of the efficiency of software development. The evidence supporting these models is however not more impressive than that for model I. In addition, these models are rather more complicated than model I and require the estimation of a number of parameters, while there is no theoretical foundation for these parameters. Finally each of these other models has certain inherent weaknesses, as pointed out in Chapter 3 and in the beginning of this paragraph.
Model I is therefore definitely preferred and, as its validity has been shown to be strongly supported, it is considered to be a good representation of efficiency and will be used as such in the rest of this study.

6.5 FURTHER ANALYSIS OF SURVEY DATA

The most interesting results of the survey are the efficiencies obtained in the development of the various systems. These efficiencies have been calculated (using model I) for the 20 systems for which data were collected in Section A of the questionnaire. The resulting figures (multiplied by 10 000 for convenience) are shown in Table 6E. The inherent similarity of the efficiencies is clearly shown by their clustering around the regression line in Exhibit 6.5. However, from a different point of view one might be more interested in the observed differences in efficiency than in their similarity. A frequency distribution

highlighting the diversity of the efficiencies observed is shown in Exhibit 6.10.

DISTRIBUTION OF OBSERVED EFFICIENCIES

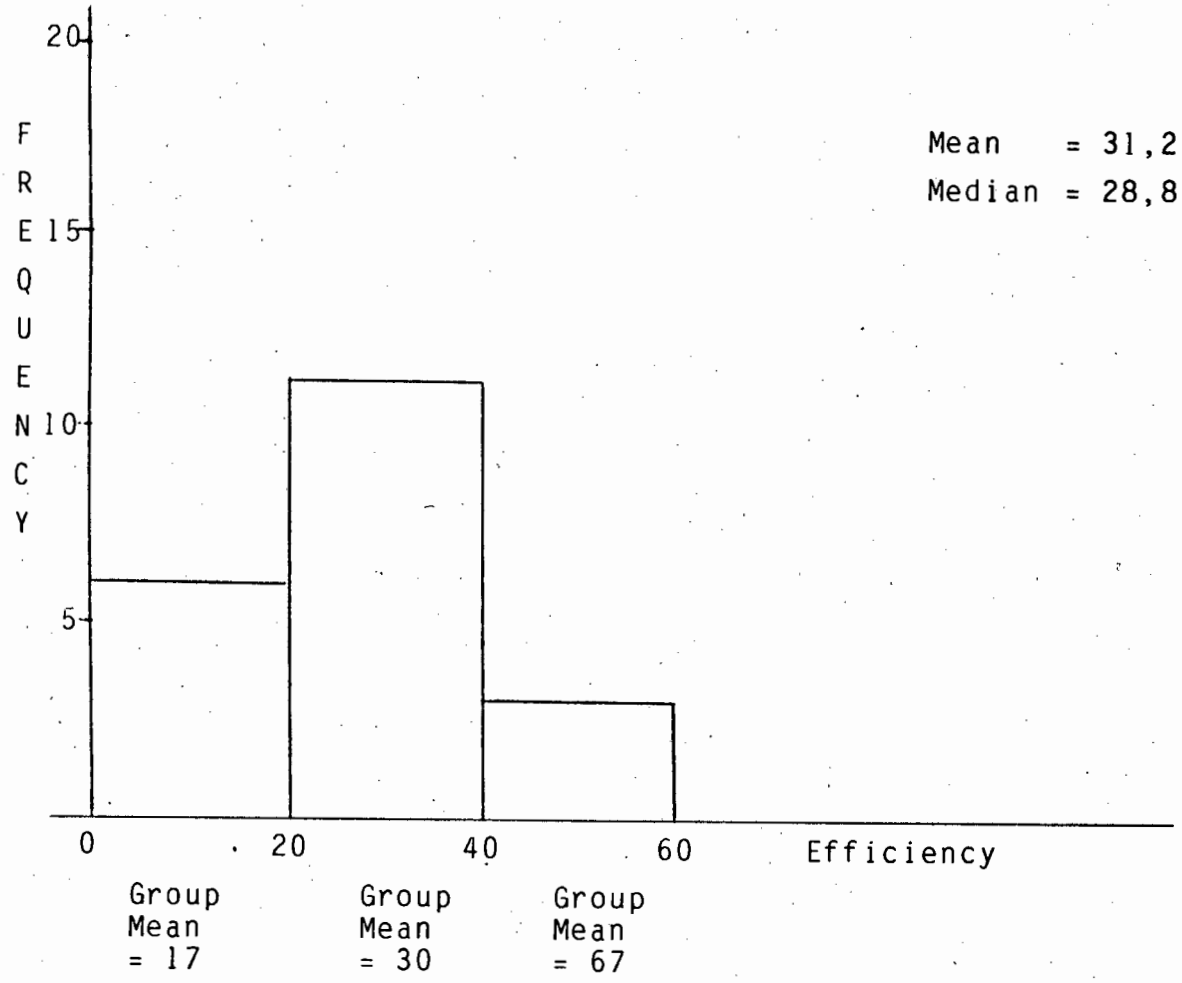

EXHIBIT 6.10

It is interesting to note the big difference between the lowest efficiency of 15,7 and the highest of 90,2: nearly a factor 6. To have shown the existence of this large difference is deemed to be an important result of this

survey. To check some of the high efficiencies, the data have been verified thoroughly with the respondents and found to be conservative, if anything. It is obvious that these high efficiencies are of particular interest, as they may provide useful indications for the rest of the industry.

It was suggested in Chapter 1 that it would be of interest to investigate to what extent efficiencies in software development differ among various types of organisations. Average efficiencies for each of the 5 sectors of industry distinguished in paragraph 5.3 were therefore calculated. The results are presented in Exhibit 6.11 and show, in fact, a remarkable uniformity between the sectors. (A somewhat larger difference was observed between the average efficiencies in the Cape Town (32,5) and Johannesburg (20,3) areas. The number of organisations interviewed in Johannesburg was, however, too small to attach much significance to this difference.)

AVERAGE EFFICIENCIES PER SECTOR OF INDUSTRY

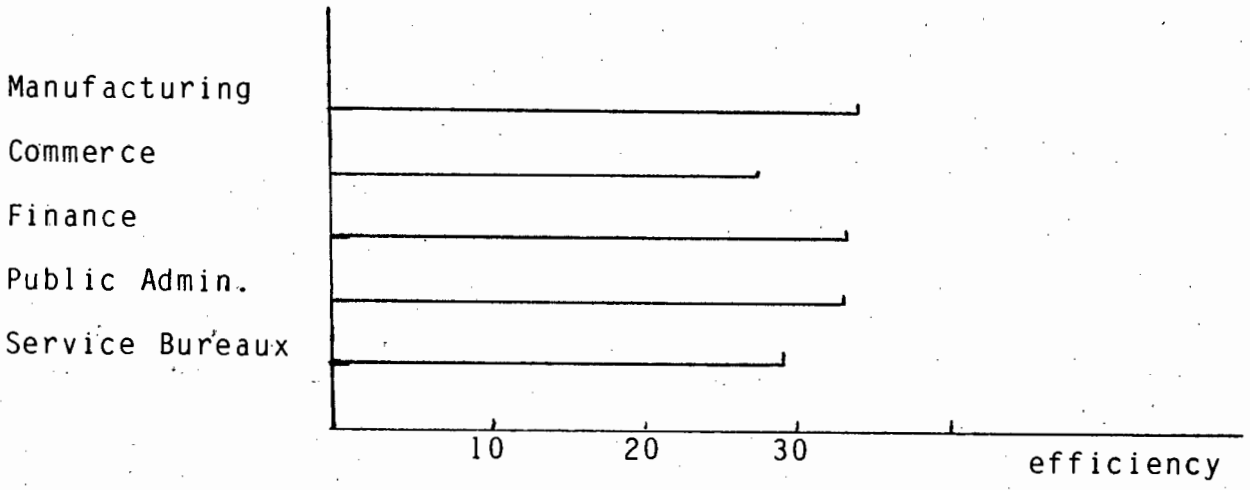

EXHIBIT 6.11

6.6 THE INCIDENCE OF THE FACTORS INFLUENCING THE EFFICIENCY OF SOFTWARE DEVELOPMENT

Section B of the questionnaire investigated the incidence

of a number of factors likely to influence the efficiency of software development. Analysis of these factors and of their relationship with observed efficiencies does not form part of the objectives of this study, and therefore only a brief summary of the main findings is included. The information obtained from the 28 questions in Section B of the questionnaire, together with background information about the sector of industry and the location of the organisation, is represented in Table 6F. The results are largely self-explanatory, but some salient facts are:

- A large proportion of respondents (13/20) were dealing with some novel tools during the development. This is in keeping with the dynamic image of the data processing industry, but it certainly contributes to the complexity of software development.

- Type of computer used, sector of industry and location were according to predetermined ratios.

- Only some use was made of utilities (8/20) and very little non-procedural languages (3/20). The type of utility most frequently mentiond (5/8) was source library and test data handling. The non-procedural languages used were Mark IV and Easytrieve (twice but in a minor way).

- The one additional tool mentioned was an in-house pre-compiler, along the lines of Meta-cobol.

- In only a few cases (5/20) was any use made of techniques for system analysis and design. Techniques mentioned were: work study (1), structured design (3) and decision trees/ tables (2).

- Standards were generally (18/20) found to be in existence and, though the pre-coded answer only made provision for yes/no answers, additional information requested made it

possible to distinguish between loose and strict enforcement of standards. It is revealing that in two cases no formal standards were found to exist.

- The other techniques mentioned were: skeleton programming (4) (i.e. the use of standard routines that are slightly adapted for specific applications), modular programming (2), structured walk-through (2) and code inspection (3).

- The average number of years of experience was in many cases not a very meaningful figure, as teams tended to be made up of some highly experienced and some inexperienced people.

- In most projects (14/20) there were no alterations to the team and the worst case was 2 out of 6. This contrasts with the often projected image of data processing professionals as job hoppers.

- The other personnel-related factors recorded were: first project of a new team leader in the organisation (2), sickness and absence of key personnel (1), much overtime (2).

- As far as Chief Programmer Teams were encountered, the chief programmer's task was always an amalgamation of design, programming and management. These teams were therefore classified as "some form" of Chief Programmer Team.

- In a significant number of cases (6/20) top management support was rated as low or average. As indications of this clearly unsatisfactory situation, were quoted: insufficient resources made available, delay in approval and haphazard definition of requirements.

- The other organisational factors mentioned were: the team

was brought in from the outside(1), a closely knit team(3) concurrent work of higher priority(1), strong supporting structure (1) two different user groups(1).

- Apart from the two service bureaux, three other organisations reported doing system development for third parties but each to an extent of less than 30%.

- The other factors mentioned were: the department was relocated twice(1), there was a departmental reorganisation (1).

The incidence of the factors reported above points to a wide variety of practices in software development. It is clear that some of the practices that have been strongly advocated in the literature as absolutely essential (such as the enforcement of standards and top management support) still cannot be taken for granted. More elaborate tools and techniques (such as system design techniques and non-procedural languages) were found to be only rarely used.

Further analysis fo the incidence of factors and their connection with efficiency would obviously be of interest but is outside the scope of this study.

6.7 REVIEW

In paragraph 5.2 three objectives of the survey were specified, namely:

1. To test the validity of model I as a measure of efficiency and compare it with that of other models.

   It was found in paragraph 6.3 that the validity of model I was amply supported by four different tests. The

evaluation of other models in paragraph 6.4 showed that these models were in principle feasible, but not preferable to model I.

2. To determine the availability of cost elements and their relative importance.

   The relevant results were summarised in Table 6C and Exhibit 6.3. The availability of cost data was shown to be a problem but not an insurmountable one.

3. To use the selected measure to describe and compare efficiencies attained.

   This was done in paragraph 6.5 and the results are likely to provide a basis for further research.

The objectives of the survey were therefore clearly attained. In addition, much insight was gained into software development practices and their efficiency, which will form the basis for further analyses and conclusions.

TABLE 6 A

PHYSICAL CHARACTERISTICS OF ALL SYSTEMS

Purpose of the system	Identification	Elapsed Time Months	Number of File	Flow	Proc	Size*	Lines of Source	Documentation Func	Tech	User	OPS	Total
Nature conservation control	Nature	12	2	16	24	42	7 000	40	150	30	15	235
Medical records	Medical	5	1	8	8	17	1 800	2	120	30	8	160
Matric results	Matric	18	4	34	38	76	18 000	90	220	100	1	411
Job costing & progress	Job cost	7	14	57	60	131	9 500	NP	NA	20	1	NA
Personnel administration	Pers ad	NA	5	20	28	53	2 117	NP	75	8	1	84
Advertising statistics	Advert	10	1	31	35	67	31 909	NP	161	37	60	258
Advertising contracts	Ad con	2	1	5	5	11	550	NA	NA	NA	NA	NA
Debtors control	Deb con	12	3	41	31	73	12 900	NA	NA	NA	NA	NA
Group life assurance	Gr life	11	5	32	40	77	36 000	77	240	38	24	379
Mortality investigation	Mortal	3	3	8	13	24	8 300	5	60	8	7	80
Pension administration	Pension	9	9	67	105	181	57 500	250	400	NP	60	810
Reconciliation	Recon	12	7	15	32	54	12 842	27	55	NP	20	102
Reciprocal business	Recipr	2	4	6	7	17	NA	12	42	20	10	84
Spot finance control	Spot fin	21	49	58	95	192	NA	250	360	100	100	810
Material utilisation	Mat util	12	10	44	99	153	54 000	47	27	42	6	122
Stores inventory	Stores	6	5	29	82	116	40 000	30	25	36	8	99
Workshop costing	Workshop	12	9	57	172	238	75 000	50	64	53	8	175
Stock transactions	Stock	7	9	15	17	41	6 018	15	112	18	14	159
Wages	Wages	NA	1	53	65	119	NA	NA	NA	NA	NA	NA
Work in progress	WIP	7	3	17	34	54	9 263	NP	60	13	44	117
Debtors	Debtors	3	2	11	14	27	NA	NA	NA	NA	NA	NA
Inventory control	Invent	15	4	23	72	99	NA	NA	NA	NA	NA	NA
Trimming control	Trim	2	4	3	3	10	2 190	NP	57	NP	2	59
Cloth utilisation	Cloth	NA	16	95	39	150	NA	NA	NA	NA	NA	NA
Sales/stock/debtors	Sales	7	5	48	17	70	8 900	NP	120	70	5	195
Financial reporting	Report	2	0	4	3	7	1 000	NA	NA	NA	NA	NA
Invoicing/stocks	Invoice	12	6	59	38	103	NA	NA	NA	NA	NA	NA
New business	New bus	15	6	50	21	77	NP	25	800	30	50	905
General ledger	Ledger	9	1	21	6	28	NA	NA	NA	NA	NA	NA
Members' accounts	Members	NA	4	74	31	109	NA	NA	NA	NA	NA	NA
Brokerage	Broker	5	4	19	15	38	5 730	11	79	2	9	101
Batch allocation	Batch	1	1	7	4	11	2 000	NP	24	NP	3	27
Beneficiary	Benef	3	4	44	17	65	8 200	NP	115	NP	12	127
Property administration	Prop	15	16	187	137	340	55 198	202	145	220	114	681
Assets control	Asset	7	1	38	26	65	17 184	25	1	22	7	55
Investments administration	Invest	10	9	139	60	208	40 648	110	13	32	41	199
Stock book	S. book	12	4	41	13	58	12 593	45	156	115	60	376
Fixed asset register	Fixed	4	3	20	18	41	8 665	NP	30	24	4	58
Pension fund	Fund	1	1	9	10	20	2 602	NP	15	5	3	23
Stock on consignment	Consign	2	6	23	35	64	8 848	NP	60	NP	5	65
Cost reporting	Cost	18	6	227	50	283	NP	79	450	50	50	629
Stock gains and losses	Gains	12	6	24	21	51	13 513	55	168	17	40	280
Equipment spares	Spares	3	3	13	8	24	2 886	20	60	NA	20	NA
Material balancing	Mat bal	12	5	32	16	53	14 503	150	400	110	60	720
Life insurance	Life	18	20	80	80	180	73 000	NP	1150	150	130	1430
Debit orders	Debit	4	1	8	11	20	5 700	NP	72	30	20	122
Contract accounting	Contr	24	11	110	75	196	80 000	NP	1400	300	250	1950
Salaries and wages	Sal	6	2	33	15	50	7 653	NP	50	NP	3	53
Name and address list	Name	2	1	3	5	9	2 019	NP	5	NP	1	6
Financial history	Hist	12	2	150	15	167	8 800	NP	23	NP	4	27
Order entry/stocks	Order	4	4	44	18	66	7 500	15	630	250	56	941
Depot replenishment	Depot	4	5	22	10	37	5 000	8	232	12	4	256
Material requirements	Mat req	6	6	12	9	27	4 000	50	200	91	12	353
No. of cases		49	53	53	53	53	42	26	42	33	43	41
Mean		8	6	43	36	86	18 584	65	205	63	31	334
Standard deviation		6	7	46	72	76	21 873	70	303	72	95	419

NP = Not produced; NA = Not available

* Size is defined as the sum of flows, files and processes.

TABLE 6B: Analysis of Results Produced and Resources Absorbed.

IDENTIFICATION	FILES	FLOWS	PROC.	SOURCE	DOC	COST					
						USER MANP	DP MANP	COMP POWER	SW	OTHER	TOTAL
ORDER	4	44	18	7 500	941	1000	17000	3600	-	1000	22600
SAL	2	33	15	7 653	53	2000	10000	3000	-	-	15000
LIFE	20	80	80	73 000	1430	13500	51075	13500	-	-	78075
GAINS	6	24	21	13 513	280	1600	19038	7350	-	-	27988
COST	6	227	50	NP	629	6000	64500	10000	-	-	80500
FIXED	3	20	18	8 665	58	3500	12600	8400	-	-	24500
S.BOOK	4	41	13	12 593	376	-	17650	1600	-	1200	20450
PROP	16	187	137	55 198	681	41580	NA	NA	NA	NA	147600
BROKER	4	19	15	5 730	101	400	3000	810	-	-	4210
NEWBUS	6	50	21	NP	905	12000	25950	10247*	-	-	48197
SALES	5	48	17	8 900	195	800	15200	4500	-	-	20500
TRIM	4	3	3	2 190	59	1000	3500	750	-	-	5250
WIP	3	17	34	9 263	117	-	12400	5040	-	-	17440
STOCK	9	15	17	6 018	159	880	6105	270	-	-	7255
MATUTIL	10	44	99	54 000	122	2000	43465	15956	-	-	61321
RECON	7	15	32	12 842	102	6000	NA	NA	NA	NA	34495
GRLIFE	5	32	40	36 000	379	3000	29800	8800	-	-	41600
ADVERT	1	31	35	31 909	258	2500	15200	9120	-	-	26820
JOBCOST	14	57	60	9 500	NA	3000	14800	7500	-	-	25300
NATURE	2	16	24	7 000	235	1370	8951	2787*	-	-	13108
Mean	7	50	37	20 082	373	5106	29563	6412	-	-	36110
St. dev.	5	55	33	20 893	378	9105	17479	4319	-	-	34164
Coeff. of var.	0,7	1,1	0,9	1,0	1,0	1,8	0,8	0,7	-	-	0,9

NA = Not available; NP = Not produced; * = Calculated

TABLE 6C

AVAILABILITY OF COST FIGURES IN THE MANPOWER AND COMPUTER POWER CATEGORIES

		No. of cases
A.	Manpower Costs	
1.	All man-month figures estimated	4
2.	Actual man-months, costs estimated	2
3.	dp Department actual costs, users estimated	7
4.	All actual costs	6
5.	Total actual costs, but manpower not specified	1
		20
B.	Computer Power	
1.	No estimate endeavoured	2
2.	Machine utilisation estiamted	11
3.	Actual machine utilisation, costs estimated	1
4.	Actual machine costs	5
5.	Total actual costs, but computer power not specified	1
		20

TABLE 6 D

COMPARISON OF MEASUREMENTS OF SUBJECTIVELY RANKED SYSTEMS

Smaller System			Larger System		
Size	Source	Doc.	Size	Source	Doc.
17	1 800	160	42	7 000	235
42	7 000	235	76	18 000	411
53	2 117	84	131	9 500	-
11	550	-	67	31 909	258
67	31 909	258	73	12 900*	-
24	8 300	80	77	36 000	379
77	36 000	379	181	57 500	810
17	-	84	54	12 842	102
54	12 842	102	192	-	810
116	40 000	99	153	54 000	122
153	54 000	122	238	75 000	175
24	3 142	-	41	6 018	159
41	6 018	159	119	-	-
27	-	-	54	9 263	117
54	9 263	117	99	-	-
10	2 190	59	150	-	-
7	1 000	-	70	8 900	195
70	8 900	195	103	-	-
28	-	-	77	-	905
77	-	905	109	-	-
12	2 000	27	38	5 730	101
38	5 730	101	65	8 200	127
65	17 184	55	340	55 198	681
208	40 648	199	340	55 198	681
20	2 602	23	41	8 665	58
41	8 665	58	64	8 848	65
24	2 886	100	51	13 513	280
51	13 513	280	53	14 503	720
20	5 700	122	180	73 000	1430
180	73 000	1430	196	80 000	1950
9	2 019	6	50	7 653	53
50	7 653	53	167	8 800	27*
37	5 000	256	66	7 500	941
27	4 000	353	66	7 500	941

* Indicates smaller value where larger is expected

TABLE 6E

OBSERVED EFFICIENCIES IN 20 SYSTEMS

(Using Model I)

IDENT.	EFFICIENCY
ORDER	29,9
SAL	33,3
LIFE	23,1
GAINS	18,2
COST	35,2
FIXED	16,7
S.BOOK	28,4
PROP	23,0
BROKER	90,2
NEWBUS	16,2
SALES	34,1
TRIM	19,0
WIP	31,0
STOCK	58,7
MATUTIL	25,0
RECON	15,7
GRLIFE	18,5
ADVERT	25,0
JOBCOST	51,8
NATURE	32,6
Mean	31,2
St. dev.	17,4
Coeff. of var.	0,6

TABLE 6F

INCIDENCE OF FACTORS INFLUENCING THE EFFICIENCY OF SYSTEM DEVELOPMENT

Tools

1. Was an important hardware or software tool new to the development team?

Yes	No
13	7

2. What make of computer was used for system development?

IBM	ICL	BURR.	NCR	OTHER
6	6	3	2	3

3. What was the availability of the required computer capacity?

On Line	Better than once a day	Less than once a day
6	12	2

4. Was use made of special utilities or systems software?

Yes	No
8	12

5. Which programming language was used?

COBOL	RPG	OTHER	NONE
17	1	2	

6. Was use made of system generation software or non-procedural languages

Yes	No
3	17

7. Any other tools?

Yes	No
1	19

Techniques

8. Was use made of specific methods for requirement specification and system design?

Yes	No
5	15

9. Was structural programming used?

No	Some Form	Strictly
3	11	6

10. Were standards covering most of the system development process enforced?

No	Loosely	Strictly
2	10	8

Table 6F (continued)

11. Any other techniques?

Yes	No
10	10

Personnel Related Factors

12. Familiarity of the development team with the subject matter of the system

1	2	3	4	5
0	3	8	7	2

13. Average number of years of experience

< 4	5-9	> 10
8	9	3

14. Intellectual ability of the development team

1	2	3	4	5
0	0	5	13	2

15. Level of motivation of the development team

1	2	3	4	5
0	2	0	10	8

16. Number of alterations in the team

0	1	2
14	4	2

17. Any other personnel related factors?

Yes	No
6	14

Organisation, Communication

18. Degree of user participation

1	2	3	4	5
1	3	2	6	8

19. Was chief programmer team used?

No	Some Form	Strictly
13	7	0

20. Was progress formally reported?

No	Verbal	Written
1	10	9

21. Top management support

1	2	3	4	5
2	1	3	6	8

22. Any other organisational factor?

Yes	No
5	15

Extraneous Factors

23. Development work done for third parties?

Yes	No
5	15

Table 6F (continued)

24. dp Practices influenced by a parent organisation?

Yes	No
3	17

25. How many years has the organisation performed DPSD?

6-10	11-15	> 16
3	6	11

26. How many people are full-time involved in dp?

< 10	11-50	51-100	> 100
3	7	5	5

27. How many people are involved in DPSD?

< 6	7-25	26-50	> 50
8	7	2	3

28. Any other factor?

Yes	No
2	18

29. Sector of industry

Man.	Comm.	Fin.	Govt.	Other
7	4	5	2	2

30. Location

CT	JHB
18	2

CHAPTER 7

CONCLUSION

CONCLUSION

7.1 REVIEW OF RESULTS

In Chapter 1, five objectives were stated for this study. The main conclusions reached can now be related back to these objectives as follows:

1. To define the basic concepts necessary for the construction of a measure of efficiency of software development in a data processing environment.

   - Principles of information and systems theory were used to identify flows, files and process as essential elements of data processing software.

   - The resources used in the software development process were enumerated and the total cost of these resources was identified as an important parameter of the measure of efficiency.

2. To consider specific formulations of a measure of efficiency of software development and select a preferred model.

   - Lines of source code divided by man-hours was rejected as a suitable measure of the efficiency of the total software development process.

   - Principles for the definition of a suitable measure of efficiency were defined, and based on these principles various models of efficiency consisting of functions of flows, files and processes divided

by the total project cost were suggested.

- A model consisting of the sum of flows, files and processes divided by the total project cost was defined as the preferred model of efficiency.

3. To establish criteria for the evaluation of the proposed measure of efficiency.

- Validity and reliability were identified as important criteria of quantitative measures in general, and as particularly relevant to the proposed measure of efficiency.

- It was found that the validity and reliability of measure of efficiency proposed in the literature were generally not tested.

- It was shown that the reliability and validity of the proposed measure could be assessed through various experiments.

4. To test the proposed measure of efficiency experimentally, using the previously defined criteria.

- A special experiment was conducted to test the reliability of the measure which was found to be adequate.

- Four different tests corroborated the validity of the proposed measure.

5. To describe and compare the efficiency and other characteristics of a number of typical software development projects that have recently been undertaken in a data processing environment by South African organisations.

- The main characteristics of 53 systems were identified and compared. The systems were found to be diverse in purpose and magnitude, but development times were within a limited range. Documentation was found to be poorly correlated with system size.

- Of 20 target systems, full details about magnitude and development costs were collected. Differences of up to a factor of six were encountered in the efficiency of the development of the systems.

- Routine recording of development cost was found to be lacking in many cases, but reasonable ad hoc estimates could in most cases be made.

Apart from the above results which can be directly related back to the explicit objectives of the study, several additional results were achieved, namely:

- It was established that there exists a substantial interest amongst data processing professionals in techniques for measurement and improvement of the efficiency of software development. However, the recording of relevant data was, in the majority of cases, found to be problematic.

- A categorisation was made of the factors that are likely to influence the efficiency of software development. Five categories were proposed, namely tools, techniques, organisation, personnel related factors, extraneous factors, and a total of 29 factors was distinguished. The survey was used to record the incidence of these factors. This may be useful for future research.

7.2 PRACTICAL APPLICABILITY OF THE RESULTS

A possible criticism of the practicality of the measure of efficiency defined in this study is the introduction and definition of rather novel concepts, namely flows, files and processes, and the need to know the total cost of the project. As discussed in Chapter 2, the conceptual framework necessary to define total costs certainly exists. In practice, the relevant figures may not always be recorded as was found in the survey. This detracts from the ability to make an incidental comparison of efficiency achieved in the past. Given the interest in the measurement of efficiency, the effort required to record relevant costs for current and future projects is, however, minimal and adequate record keeping is certainly feasible. The introduction of concepts like flows, files and processes was found to be absolutely necessary, firstly on theoretical grounds, but also because more traditional measures like lines of source code or pages of documentation were clearly inadequate.

Before the proposed measure can be used more widely in practice, it will be necessary to gain acceptance of the new concepts from the managers involved. In the process, the definitions of these concepts may be refined and altered to suit specific situations more precisely. By making such refinements and adaptations, the validity and reliability of the measure will probably be improved. Given this willingness to accept the principles and adapt the details, there is little doubt that the interested manager can make practical use of the proposed measure of efficiency in the area of data processing as defined in this study.

Another limitation of the practical applicability of the measure is the narrow definition of data processing systems in Chapter 2. This was done to be able to arrive at concise definitions of the structural elements without getting lost

in exceptions and alternatives. The measure is therefore strictly applicable only to data processing systems as defined. In other areas of software development a similar need exists for quantification and measurement. To meet these needs, it will be necessary either to redefine the concepts of flows, files and processes or select entirely different attributes. In either case the validity and reliability of the resulting measures will have to be tested. It is considered to be a distinct contribution of this study to have shown how relevant attributes can be selected and how the validity and reliability of a proposed measure can be tested against empirical data.

7.3 A PERSPECTIVE ON EFFICIENCY IN SOFTWARE DEVELOPMENT

Apart from achieving the specific objectives set in Chapter 1, this study has also resulted in an increased understanding of the phenomenon of efficiency in the context of software development. This may be reflected in a perspective on software development in which certain aspects that are particularly related to efficiency are highlighted. This perspective involves both causes and consequences of efficiency. It has, in the following paragraphs, been cast in the form of a number of popular misconceptions about software development which are refuted, using insights gained by this study.

- Software development and the problems related to it can only be understood by computer experts.

  This notion confuses the understanding of the technology supporting data processing and software development with the activity itself. Unfortunately this preoccupation with the supporting technology has often prevented careful attention to the actual activity. The software development activity is, however, not at all difficult to understand,

certainly not to a person who is a manager of other production processes. As argued in this study, software development is a production process which produces certain results and requires certain resources. The resources can be enumerated and the most important dimensions of the results can be defined. Control of the software development activity requires control, and therefore understanding, of these dimensions. The most important physical dimensions (flows, files and processes) have been identified. Beyond that, a number of other dimensions of the results of software development, notably effectiveness, flexibility and complementary resources have been discussed in Chapter 3. The challenge to anyone involved in software development is to understand and control these aspects.

- Measurement of the efficiency of software development requires complex calculations involving a large number of factors.

  It has been suggested that simplicity, rather than complexity, is the hallmark of a useful formula. Furthermore, it was shown that a simple formula expresses the relationship between resources required and results produced just as well as a more complicated formula. In the literature even more complex formulations of efficiency of software development have been proposed. Such formulations are the result of two tendencies. Firstly, there is the tendency to attempt to cover all possible ways and means of software development by one general formulation. This leads necessarily to a complexity which obscures rather than clarifies the underlying structure.

  Secondly, the tendency exists to include in the analysis factors that influence efficiency. Examples are the complexity of the subject matter or the programming

language used. In this study efficiency has been defined and measured without regard to any of the factors that may influence it. Surely such factors are important for understanding the substantial differences in efficiency that were observed. That analysis should, however, follow the determination of efficiency as a second stage. The foundations for this second stage have been laid in Chapter 5 with the identification of the major categories of the factors influencing efficiency.

- Recording of the effort spent on software development serves only historical purposes.

  This statement is not often made explicitly, but it appears that many organisations adhere to it implicitly. It was found in this study that comprehensive recording of all cost elements of each software development project is the exception rather than the rule. Yet, it is clear that without reasonably reliable cost figures it will not be possible to calculate and control the efficiency of software development properly. It is not clear whether this indicates a lack of real interest in the promotion of efficiency or a despair about the ability either to calculate or control it. It is hoped that this study has contributed to the ability to calculate efficiency and may thereby provide an incentive to collect the neccessary data to control it.

- Productivity of software development can be effectively controlled by measuring lines of code per programmer month.

  Firstly, this notion ignores the software development life cycle and the important contributions made by staff other than programmers. Programming is generally accepted to be only a small part of the total software development effort and it is not at all certain that the

efforts of others (e.g. analysts) stand in a fixed relation to those of programmers. Furthermore, other cost elements such as purchased hardware and software are not insignificant.

Secondly, the preoccupation with lines of code as the only result of the software development process is not realistic. The code must eventually incorporate definitions of the structural elements of the data processing system, but it may contain many duplications, while the definition of a line of code is not at all trivial. Lines of code per programmer month may have been a suitable measure in the early days of data processing when software consisted of one program produced by one programmer but it is an increasingly deficient measure of the efficiency of modern software development.

- Estimating the effort required to complete a software development project must remain largely guesswork.

This statement may be true when nothing is known of the system except maybe its name or a rough indication of its function. When the design has progressed far enough to identify the structural elements of the system, it is possible to make a reasonable estimate of further development effort. This can be done by making use of the relationship between development costs and the structural elements of data processing systems which were identified in the survey. To do this, it is necessary to postulate a certain level of efficiency. Average efficiencies attained in the past may be used for this purpose, but the estimates will obviously be better if the efficiency to be achieved in this project can be defined with some precision. This implies paying attention to the factors that are likely to influence efficiency. Rather than just a forecase of a future date

(completion of the project), the estimating task thus becomes the definition of a specific occurrence (e.g. the level of relevant know-how of the staff). Estimation of software development effort therefore never becomes automatic. Application of the insights gained in this study can, however, alter it from a gamble to a calculated risk.

- Intuitive validation is enough to propose or accept any measure.

  This is not a statement which is often made explicitly, but implicitly it appears to be readily accepted in the data processing industry. It has been emphasised in this study that the uncritical acceptance (or rejection) of a measure is neither correct nor necessary. The validity, reliability and sensitivity of a measure should be formally tested by procedures such as those discussed in Chapter 4 before any faith is put in it.

- Every organisation and every software project is different and comparison of efficiency of software development is therefore pointless.

  It may be true that organisations and projects are different in detail, but it is certainly possible to detect common structures. A common structure expresses itself firstly in the common structural elements. Secondly, the software development life cycle and its participants provide another common basis. By linking together these common aspects, the foundation is laid for meaningful comparison of quantifiable differences. These quantifiable differences as well as an appreciation of the factors causing the differences, will lead to a more complete understanding of software development. The quantification of similarities and differences and the identification of influencing factors is therefore not

pointless, but rather the basis for the evaluation and thereby hopefully the improvement of practice.

7.4 SUGGESTIONS FOR RELATED RESEARCH

Opportunities for further research are indicated by some of the limitations and problems identified in this study. A definite problem area was identified in Chapter 3, when the characteristics of software were discussed. It was found that some important dimensions describing the quality of software were ill-defined and therefore particularly hard to measure. Concepts such as effectiveness, flexibility and maintainability are of obvious importance for a more complete understanding of software. However, Lord Kelvin's dictum linking quantification and understanding of a phenomenon is particularly apt in this area. The study and proposal of measures of effectiveness, flexibility and maintainability of software appears therefore to be an important area for future research. Such research will be difficult, due to the lack of stability of software and the rapid changes in requirements, tools and techniques. It will, however, ultimately be very fruitful, as it will provide the definitions and foundations upon which improvements in practices must be based.

The limitations of this study stated in the Chapters 1 and 2 indicate that there are substantial areas of software development to which this study may be extended. The most obvious ones are data base oriented systems and other more technical types of software development. Efficiency of development is of no less concern in these areas and comparative measures and studies are not known to exist. It may be particularly relevant to extend this study to the area of data base systems, where three design philosophies, notably hierarchical, network and relational systems are in competition. The choice of design philosophy must to a

large extent depend on the efficiency of application system development. Research into this aspect should be of substantial practical value.

Refinements of the results of this study, aimed at developing a measure of efficiency for a specific organisation would also be of interest. As indicated in paragraph 7.2 such refinements could lead to a more elaborate, but particularly a more accurate model of efficiency for a specific organisation. This may result in greater accuracy in the forecasting of software development effort for that organisation.

A further refinement of the results of this study is indicated in the area of software development cost elements and their relative importance. This study has provided some limited survey data, but a survey specially designed for this purpose could yield a lot more insight. Such insight is important for the management and control of software development.

It was indicated in Chapter 5 that, apart from the measurement of efficiency, insight into the factors influencing the efficiency of software development would be of great significance. The survey provided an indication of the incidence of some of these factors. It is clear that the identification of factors that coincide with (and can possibly be shown to cause) high or low efficiency, is of great potential significance. By manipulating these factors, management may ultimately be able to achieve the improvements in the efficiency of software development that are needed. Studies relating efficiency to specific factors will therefore be of particular interest.

Finally, certain software development activities were described in Chapter 2 as ill-defined and ill-understood. The most notable ones are systems analysis and system

design. It is generally recognised that these activities have a major influence on the quality of software and on the efficiency of its development. Recently, various ways of structuring (and thereby hopefully improving) the analysis and design activities have been suggested. It is interesting to note that in some of these techniques flows, files and processes or at least very closely related concepts play an important part. The effect of these techniques on the quality of software and on the efficiency of its development appears to be a worthwhile area for research.

7.5 SUMMARY

In this chapter the results of this study have been assembled and reviewed. The conclusions reached in previous chapters were summarised and related to the objectives set out in Chapter 1.

The results of the survey, combined with insights from the literature made it possible to examine various facets of software development from a common perspective.

Finally, a wide range of indications for related research made it clear that much effort is still required to lift software development from a craft to a profession.

BIBLIOGRAPHY *

AAKER, D.A. : Multivariate analysis in marketing, Wadsworth Publishing Company Inc, Belmont Calif., 1971

ACKOFF, R.L. : Management Mis - information systems, Management Science, December 1967

ACKOFF, R.L. : A concept of corporate planning, John Wiley and Sons, New York, 1970

ACKOFF, R.L. : Towards a system of systems concepts, Management Science, July 1971

ADRIAANSE, P. : Vormen en komponenten van gedistribueerde systemen, Informatie, April 1977

ALTER, S.L. : How effective managers use information systems, Harvard Business Review, Nov/Dec., 1976

ALTER, S.L. and GINZBERG, M. : Managing uncertainty in MIS implementation, Sloan Management Review, Fall 1978

ALTER, S.L. : Decision support systems : current practices and continuing challenges, Addison Wesley Publishing Company, Reading Mass, 1980

ANDERSON, T.W. : An introduction to multivariate statistical analysis, John Wiley and Sons, New York, 1958

ANSOFF, H.I. : Corporate strategy, Penguin Books Ltd, Hammondsworth, England, 1968

ANTONY, R.N. : Planning and control systems : a framework for analysis, Graduate School of Business Administration, Harvard University, Boston, 1965

ANTONY, R.N., DEARDEN, J., VANCIL, R.F. : Management control systems, Richard D. Irwin, Homewood Ill., 1972

* Note: These publications were consulted for this thesis but are not all referenced explicitly.

ARGYRIS, C. : Management information systems : the challenge to rationality and emotionality, Management Science, February 1971

ARON, J.D. : The program development process Part I, Addison Wesley Publishing Co, Reading Mass, 1974

AUERBACH, I. : The information revolution : will it improve the quality of life in : Data processing manual, Auerbach Publishers, Philadelphia, 1975

AUERBACH Information Management Series, Data processing manual, Auerbach Publishers, Philadelphia, 1975

AVOTS, I. : Making project management work, Datamation, January 1973

BAILEY, G. : Its quicker and cheaper to program on-line, Data Systems, October 1977

BAKER, F.T. : Chief programmer team management of production programming, IBM Systems Journal, No.1 1972

BAKER, F.T. and MILLS, H.D. : Chief programmer teams, Datamation, December 1973

BAKER, F.T. : Structured programming in a production programming environment, IEEE Transaction on software engineering, June 1975

BANBURY, J. : The concept information system effectiveness, in : Frielink, A.B. : The economics of informatics, North Holland Publishing Co, Amsterdam, 1975

BEER, S. : Cybernetics and management, The English University Press, London, 1959

BEER, S. : The brain of the firm, The Penguin Press, London, 1972

BENJAMIN, R. : Control of the information systems development cycle, John Wiley and Sons, New York, 1971

BLEE, M. : Software - cutting the costs, Data Systems : September 1974

BLUMENTHAL, S.C. : Management information systems : a framework for development, Prentice Hall Inc, Englewood Cliffs N.J., 1969

BOEHM, B. : Software and its impact : a quantitative assessment, Datamation, May 1973

BOEHM, B. : Software engineering, IEEE Transaction on computers, Vol. C - 25 December 1976

BOEHM, B., BROWN, J.R., KASPAR, H., LIPOW, M., MACLEOD, G.J., MERRITT, M.J. : Characteristics of software quality, TRW series on software technology, North Holland Publish Company, Amsterdam 1978

BOHM, C. and JACOPINI, G. : Flow diagrams, Turing machines and languages with only two formation rules, Communications of the ACM, May 1966

BOSCH, P.G. : Een raamwerk voor de analyse van informatie behoeften, Informatie, Sept, 1972

BOTHA, F.G. : An analysis of the role of software languages in the development of the computer information systems of the major financial organisations in South Africa, Unpublished MBA thesis, The Graduate School of Business, Cape Town, 1977

BOULDEN, J.B. and BUFFA, E.S. : Corporate models: on-line, real-time systems, Harvard Business Review, July-August 1970

BOULDING, K. : General systems theory - The skeleton of science, Management Science, April 1956

BRANDON, D.H. : Management standards for data processing, D. van Nostrand Co Inc, Princeton, New Jersey, 1963

BREEVOORD, C. : Distributie en informatie, H.E. Stenfert Kroese NV Leiden, 1969

BRODIE, M. and BENNET, R. : Effective management and the auditing of performance, Journal of General Management, Spring 1979

BROOKS, F.P. : The mythical man-month, Addison-Wesley Publishing Co, Reading, Mass, 1975

BROWN, H.L. : User control of data processing, Management Accounting, March 1976

BURCH, J.G. and STRATER, F.R. : Information systems : Theory and practice, Hamilton Publishing Co, Santa Barbara, Calif., 1974

BYLINSKI, G. : Help wanted : 50000 Programmers, Fortune, March 1967

BYLINSKI, G. : EDP managers put on business suits, Fortune, Nov 6, 1978

CAMMAN, C., NADLER, D.A. : Fit control systems to your managerial style, Harvard Business Review, Jan/Feb, 1976

CANNING, R.G. : The cautious path to data base, EDP Analyser, June 1973

CANNING, R.G. : Are we doing the right things, EDP Analyser, May 1975

CANNING, R.G. : Are we doing things right, EDP Analyser, June 1975

CANNING, R.G. : The benefits of standard practices, EDP Analyser, August 1975

CANNING, R.G. : Progress toward easier programming, EDP Analyser, Sept, 1975

CANNING, R.G. : APL and decision support systems, EDP Analyser, May 1976

CANNING, R.G. : Toward better management of data, EDP Analyser, Dec, 1976

CANNING, R.G. and SISSONS, R.L. : The management of data processing, John Wiley and Sons, New York, 1967

CHAMBERLIN, D.D. : Relational data base management systems, Computing Surveys, March 1976

CHAMPINE, G.A. : Six approaches to distributed data bases, Datamation, May 1977

CHANDLER, A.D. : Strategy and Structure, The M.I.T. Press, Cambridge Mass, 1962

CHAPIN, N. : A measure of software complexity, Proceedings of the National Computer Conference, 1979

CHRYSLER, E. : Programmer performance standards, Journal of Systems Management, Feb, 1978

CHURCHILL, G.A. : A paradigm for developing better measures for marketing constructs, Journal of Marketing Research, February 1979

CHURCHMAN, C.W. : The systems approach, Dell Publishing Co, New York, 1968

CLOVER, V. and BOLSEY, H. : Business research methods, Grid Inc, Columbus, Ohio, 1974

CODASYL, Data base task group, Report, ACM, New York, 1971

COOKE, L.H. : On estimating, Datamation, June 1979

CORSON, M. : A comparative analysis of computer suppliers in South Africa, Unpublished MBA thesis, The Graduate School of Business, Cape Town, 1976

COUGER, J.D. : Evolution of business systems analysis techniques, Computing Surveys, September, 1973

COUGER, J.D. and KNAPP, R.W. : System analysis techniques, John Wiley and Sons, New York, 1974

COUGER, J.D. and ZAWACKI, R.A. : What motivates DP professionals, Datamation, September, 1978

COX, G. : Solving the problem of project management, Data Systems, July/August 1976

CROSS, H. : Computer and management, Graduate School of Business Administration, Boston, 1967

CROSSMAN, T.D. : Taking the measure of programmer productivity, Datamation, May 1979

CROSSMAN, T.D. : Programmer productivity measurement, Systems/Stelsels, March, 1978

CURTIS, R.M. : Data independence in data base management systems, Datamation, April, 1975

DAHL. O.J., DYKSTRA, E.W., and HOARE, C.A.R. : Structured programming, Academic Press, 1972

DALY, E.B. : Organising for successful software development, Datamation, December 1979

DATAPRO : The EDP buyers bible, Datapro Research Corporation, Delran, N.J., 1979

DATAPRO : A buyers guide to data base management systems, Datapro Research Corporation, Delran N.J., 1979

DATAPRO : A buyers guide to data management systems, Datapro Research Corporation, Delran, N.J., 1975

DATE, C.J. : An introduction to database systems, Addison Wesley Publishing Co, Reading, Mass, 1975

DAVIS, G.B. : Computer data processing, McGraw Hill Book Company, New York, 1973

DAVIS, G.B. : Management information systems: Conceptual foundations, structure and development, McGraw Hill, New York, 1976

DAY, S.A. : Consequences of data base for the management of large organisations, Informatie, May 1976

DEARDEN, J. : MIS is a mirage, Harvard Business Review, Jan - Feb, 1972

DEARDEN, J. and NOLAN, R.L. : How to control the computer resource, Harvard Business Review, November - December 1973

DE KERF, J. : De geschiedenis van de automatische digitale rekenmachine, Informatie, October 1977

DE MAAGD, G.R. : Matrix management, Datamation, September 1971

DEPARTMENT of Statistics : Computer survey 1977, Government Printer, Pretoria, 1978

DERKINDEREN, F.G.J. and EPPINK, J. : Better roughly right than exactly wrong, Journal of General Management, Autumn 1979

DEURNINCK, G. : Productivity measurement Vol.I: Concepts, Organisation for Economic Cooperation and Development, Paris, 1955

DICKSON, G.W. and SIMMONS, J.K. : The behavioral side of MIS, Business Horizons, August 1970

DICKSON, G.W. and POWERS, R.F.: MIS project management : myths opinions and reality, in : Mc Farlan, F.W., Nolan, R.L., and Norton, D.P. : Information Systems Administration, Holt, Rinehart, Winston Inc, New York, 1973

DOLOTTA, T.A., BERNSTEIN, M.I., DICKSON, R.S., FRANCE, N.A., ROSENBLAT, B.A., SMITH, D.M., STEEL, T.B. : Data processing in 1980-1985, John Wiley and Sons, New York, 1976

DONELSON, W.S., Project planning and control, Datamation, June 1976

DONOVAN, J.J. : Systems programming, McGraw Hill Book Co, New York, 1972

DUFFY, N.M. : The design of flexible computer-based information systems, Unpublished doctoral thesis, The University of South Africa, Pretoria, 1976

DUFFY, N.M. : Towards more effective information systems, Systems/Stelsels, August, 1976

DRUCKER, P.F. : The practice of management, William Heinemann, London, 1961

DYKSTRA, E.W. : Go to statement considered harmful, Communications of the ACM, March, 1968

EBEL, E. : The evolution of the role of the data processing manager in South Africa, Unpublished MBA thesis, The Graduate School of Business, Cape Town, 1979

EDITORIAL : A new approach to training, Business South Africa, June 1978

ENGER, N.L. : Management standards for developing information systems, Amacom, New York, 1976

ERDOS, P.L. and MORGAN, A.J. : Professional mail surveys, McGraw Hill Book Company, New York, 1970

FARADAY, J.E. : The management of productivity, Management Publications, British Institute of Management, Tonbridge 1971

EXLEY, M., HARDING, N. : Computers for people - designing human systems, Data Systems, February 1977

FELDBERG, M.F. : Organisational behaviour, Juta & Co Ltd, Cape Town, 1975

FERGUS, R.M. : Decision tables - What, why and how, in:Couger, J.D. and Knapp, R.W., Systems Analysis Techniques, John Wiley and Sons, New York, 1974

FERREIRA, A.A. : The use of computerised models in financial planning, Unpublished MBA thesis, The Graduate School of Business, Cape Town, 1977

FERREIRA, J. and NILLES, J.M. : Five-year planning for data communications, Datamation, October 1976

FINE, L.H. : Management standards in systems and programming, Systems/ Stelsels, August, 1974

FINLAYSON, R.L. (Ed.) : Computer users handbook 1977/1978, Systems Publishers Johannesburg, 1978

FITZ-ENZ, J. : Who is the DP professional, Datamation, September, 1978

FITZROY, P.T. : Analytical methods for marketing management, McGraw Hill, London, 1976

FORRESTER, J.W. : Industrial Dynamics, The MIT Press, Cambridge Mass, 1961

FRANK, R.E., GREEN, P.E. : Numerical Taxonomy in marketing analysis : A review article, in : Aacker, D.A., Multivariate analysis in marketing, Wadsworth Publishing Company, Belmont, Calif., 1971

FREEMAN, P. : Software reliability and design : a survey, Proceedings of the 13th design automation conference of the IEEE, 1976

FRIED, L. : Estimating the cost of system implementation, in Couger, J.D. and Knapp, R.W. : System Analysis Techniques, John Wiley and Sons, New York, 1974

FRIELINK, A.B. : The economics of informatics, North Holland Publishing Co, Amsterdam, 1975

FRYER, H.C. : Concepts and methods of experimental statistics, Allyn and Bacon Inc, Boston, 1966

GANE, C. and SARSON, T. : Structured systems analysis, Prentice Hall Inc, Englewood Cliffs N.J., 1979

GANZHORN, K. and WALTER, W. : Die geschichtliche entwicklung der datenverarbeitung, IBM Deutschland, Augsburg, 1975

GEENEN, J. : Inleiding tot methodisch ontwerpen, Informatie, Maart, 1977

GELPER, R. : Lange - termynplanning voor computer automatisering, Informatie, April, 1978

GIBSON, C.F. and NOLAN, R.L. : Managing the four stages of EDP growth, Harvard Business Review, Jan - Feb, 1974

GILB, T. : Generalised data base management systems, Informatie, March, 1974

GILBERT, J.C. : Can today's MIS manager make the transition, Datamation, March, 1978

GILDERSLEEVE, T.R. and LADEN, H.N. : System design for computer applications, John Wiley and Sons, New York, 1967

GILDERSLEEVE, T.R. : Successful data processing systems analysis, Prentice Hall Inc, Englewood Cliffs, N.J., 1978

GLASER, B.G. and STRAUSS, A.L. : The discovery of grounded theory, Aldine Publishing Co, Chicago, 1967

GORRY, A. and SCOTT MORTON, M.S. : A framework for management information systems, Sloan Management Review, Fall, 1971

GOSDEN, J.A. : Some cautions in large-scale system design and implementation, Information and Management, No.2 1979

GREEN, P. and TULL, D. : Research for marketing decisions, Prentice Hall Inc, Englewood Cliffs, N.J., 1970

GEORGE,D. et.al.: Predicting cost of change from design structure metrics, Software Engineering Notes, Jan, 1982.

HEALTH SCIENCES COMPUTING FACILITY : Biomedical computer programs, University of California Press, Berkeley, Calif., 1977

HELD, G., WOLTERS, M. : From pocket solutions to coherent software technology, Dataweek, November, 1979

HILL, M.O. : Correspondence Analysis : A neglected multivariate method, Journal of Applied Statistics, No.3, 1974

HIRSCH, R.E. : The value of information, The Journal of Accountancy, June, 1968

HOETINK, B.J. : Besturingseffektiviteit van informatie systemen, Informatie, September, 1976

HOLTON, J.B. : Are the new programming techniques being used, Datamation, July, 1977

IBM Corporation : An introduction to IBM data processing, IBM Technical Publications Dept, White Plains, 1967

IBM Corporation : Improved technologies for application development, Productivity Improvement Department IBM Corporation, Betseda, Md., 1973

IBM Corporation : Business systems planning: Information planning guide, IBM Technical Publications Department, White Plains, 1975

INMON, B. : An example of structured design, Datamation, March 1976

JACKSON, M.A. : Principles of program design, Academic Press, London, 1975

JACOBY, J. : Consumer research : a state of the art review, Journal of Marketing, April 1978

JOHNSON, J.R. : A working measure of productivity, Datamation, February, 1977

JOHNSON, J.R. : The changing d.p. organisation, Datamation, January, 1975

JOHNSON, R., KAST, F.E. and ROSENSWEIG, J.E. : The theory and management of systems, John Wiley and Sons, New York, 1967

HERZ, D.B. : New power for management, McGraw Hill Book Co, New York, 1969

JONES, J.C. : Design methods, John Wiley and Sons, New York, 1976

JONES, T.C. : Measuring programming quality and productivity, IBM Systems Journal, No.1, 1978

KALMAN, R.E. : Towards the quantitative measurement of development of informatics, in : Frielink, A.B. : The economics of informatics, North Holland Publishing Company, Amsterdam, 1975

KANTER, J. : Management oriented management information systems, Prentice Hall Inc, Englewood Cliffs, N.J., 1977

KEEN, P.G.W. and SCOTT MORTON, M.S. : Decision support systems, Addison Wesley Publishing Co, Reading, Mass, 1978

KENDRICH, J.W. : Understanding productivity, The John Hopkins University Press, Naltimore, 1977

KING, W.R., EPSTEIN, B.J. : Assessing the value of information, Management Datamatics, No.4, 1976

KINGSTON, P.L. : Concepts of financial models, IBM Systems Journal, No.2, 1973

KIRKLEY, J.L. : Programmer productivity, Datamation, May, 1977

Kleynen, J.P.C. : Computers and profits: Quantifying financial benefits of information, Addison Wesley, Reading Mass, 1980

KOTLER, P. : Corporate models : better marketing plans, Harvard Business Review, July - August 1970

KRAFT, P., WEINBERG, G.M. : The professionalisation of programming, Datamation, October, 1975

KUEHNE, R.S., LINDBERG, H.W., BARON, W.F. : Manual of computer documentation standards, Prentice Hall, Englewood Cliffs, N.J., 1972

LANGEFORS, B. : Theoretical analysis of information systems, Studentliteratur, Lund, 1969

LAWRENCE, P.R., LORSCH, J.W. : Organisation and environment, Richard D. Irwin Inc, Homewood Ill., 1969

LEARNED, E.P. and SPROAT, A.T. : Organisation theory and policy, Richard D. Irwin, Homewood Ill., 1966

LIPOVICH, G.J. : DP manager and performance measurement, Journal of Systems Management, March 1977

LUCAS, H.C. : The evolution of an information system, Sloan Management Review, Winter, 1978

LUCAS, H.C. : Toward creative systems design, Columbia Press, New York, 1974

LUCAS, H.C. : Impacting the organisation through systems development, Data processing manual, Auerbach Publishers, Philadelphia, 1975

LUCAS, H.C. : Why information systems fail, Columbia University Press, New York, 1975

LUCAS, H.C. : The analysis, design and implementation of information systems, McGraw Hill Book Co, New York, 1976

McCRACKEN, D.D. : The changing face of applications programming, Datamation, November, 1978

McFADDEN, F.R. and SUVER, J.D. : Costs and benefits of a data base management system, Harvard Business Review, Jan/Feb, 1978

McFARLAN, F.W. : Effective EDP project management, Unpublished working paper, Harvard Business School, Boston, 1973

McFARLAN, F.W., NORTON, D.P. and NOLAN, R.L. : Information systems administration, Holt, Rinehart, Winston, New York, 1973

McFARLAN, F.W. : Management audit of the EDP department, Harvard Business Review, May - June, 1973

McKINSEY : Unlocking the computer's profit potential, McKinsey and Co, New York, 1968

McLAREN, K.G. and BUESNEL, E.L. : Network analysis in project management, Cassil, London, 1969

McLAUGHLIN, R.S. : That old bugaboo, Turnover, Datamation, October, 1979

McLUHAN, M. : Understanding media: The extension of man, Routledge & Kegan Paul Ltd, London, 1964

MARTIN, J. : Design of real time computer systems, Prentice Hall Inc, Englewood Cliffs, N.J., 1964

MARTIN, J. : Principles of data base management, Prentice Hall Inc, Englewood Cliffs, N.J., 1976

METZGER, P.W. : Managing a programming project, Prentice Hall Inc, Englewood Cliffs, 1973

MEYER, J.H.F. : The application of educational technology in selected areas and disciplines in university training, Unpublished doctoral thesis, University of the Witwatersrand, Johannesburg, 1974

MIDDLETON, C.J. : How to set up a project organisation, Harvard Business Review, March/April 1967

MILLS, H. : Chief programmer teams, principles and procedures, IBM report FSC 71-5108, Gaitheysburg, Md., 1971

MILUTINAVICH, Y. : Management information systems, Data Systems, April/May 1978

MINZBERG, H. : Strategy making in three modes, California Management Review, Winter, 1973, p.44

MORGAN, J.H. and LIGHTMAN, M.S. : System for developing systems, Datamation, April 1976

MUMFORD, E. and WARD, T.B. : Computers : Planning for people, B.T. Batsford Ltd, London, 1968

MURDICK, R.G. : MIS development procedures, in : System Analysis Techniques, D.J. Couger and R.W. Knapp (Ed.), John Wiley and Sons, New York, 1974

MURDICK, R.G., ROSS, J.E. : Information systems for modern management, Englewood Cliffs, N.J., Prentice Hall Inc, 1971

NOLAN, R.L. : The plight of the EDP manager, Harvard Business Review, May/June, 1973

NOLAN, R.L. : Computer data bases : The future is now, Harvard Business Review, Sept/October, 1973

NOLAN, R.L. : Business needs a new breed of EDP manager, Harvard Business Review, March/April, 1976

NOLAN, R.L. : Controlling the cost of data services, Harvard Business Review, July/August, 1977

NOLAN, R.L. : Managing the crisis in data processing, Harvard Business Review, March/April, 1979

NAYLOR, T.H. and SCHAULAND, H. : Experience with corporate simulation models - a survey, Long Range Planning, April, 1976

OPTNER, S.L. : Systems Analysis, Penguin Books Ltd, Hammondsworth, England, 1973

ORPEN, S. : The chip revolution, Management, December, 1979

OSGOOD, C.E., SUCI, G.J., TANNENBAUM, P.H. : Measuring the meaning, University of Illinois Press, Urbana, 1957

PARNAS, D.L. : On the criteria to be used in decomposing systems into modules,Communications of the ACM, December, 1972

PARTEN, M. : Surveys, polls and samples : practical procedures, Harper and Row, New York, 1975

PATRICK, R.L. : The productivity gap, Datamation, December, 1979

PEEPLES, D.E. : Measure for productivity, Datamation, May, 1978

PERRY, W.E. : Trends in EDP auditing, Edpacs, December, 1976

PETERS, L.J. and TRIPP, L.L. : Is software design "wicked", Datamation, May, 1976

PETTIGREW, A. : Managing specialist groups, Data Systems, May, 1975

HALSTEAD,M.H.: Elements of Software science, Elsevier, New York, 1977.

GREENACRE, M.J. : Some objective methods of graphical display of a data matrix, Special report, University of South Africa, Pretoria, 1978

GRINDLEY, C.B.B. : The rationale of measuring efficiency, in:Frielink A.B. : The Economics of Informatics, North Holland Publishing Company, Amsterdam, 1975

GRINDLEY, K. and HUMBLE, J. : The effective computer, McGraw Hill Book Co, London, 1973

GRINYER, P.H. and WOOLER, J. : Corporate models today : a new tool for management, Institute of chartered accountants in England and Wales, London, 1975

GRUENBERGER, F. (Ed.) : Effective versus efficient computing, Prentice Hall Inc, Englewood Cliffs, N.J., 1973

HACKMAN, J.R., OLDHAM, G.R., JANSON, R., PURDY, K. : A new strategy for job enrichment, California Management Review, No.4 1975

HALL, C.S. and LINZEY, G. : Theories of personality, John Wiley and Sons, New York, 1970

HALL, W.K. : Strategic planning models : are top managers really finding them useful, Journal of Business Policy, No.2, 1973

HAMMINK, R.J. : Administrative automatisering in Nederland 1976 - 1981, Studiecentrum novi, Amsterdam, 1979

HAMMOND, S.J. : Do's and dont's of computer models for planning, Harvard Business Review, March-April, 1974

HANOLD, T. : An executive view of MIS, Datamation, November, 1972

HAROLD, S. : Programming for the future, Data Systems, May 1975

HARTMAN, A., MATTHES, H., PROEME, A. : Management Information Systems Handbook, McGraw Hill Book Co, New York, 1968

HAYES, R.H. and NOLAN, R.L. : What kind of corporate modelling functions best, Harvard Business Review, May - June, 1974

HEAD, R.V. : Management information systems : a critical appraisal, Datamation, May 1967

PINKNEY, A. : An audit approach to computers, The general educational trust of the institute of chartered accountants in England and Wales, London, 1970

PRINCE, T.R. : Information systems for management planning and control, Richard D. Irwin Inc, Homewood, Ill., 1970

PULLEN, E.W., SIMKO, R.G. : Our changing industry, Datamation, January, 1977

PUTNAM, L.H. and FITZSIMMONS, A. : Estimating software costs, Datamation, September, 1979

RAVER, N. and HUBBARD, G.U. : Automated logical data base design concepts and applications, Informatie, June, 1978

ROSS, R.G. : Data base systems, Amacom, New York, 1978

ROSS, J.E. : Management by information systems, Prentice Hall Inc, Englewood Cliffs, N.J., 1970

RUISCH, R. : Groei van de automatisering binnen een bedryf, Informatie, March, 1978

SACKMAN, H., ERIKSON, W.J., GRANT, E.E. : Exploratory experimental studies comparing on-line and off-line programming performance, Communication of the ACM, January, 1968

SAMMET, J.E. : Programming languages: history and fundamentals, Prentice Hall, Englewood Cliffs, N.J., 1969

SCHACH, S.R. : A unified theory for software production, Unpublished study, Department of Computer Science, UCT Cape Town, 1979

SCHUTTE, G. : Information Systems: A framework for development, Bedryfsleiding, No.2, 1975

SCHWARTZ, M.H. : MIS Planning, Datamation, September, 1970

SCHWARTZ, FINE, KANE and CO : System documentation and standards, Unpublished manual, Cape Town, 1974

SCHEEPMAKER, B. : Organisatorische konsekwenties van de invoering van gedistribueerde gegevensverwerking, Informatie, April, 1977

SCOTT, R.F. and SIMMONS, D.B. : Programmer productivity and the delphi technique, Datamation, May, 1974

SCOTT, W.R., BLAU, P.M. : Formal organisations, Chandler Publishing Company, San Francisco, 1962

SELLTIZ, C., JAHODA, M., DEUTSCH, M., COOK, S.W. : Research methods in social relations, Holt, Rinehart and Winston, New York, 1967

SHANNON, C.E. and WEAVER, W. : The mathematical theory of communications, The University of Illinois Press, Urbana, 1949

SHANNON, R.E. : Systems simulation: The art and science, Prentice Hall Inc, Englewood Cliffs, N.J., 1975

SHARPE, W.F. : The economics of computers, Columbia University Press, New York, 1969

SHAW, M.E., WRIGHT, J.M. : Scales for the measurement of attitudes, McGraw Hill Book Co, New York, 1967

SHONE, L. : Towards the eighties: the data processing department, its organisations and function, Data Systems, Nov/Dec, 1979

SIEGEL, S. : Nonparametric statistics for the behavioral sciences, McGraw Hill Book Co, New York, 1956

SILVER, G.S. and SILVER, J.B. : Data processing for business, Harcourt Brace Favanovitch Inc, New York, 1973

SIMON, H.A. : The new science of management decisions, Harper and Row, New York, 1960

SOKAL, R.R., SNEATH, P.H. : Principles of numerical taxonomy, W.H. Freeman and Company, San Francisco, 1963

SPIEGEL, M.R. : Theory and problems of statistics, McGraw Hill Book Co, New York, 1961

SPETT, M.C. : Standards for evaluating data processing management, Datamation, December, 1969

STAELIN, R. : Another look at AID, Journal of Advertising Research, October, 1971

STARREVELD, R.W. : Bestuurlyke informatie technologie, Samson Uitgevery, Alphen aan de Ryn, 1971

STEINER, G.A. : Top management planning, The MacMillan Company, London, 1969

STEVENS, S.S. : Measurement and man, Science, 21 February, 1958

STEVENS, W.P., MYERS, G.J., CONSTANTINE L.L. : Structured design, IBM Systems Journal, Vol.13, No.2, 1974

STEWART, T.J. : A descriptive approach to multiple criteria decision making, Technical Report, National Research Institute for Mathematical Sciences, Pretoria, 1979

STRASSMAN, P.A. : Managing the costs of information, Harvard Business Review, Sept/Oct, 1976

STRASSMAN, P.A. : Stages of growth, Datamation, October, 1976

SULCAS, P. : Management accounting for in-house computers, Unpublished doctorial thesis, The University of Stellenbosch, 1978

SUTERMEISTER, R.A. : People and productivity, McGraw Hill Book Co, New York, 1969

TAYLOR, W.J. and WATLING, T.F. : Successful project management, Business Books Ltd, London, 1970

TEICHROW, D. and HERSEY, E. : PSL/PSA : A computer-aided technique for structured documentation and analysis, IEEE Transitions in software engineering Vol, se-3, January, 1977

TEICHROW, D. and SAYANI, A. : Automation of system building, Datamation, August, 1971

THE DIEBOLD GROUP INC. : Automatic data processing handbook, McGraw Hill Book Co, New York, 1977

THE SOUTH AFRICAN COMPUTER USERS HANDBOOK : Systems Publishers, Braamfontein, 1978

THIERAUF, R.J. : Systems analysis and design of real time management information systems, Prentice Hall, Englewood Cliffs, N.J., 1975

THORNDIKE, R.L. and HAGER, E. : Measurement and evaluation in psychology and education, 3rd edition, John Wiley and Sons, New York, 1969

TULL, D. and ALBAUM, G. : Survey research: A decisional approach, International Textbook Co, Aylesbury, England, 1973

VAN DER POEL, K.G., KAGAN, M.M. : Defining the benefits of information systems, Systems/Stelsels, August, 1979

VAN DER POOL, J.A. : Trends naar en in gedistribueerde gegevens verwerking, Informatie, May, 1978

VAN GELDER, A. : Structured programming in Cobol, Communications of the ACM, January, 1977

VAN WYK, W. and KAMFER, L. : Software development, a new technology, Systems/Stelsels, September, 1976

VAN WYK, W. : Hitting the moving target, Systems/Stelsels, August, 1978

VOLLMAR, H. : De planning van de automatisering, Informatie, November, 1977

VON BERTALANFFY, L. : The history and status of general systems theory, in : Couger J.D. and Knapp R.W. (Ed.), Systems Analysis Techniques, John Wiley and Sons, New York, 1974

WAGNER, F.V. : Is decentralisation inevitable, Datamation, November, 1976

WALSTON, C.E., FELIX, C.P. : A method of programming measurement and estimation, IBM Systems Journal, No.1, 1977

WARNIER, J.D. : Logical construction of programs, Van Nostrand Reinhold Co, New York, 1976

VAN DER POEL, K.G.: Programmatuur - maken of kopen, Informatie, October 1981.

YOURDON, E. : A brief look at structured programming, Systems/Stelsels, May, 1974

YOURDON, E. : Techniques of program structure and design, Prentice Hall Inc, Englewood Cliffs, N.J., 1975

YOURDON, E., CONSTANTINE, L.L. : Structured design, Prentice Hall Inc, Englewood Cliffs, N.J., 1979

YASAKI, E.K. : The many faces of the DBA, Datamation, May, 1977

ZANI, W.M. : Blueprint for MIS, Harvard Business Review, November/ December, 1970

ZETTERBERG, H.L. : On theory and verification in sociology, The Bedminster Press, New York, 1965

ZIMMER, I. : DP management in a production environment, Systems/Stelsels, November, 1978