

**REQUIREMENTS FOR EFFICIENT COMMERCIAL SYSTEMS
ANALYSIS AND DESIGN**

by

RIAAN TERBLANCHE. IAC. NHD (Computer Systems).

**Dissertation presented in fulfilment of the
requirements for the degree**

**MASTER OF TECHNOLOGY
(Information Technology)**

in the

FACULTY OF COMMERCE

at the

CAPE TECHNIKON

Study leader: DR. P J S BRUWER

**OCTOBER 1996
CAPE TOWN**

ACKNOWLEDGEMENTS

I would like to express sincere appreciation to my supervisor, Dr P J S Bruwer of the Department of Information Technology, for his academic and moral support for the major part of this research project. The many ideas and suggestions that paved the way for the research are greatly appreciated.

I am also appreciative for the initial guidance from Prof Steve Rossouw without whom I would never been able to start this research.

A further note of thanks to the other members of staff at the School of Business Informatics of the Cape Technikon, who were prepared to lend an ear and assistance from time to time as well as Ida Hannan for editing the script.

I would like to thank the Cape Technikon's Research Fund for the financial assistance granted.

The opinions expressed in this publication, or conclusions arrived at, are those of the author and are not necessarily to be attributed to the School of Business Informatics.

A final word of thanks to my wife and children who demonstrated remarkable patience, understanding and support right up to the end.

I also want to thank my heavenly Father, without Whom, none of this would have been possible.

TABLE OF CONTENTS

CHAPTER 1 PROBLEM DEFINITION AND RESEARCH METHODOLOGY	1
1.1 Introduction	1
1.2 Problem statement	3
1.3 Systems development problems caused by ineffective training	4
1.4 Systems development training problems	10
1.5 Purpose of research	14
1.6 Dissertation Structure	15
1.7 Limitations	17
CHAPTER 2 SYSTEMS DEVELOPMENT LIFE CYCLES	19
2.1 Introduction	19
2.2 Systems Development Life Cycles (process models)	21
2.2.1 The Classic Life Cycle	23
2.2.2 The Modern Life Cycle	28
2.2.3 The Waterfall Model	35
2.2.4 The Rapid Prototyping Model	39
2.2.5 Incremental Model	44
2.2.6 Spiral Model	48
2.3 Comparison of process models	52
2.4 Conclusions	53

CHAPTER 3 SYSTEMS DEVELOPMENT TOOLS, TECHNIQUES AND
METHODOLOGIES

	55
3.1 Introduction	55
3.2 Systems Development tools and techniques	57
3.2.1 Data Flow Diagrams	59
3.2.2 Entity Relationship Diagrams	60
3.2.3 Decision trees and tables	61
3.2.4 Inspections and walkthroughs	66
3.2.5 Fact-finding techniques	71
3.2.6 Feasibility analysis	75
3.2.7 Cost Benefit Analysis	77
3.2.8 System Flowcharts	80
3.2.9 Pseudocode	81
3.2.10 Cross-Reference: Data and Process Model	83
3.2.11 Logicalization	83
3.2.12 Location Connectivity Diagrams	89
3.2.13 Data Analysis	91
3.2.14 Event Analysis	93
3.2.15 PERT charts (diagrams)	96
3.3 Systems Development Methodologies	103

CHAPTER 4 SYSTEMS DEVELOPMENT PRINCIPLES AND REQUIRED

SKILLS

107

4.1	Introduction	107
4.2	Systems development principles	107
4.3	Systems development skills required	110

CHAPTER 5 EMPIRICAL RESEARCH

115

5.1	Introduction	115
5.2	Research methodology	115
5.2.1	Development of questionnaire	116
5.2.2	Sampling procedure	122
5.2.3	Evaluation of responses to survey	124
5.3	Analysis of data	126
5.3.1	Frequency analysis	126
5.3.2	Regression analysis	145
5.4	Conclusions	150

CHAPTER 6 INTERPRETATION OF RESULTS AND CONCLUSIONS	151
6.1 Introduction	151
6.2 Frequency analysis	151
6.2.1 Systems development life cycle	152
6.2.2 Tools and techniques	154
6.2.3 Skills	165
6.3 Regression analysis	169
6.3.1 Full population	170
6.3.2 Information technology and project managers population	172
6.3.3 Systems analysts and analyst/programmers population	174
6.4 Model for efficient Systems Analysis and Design training	177
6.5 Additional considerations	181
6.6 Conclusions	183

APPENDICES:

- Appendix A: Case study: Textbook Inventory System
- Appendix B: Questionnaire
- Appendix C: Results of frequency analysis

SUMMARY/OPSOMMING

BIBLIOGRAPHY

CHAPTER 1: PROBLEM DEFINITION AND RESEARCH METHODOLOGY

1.1 INTRODUCTION

Computers are part of our lives. In most countries world-wide computers are used to support the day-to-day operations of governments and business enterprises. This, in effect, affects the lives of everybody dealing with those institutions.

Most people know that a computer should be programmed to execute instructions in order to produce what is required by the user. The problem, however, is that people tend to focus only on a single program, without realising that a single program is one component of something much broader. Someone must have planned not only the program, but the entire environment in which the program exists. This 'something broader' is what is referred to as a system. In information technology, "a system is a collection of men, machine and methods organized to accomplish specific functions" (Davis, 1983:4).

The process of planning such a system is called systems analysis and design. "Systems analysis and design is a systematic approach to identifying problems, opportunities and objectives; to analysing the information flows in organizations; and to designing computerized

information systems to solve a problem" (Kendall, 1992:12).

"It is often difficult to explain what is achieved by systems analysis and design - especially when talking to a user who wants the system tomorrow. After all, their 14-year old son can knock together a quick program on his PC in a couple of hours. Surely, a larger system is just the same - just a little bit bigger. Why, then, should it take so long to design a system" (Ashworth, 1990:6).

Furthermore, what is the difference between a systems analyst (or a systems developer) and a programmer? Surely, they perform the same tasks - writing programs. This is the perception of many users, the systems developer's clients.

The above two perceptions of developing information systems constitute the main problem in business today. People using information systems are not aware of all the effort that is required to develop information systems. Most users know that programs need to be written. That is most probably the reason why users often approach a systems developer with a statement such as: "I want you to write a computer program to". Users are not able to differentiate between programs and systems.

Some more computer-literate users believe that building systems is

exactly the same as building a bridge. Therefore, it can be regarded as an engineering process. It should be admitted that bridges collapse, but far less frequently than, for example, accounting systems. This causes what is termed as the software crisis.

All the above are symptoms of problems. Most of the real problems still need to be properly defined, analyzed and hopefully, solved.

1.2 PROBLEM STATEMENT

A problem is a question or situation that presents uncertainty, perplexity or difficulty. It is therefore difficult to handle and is an obstacle.

Most of the symptoms described in the previous section are usually the result of problems caused by ineffective training. Some of these problems are described in section 1.3.

Ineffective training is the major problem, but it is caused by obstacles experienced during Systems Development training. These problems are described in section 1.4.

1.3 SYSTEMS DEVELOPMENT PROBLEMS CAUSED BY INEFFECTIVE TRAINING

For the purpose of this study, the general problems in systems development that can to a large extent be eliminated by proper training, are as follows:

1.3.1 LACK OF KNOWLEDGE AND UNDERSTANDING

One of the major problems experienced in systems development is the lack of knowledge and understanding of the basic principles, procedures, techniques and methods being used when systems are built.

1.3.2 LACK OF PROPER COMMUNICATION

Developers and users do not communicate properly. Confusion even amongst developers themselves exist. Goldsmith (1993:5) states that "people involved in software development tend to come from a huge range of backgrounds - you do not just get 'trained computer scientists' as in a number of other areas of engineering. As long as someone is numerate, they can end up doing software development. As these people have different backgrounds and have been trained in different ways, they will do development

differently."

1.3.3 DIFFERENCE BETWEEN SYSTEMS DEVELOPER AND USER

A further problem is that people do not really know what is meant by a systems developer (or systems analyst) and a systems user. It is therefore necessary to define these two terms right at the outset:

According to Whitten (1994:8) "the systems analyst studies the problems and needs of an organization to determine how people, data, processes, communications and technology can best accomplish improvements for the business. When computer technology is used, the analyst is responsible for the efficient capture of data from its business source, the flow of that data to the computer, the processing and storage of that data by the computer, and the flow of useful and timely information back to the business and its people."

Edwards (1993:10) states that "the systems analyst is the person (or persons) who guides the analysis, design, implementation, and maintenance of a given system. In performing these four tasks, the analyst must always match the information system objectives with the goals of the organization."

Modell (1988:8) describes the systems analyst as "one who engages in the study of, and separation of, a group of interacting, interrelated, or interdependent business functions, processes, activities or elements forming a complex whole into its constituent parts for individual study."

For the study purposes, a systems analyst is a person who

- * identifies, defines and studies business problems to determine if it can be solved by a computer system,
- * studies the current system in detail to determine if it is worth solving the problem,
- * identifies, describes and designs alternative solutions to the problem,
- * communicates these alternatives through to the user so that the user can select the best alternative,
- * designs the selected alternative in detail,
- * writes the specifications for the programmers to construct the solution in computer terms,
- * trains users to use the new system and
- * assists users when the new system is in production.

On the other hand "system users are the people who use (and directly benefit from) the information system on a regular basis - capturing, validating, entering, responding to and storing data and

information." (Whitten, 1994:42).

Kendall (1992:3) defines a system user as "anyone who interacts with an information system in the context of his or her work in the organization."

For the purpose of this study a system user is anyone using an information system for the purpose of doing his work in the organization.

1.3.4 RATE OF INCREASING NEED FOR SYSTEMS

The need for computer systems has increased at an alarming rate. "Our society has adopted the use of computers at a rate that staggers the imagination. Every segment of society has discovered programs that seem to make its job more productive. This demand for programs has required a tremendous amount of code to be written by a population of producers whose training and tradition are recent." (Goldberg, 1986:334)

Although the above statement by Goldberg in respect of programmers was made in 1986, the same can be said of systems analysts today. In 1986 programs were needed; today systems are needed to solve users' problems.

Time is of the essence. User acceptable systems must be developed quickly and correctly. It is therefore of vital importance that the systems analyst has the right approach to problem-solving, and be able to apply the best available tools and techniques to analyze, design and implement a system as quickly as possible.

1.3.5 DIFFERENT APPROACHES TO SYSTEMS DEVELOPMENT

There are numerous approaches, principles, tools and techniques (APTT). It is up to the systems analyst to decide which of these to apply in a given situation. For the systems analyst to make such a decision, he should know and understand as many as possible of these APTT's.

It is therefore necessary to investigate all the different APTT's and then select those that should be taught to prospective systems analysts so as to equip them sufficiently for a career as a systems analyst.

1.3.6 VARIETY OF SKILLS REQUIRED TO BECOME A SYSTEMS ANALYST

Systems analysts should be "people who communicate with

management and users at the management/user level; document their experience; understand problems before proposing solutions; think before they speak; facilitate systems development, not originate it; are supportive of the organization in question and understand its goals and objectives; use good tools and approaches to help solve systems problems; and enjoy working with people" (Whitten, 1994:19).

Therefore the following skills and attributes are most frequently cited by practising systems analysts:

- * Working knowledge of information systems and technology.
- * Computer programming experience and expertise.
- * General business knowledge.
- * Problem-solving skills.
- * Interpersonal communications skills.
- * Interpersonal relations skills.
- * Flexibility and adaptability.
- * Character and ethics.
- * Systems analysis and design skills.

1.4 SYSTEMS DEVELOPMENT TRAINING PROBLEMS

1.4.1 GENERAL TRAINING PROBLEMS

It is very important that systems developers and systems users are trained properly in all (or at least most) of the aspects required to communicate and understand information systems. The training, however, is very difficult due to the following:

- 1.4.1.1 There are several approaches to systems development.
- 1.4.1.2 The most popular systems development process (the Systems Development Life Cycle) has not been standardised as yet.
- 1.4.1.3 Technology changes and improves at a tremendous rate. This causes new tools and techniques to be developed at an alarming rate.
- 1.4.1.4 Trainers are not always experienced in all the different approaches and processes of systems development.
- 1.4.1.5 It is extremely difficult to obtain literature that covers all the aspects of systems development in a single volume for use by students and trainees.
- 1.4.1.6 Prospective systems analysts and users should be trained to perform tasks and duties in industry. Sometimes the industry requires skills that cannot be acquired from a textbook or in a class situation. *These skills can only be acquired by exposure and*

experience.

1.4.2 TRAINING PROBLEMS AT TERTIARY INSTITUTIONS

The following problems are experienced in training prospective systems analysts at an institution such as the Technikon:

1.4.2.1 INEXPERIENCED STUDENTS

The majority of students never had any working experience before they started their studies. Students do not really appreciate the complexities of the industry and therefore regard many of the terms used in systems analysis and design as pure theory. Many students cannot visualise the industry environment; therefore having a limited framework to draw from.

1.4.2.2 LACK OF EXPOSURE TO COMPUTERS

Large numbers of students never had exposure to any type of computer-related environment before they started their studies.

1.4.2.3 STUDENT EVALUATION

It is very difficult to evaluate students' performance as systems

analysts when one is restricted to tests, examinations and customized projects only. At a tertiary institution every student should be asked the same set of questions and be given a performance mark (e.g. 65% or 70%). One cannot simply pass or fail a student on work performed on a project. A specific mark should be awarded for the work. Students are not exposed to real-life industry problems; therefore a realistic evaluation cannot be made.

1.4.2.4 LARGE GROUPS

There are sometimes more than 100 students in one class. That means that a student cannot be exposed to the use of modern technology such as computer-aided software engineering (CASE). The reason being that it is too expensive to acquire 100 CASE-tool workstations so that each student can develop a system using this technology.

1.4.2.5 PRESCRIBED MATERIAL

The training institution should prescribe a text book for the student to use as a reference. The lecturer is not always sure that the prescribed book is generally accepted in the industry.

1.4.2.6 EXCESSIVE INTERRUPTIONS

It is not always possible for a student to work on a system problem until it is solved due to the fact that he has to stop at the end of a period to attend classes in other subjects. Working on this problem can only be continued during the next period which sometimes takes place 3 to 5 days later. The result of this is that the student forgets what was done in the previous period.

1.4.2.7 LACK OF INTEREST

Students have to do Systems Analysis and Design as part of a diploma or degree course. A large number of students are not really interested in Systems Analysis and Design and therefore regard this subject as pure theory and sometimes frustrating. This attitude is often found among programming students who are used to getting evaluating results after a relative short period of time (the time it takes to build a program).

1.4.2.8 CHANGING TECHNOLOGY

It is generally accepted that technology changes very regularly and rapidly in Information Technology. This tendency causes a major problem in teaching the latest technology to students due to the

following:

- * It is very difficult to obtain proper prescribed material that covers the latest technology sufficiently for studying purposes.
- * The lecturer most often lacks practical experience in the new technology.
- * Courses cannot be adjusted rapidly to be in line with the latest technology.
- * Hardware and software required for the new technology cannot be acquired easily.

1.5 PURPOSE OF RESEARCH

The purpose of this research is to determine

- * Which APTT's and skills should be taught to undergraduate students so as to equip them sufficiently for starting a career as a systems analyst
- * Whether the topics covered by the course are sufficient for the needs of industry and how to ensure that the most acceptable books and/or publications will always be prescribed
- * Which organizations in the industry can assist the Cape Technikon in the training of systems analysts. This support is required in terms of equipment, expertise, finance and real-life projects.

- * How the industry can support the Cape Technikon in accommodating the changing technology.

1.6 DISSERTATION STRUCTURE

The first part of the research contains an overview of the aspects that are taught to undergraduate students at the Cape Technikon. This part consists of:

- * Chapter 2: a brief overview of the Systems Development Life Cycles (approaches),
- * Chapter 3: the various basic tools and techniques used in systems development,
- * Chapter 4: systems development principles and skills required to become a successful analyst.

This part also contains many references to the prescribed books for this course. These books are:

WHITTEN, JEFFREY L, BENTLEY, LONNIE D, BARLOW, VICTOR M.
1989. Systems Analysis and design methods. 2nd ed.
Homewood, Boston: Richard D Irwin.

WHITTEN, JEFFREY L, BENTLEY, LONNIE D, BARLOW, VICTOR M.

1994. *Systems Analysis and design methods*. 3rd ed.
Homewood, Boston: Richard D Irwin.

SCHACH, STEPHEN R. 1992. *Practical Software Engineering*.
Homewood, Boston: Richard D Irwin.

The method of application of the various techniques and tools as taught at the Technikon are illustrated in detail. The reason for this is that in a number of cases the methods had to be slightly adjusted to make it easier for students to understand and remember. From experience a student finds it easier to be taught how to use a tool or technique when he has to do it stepwise or according to an algorithm.

The second part of the research (chapter 5 and 6) describes -

- * the survey in various sectors of the industry to determine the most popular APTT's and skills, books and publications being used. This survey also includes the determination of the popularity of various programming languages used by industry for implementation purposes. For survey purposes a questionnaire containing relevant questions is used and
- * the analysis of the responses and the subsequent conclusions regarding those aspects that should be taught to prospective systems analysts with special reference to tertiary institutions such

as the technicians.

1.7 LIMITATIONS

The first problem encountered during the research was to determine a valid and acceptable study population due to the large number of businesses and institutions in the country using computer-based information systems. The study population had to be that part of the industry to which the Cape Technikon can supply qualified systems analysts.

A large number of the selected study population were not prepared to participate in the research. The majority of this group felt that they had not enough time to pay attention to anything else but their own work.

The response from a large portion of those who were willing to participate was very slow. The closing date for the return of the completed copies of the questionnaire had to be extended by more than a month. There were still many copies received more than one month after the extended closing date.

The questionnaire was also sent to a substantial part of the study population by internet (e-mail). The response to this method was very low (less than 5%).

Another limitation of this research was the *difference in interpretation of the terms by researcher and respondent used in the questionnaire.*

Although most of the terms were defined in an annexure to the questionnaire, many respondents connoted a different meaning to the term.

CHAPTER 2: SYSTEMS DEVELOPMENT LIFE CYCLES

2.1 INTRODUCTION

This chapter deals with the most important concept that prospective systems analysts should understand; namely the systems development life cycle (SDLC). The building of computer-based information systems to serve the management and operation of organizations is a complex, expensive and time-consuming process. It is therefore of vital importance that a guideline is needed to execute the required tasks for building systems in the correct sequence, using the right methods. This is exactly what the SDLC is all about.

Unfortunately, there is not only one SDLC being used by systems developers. It is therefore necessary to describe some of the most basic SDLC's found in the literature. These SDLC's then need to be briefly compared with the purpose of illuminating the problem in the training of prospective analysts.

For the purpose of this research a SDLC is defined as "a systematic and orderly approach to solving business problems, and developing and supporting resulting information systems, a process by which systems analysts, software engineers, programmers, and end-users build

information systems and computer applications, a project management tool to plan, execute and control systems development projects"

(Whitten, 1994:11, 91). Synonyms for the SDLC can be software process model, software model, software process, project life cycle or simply life cycle. Schach (1992:47) refers to the SDLC as "the process model".

The SDLC is firstly an approach to solving problems. The term 'problem' in systems development can be regarded as anyone of the following:

- * An undesirable situation that prevents the organization from fully achieving its purpose, goals and objectives.
- * An opportunity to improve the organization even in the absence of a specific problem.
- * A directive which is a new requirement imposed by management, government or some external influence.

Whitten (1994:92) describes the seven steps of the classical approach to solving problems as follows:

- * Identify the problem.
- * Understand the problem's environment and the problem's causes and effects.
- * Define the requirements of a suitable solution.

- * Identify alternative solutions.
- * Select the 'best' solution.
- * Design and implement the solution.
- * Observe and evaluate the solution's impact. Refine the solution accordingly.

It was therefore necessary to establish a process by which systems can be developed based on a proper problem-solving approach. Although there are major differences in the SDLC's found in the literature, all of them basically satisfy the requirements of the above problem-solving approach.

2.2 SYSTEMS DEVELOPMENT LIFE CYCLES

Before individual SDLC's are discussed, two general remarks about all the SDLC's inspected need to be mentioned:

Most of the SDLC's found in the literature agree that the systems development process should

- * be broken down into phases and tasks,
- * establish standards for consistent development and documentation,
- * establish the scope of the project in accordance with user involvement and that it should be revised from time to time,
- * get the user involved during the whole process,

- * allow that the system can be designed for growth and change.

Although the SDLC's have sometimes completely different names for the various phases and tasks, most of them agree on the tasks that should be performed during the systems development process. The major differences, however, are in the grouping of these tasks into phases.

The various life cycles described in the prescribed books (refer page 13) are briefly discussed. These life cycles and the corresponding sources are:

- * Classic Life Cycle as described by Whitten (1989)
- * Modern Life Cycle as described by Whitten (1994)
- * Software Process Models such as the Waterfall Model, the Rapid Prototyping Model, the Incremental Model and the Spiral Model as described by Schach (1992).

Tables and diagrams are abundantly used to alleviate long narrative descriptions of life cycles and the various phases, sub-phases and tasks.

Remarks and/or evaluations of the various life cycles are based on the views by the above authors in the prescribed books, unless specific reference is made to someone else.

2.2.1 THE CLASSIC LIFE CYCLE

Sommerville (1982:3) says that "the initial model of the SDLC was probably first proposed by Royce (1970) in his work *Managing the Development of Large software Systems: Concepts and Techniques*" and, since then, there have been numerous refinements to and variations of this model.

Whitten (1989:100) is of the opinion that the Classic Life Cycle, in its simplest form, consists of the following four phases:

- * Systems Analysis which is the study of a current business system and its problems, the definition of business needs and requirements, and the evaluation of alternative solutions.
- * Systems Design which is the general and detailed specification of a computer-based solution that was selected during systems analysis.
- * Systems Implementation which is putting the system into operation and where computer programs need to be written and tested, users trained to use the new system and the operations converted to the new system.
- * Systems Support which is the ongoing maintenance of the system after it has been put into operation.

Each of the first three phases can be divided into sub-phases which

consist of various tasks (refer tables 2-1 to 2-3). Strictly speaking these tasks need to be performed in a specific sequence although many situations exist where tasks (and even phases) can overlap. This depends on project sizes and resources committed to the project.

The Systems Support phase consists of maintenance activities that are essential to the system to correct errors and omissions. It also includes activities aimed at improving the system.

SYSTEMS ANALYSIS

SUB PHASE	TASKS	INPUT	OUTPUT
Survey	Survey the feasibility of the project	Project request and facts	Feasibility survey report
Study	Learn how current system operates; Analyze problems, limitations and constraints; Brainstorm preliminary solutions; Update scope and feasibility.	Approved feasibility survey report	Problem statement
Definition	Define system objectives and priorities; Outline requirements for new system; Define detailed requirements	Problem statement	Requirements statement
Selection	Specify alternative solutions; Analyze feasibility of alternative solutions; Prepare designs and implementation schedules; Sell the system	Requirements statement (as well as general requirements)	System proposal

TABLE 2-1 (Based on Whitten, 1989)

SYSTEMS DESIGN

SUB PHASE	TASKS	INPUT	OUTPUT
Acquisition	Research technical criteria and options; Solicit proposals (or quotes) from vendors; Validate vendor claims; Evaluate and rank vendor proposals; Award (or let) contract and debrief unsuccessful vendors	System proposal	Contract with vendor
Design	Design computer files and databases; Design computer outputs; Design computer inputs; Design on-line user interface; Design methods and procedures; Design computer specifications; Present and review design	System proposal	Computer hardware and software specifications; Implementation plan; Final Cost-Benefit analysis

TABLE 2-2 (Based on Whitten, 1989)

SYSTEMS IMPLEMENTATION

SUB PHASE	TASKS	INPUT	OUTPUT
Construction	Install hardware and software; Plan for programming; Build test data, test files and databases; Write and test programs; Install applications software; Modify application software	Design specifications, and hardware and software configuration	Computer Programs
Delivery	Install hardware and systems software; Train end-users to use the new system; Convert to the new system; Post-implementation review	Design and program documentation	Operational information system

TABLE 2-3 (Based on Whitten, 1989)

2.2.2 THE MODERN LIFE CYCLE

In its simplest form, the Modern Life Cycle consists of the following five phases (Whitten, 1994:11):

- * Systems Planning which is the ongoing study of a problem environment to identify problem-solving possibilities with a view to select those projects which will provide the greatest long-term benefits to the organization. This function of the life cycle seeks to identify and prioritize those technologies and applications that will return the most value to the business.
- * Systems Analysis which is the study of the problem environment (the current business and information system) and subsequent definition and prioritization of the requirements for solving the problem.
- * Systems Design which is the evaluation of alternative problem solutions, and the detailed specification of the final solution. These specifications are typically sent to programmers for systems implementation. Systems Design is often referred to as the physical design phase.
- * Systems Implementation which is the construction or assembly of the problem solution, culminating in a new environment based on the solution. Once implemented the

new system is in operation.

- * Systems Support which is the ongoing maintenance and enhancement of the solution during its lifetime.

Each of the first four phases can, similar to the Classic Life Cycle, be divided into sub-phases which consist of various tasks (refer tables 2-4 to 2-7). Strictly speaking these tasks also need to be performed in a specific sequence although many situations exist where tasks (and even phases) can overlap. This depends on project sizes and resources committed to the project.

SYSTEMS PLANNING

SUB PHASE	ACTIVITIES	INPUT	OUTPUT
Study the business mission	Establish the planning team; Define planning scope and expectations; Identify business performance measures; Develop a project plan; Review findings and communicate planning vision.	Business mission	Business plans
Definition of an information architecture	Model the enterprise; Asses current business strategies; Assess current information services and strategies; Identify and prioritize business areas; Complete the new information architecture; Identify and plan subsequent projects; Review findings and approve plan.	Business plans	Business areas (a group of logically related business functions and activities)
Business Area Analysis	Establish the analysis team; Identify business area performance measures; Model the business area; Assess current business area and IS performance; Identify and prioritize development projects; Plan application development strategy and projects; Review findings and approve plan.	Business areas	Planned application development project

TABLE 2-4 (Based on Whitten, 1994)

SYSTEMS ANALYSIS

SUB PHASE	ACTIVITIES	INPUT	OUTPUT
Survey project feasibility	Conduct initial interview(s); Define project scope; Classify problems, opportunities and possible solutions; Establish a proposed project plan; Present survey findings and recommendations.	Unplanned application development project and existing system limitations	Statement of project scope and a preliminary feasibility assessment
Study and analyze the current system	Assign project roles; Learn about current system; Model the current system; Analyze problems and opportunities; Establish new system's objectives; Modify project scope and plan; Review findings and recommendations.	Statement of project scope as well as existing system limitations and details	New system objectives and a business problem statement
Definition of user's requirements	Identify requirements; Model system requirements; Build discovery prototypes; Prioritize business requirements; Modify project plan and scope; Review requirements specifications.	New system objectives and requirements and priorities of users	Business requirements statement

TABLE 2-5 (Based on Whitten, 1994)

SYSTEMS DESIGN

SUB PHASE	ACTIVITIES	INPUT	OUTPUT
Selection of a design target	Specify alternative solutions; Analyze feasibility of alternative solutions; Recommend a system solution.	Business requirements statement	Design requirements and a system proposal
Acquisition	Research technical criteria and options; Solicit proposals (or quotes) from vendors; Validate vendor claims and performance; Evaluate and rank vendor proposals; Award (or let) contract and debrief unsuccessful vendors; Establish integration requirements.	Hardware and software package requirements and proposals and quotations	Purchase orders and contracts; Integration requirements
Design and integration	Analyze and distribute data; Analyze and distribute processes; Factor into design units; Design computer files and/or databases; Design computer outputs and inputs; Design on-line user interfaces; Present and review design	Design and integration requirements	Technical design statement

TABLE 2-6 (Based on Whitten, 1994)

SYSTEMS IMPLEMENTATION

SUB PHASE	ACTIVITIES	INPUT	OUTPUT
Build and test networks and databases	Build and test networks; Build and test databases	Technical design statement	Networks and unpopulated databases
Build and test programs	Plan for programming; Write and test programs	Technical design statement; Networks and unpopulated databases	Uninstalled computer programs
Install and test new system	Install new software package; Test package; Conduct system test; Prepare conversion plan	Technical design statement, uninstalled computer programs and software packages	Installed system
Deliver new system into operation	Install files and/or databases; Train system users; Convert to new system; Evaluate project and system	Technical design statement and installed system	Production information system and end-user documentation

TABLE 2-7 (Based on Whitten, 1994)

The Modern Life Cycle is very similar to the Classic Life Cycle although there are many differences in the grouping of tasks into phases. The biggest difference, however, is the addition of Systems Planning as the first phase. The reason for this could be that the development of computer systems has become such a major function and integral part of businesses today that top-management has to plan carefully and budget for it. Another reason could be that the importance of information technology has been acknowledged to such an extent that separate departments have been established for systems development and programming. The manager or director in charge of this department forms part of the top-structure of the business. Therefore systems development is planned similar to the other functions such as marketing and stores of the business.

The Systems Support phase again consists of maintenance activities, essential to the system, to correct errors and omissions in the system. It also includes activities aimed at improving the system as well as recovering the system from system failures or 'crashes'.

2.2.3 THE WATERFALL MODEL

The Waterfall Model of which a version appears as Figure 2-1, was first put forward by Royce (Schach, 1993:49) in 1970. It consists of the following phases:

* Requirements phase:

The concept is explored and refined, and the user's requirements are established, analyzed and agreed.

* Specification phase:

The user's requirements are presented in the form of a specification document indicating what the product (the system) is supposed to do.

* Planning phase:

A software project management plan detailing every aspect of the proposed software development is drawn up.

* Design phase:

During this phase the specifications undergo two consecutive design processes. In the first place the architectural model of the product is designed. This means that the product as a whole is broken down into components or modules. In the second stage each module is designed in detail. This indicates what the product must do and how it must be done.

- * Implementation phase:

The various components are coded and tested.

- * Integration phase:

The various components of the product are combined and tested as a whole. When the developers are satisfied that the product satisfies all the requirements, the product needs to be tested by the user. This is called the acceptance test. Once the user accepts the product, this phase ends.

- * Maintenance phase:

Maintenance includes all the updates to the product as a result of changes, enhancements and implementation of directives (as imposed by management or government). Maintenance can be categorized in two groups, namely corrective maintenance and enhancements. Corrective maintenance is mainly the correction of product errors, while enhancements can be broken down into a further two types, namely perfective and adaptive maintenance. The first type includes changes to be made to the product because the user thinks it might improve the situation, while the second type refers to directives from management or government.

- * Retirement phase:

The product is removed from service.

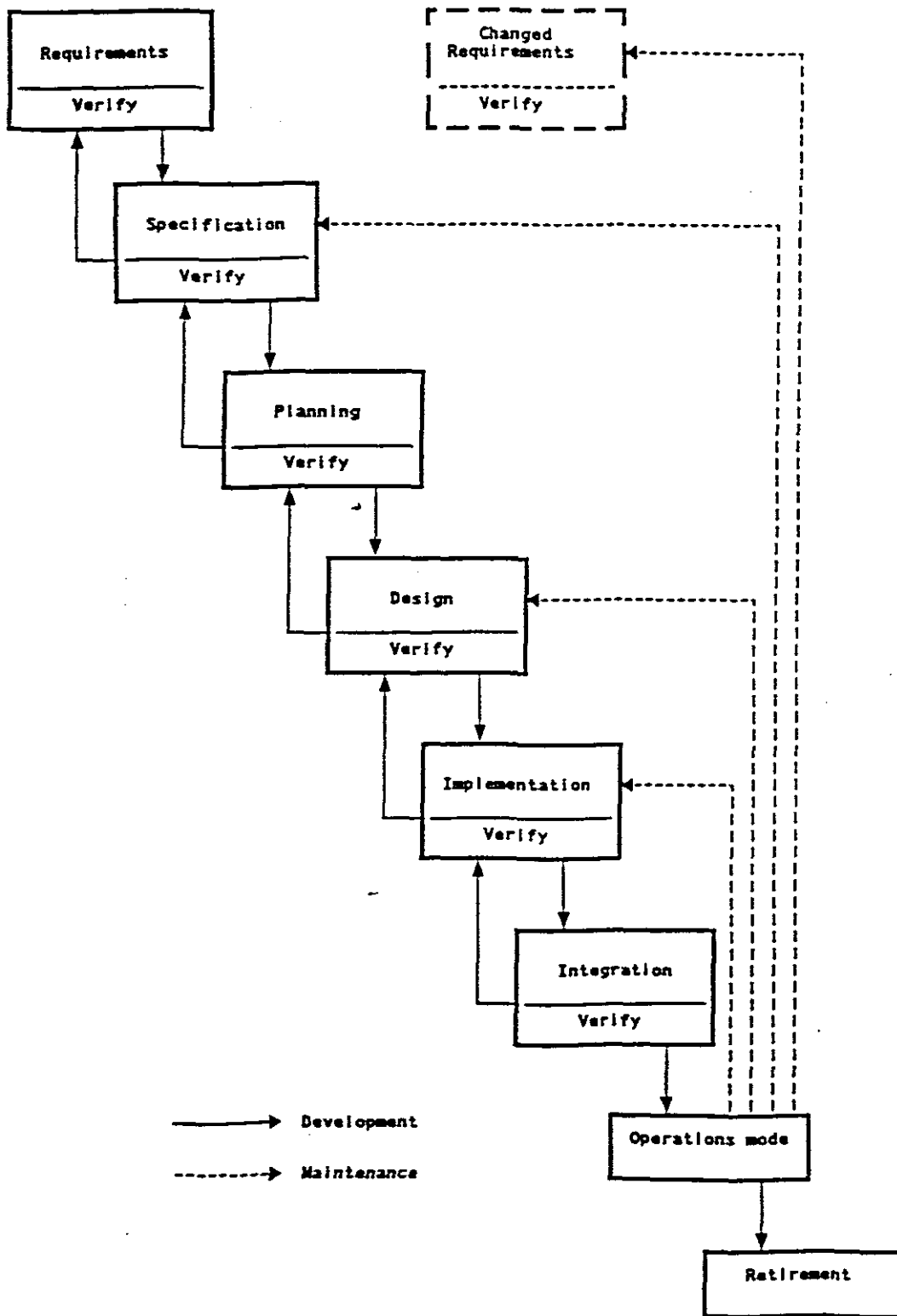


Figure 2-1: Waterfall Model (Schach, 1993:50)

A critical point about the Waterfall Model is that no phase is complete until the documentation produced during that phase has been drawn up, agreed upon by all parties involved, confirmed by the user and checked and approved by the quality assurance group. The mere fact that this enforced discipline is built into the Waterfall Model, makes it one of the main advantages of this software process Model.

Another inherent aspect of the Waterfall Model phases is testing. Testing is not a phase on its own to be done at the end of the development process, it forms an integral part of each phase. Each and every document should be checked and verified, and each and every component should be tested individually and as part of the whole.

Compared to the Classic and Modern Life Cycles, the Waterfall Model caters only for the design and implementation processes of the whole development process. It assumes that project planning and the study and analysis of the current situation are excluded from the software engineering process.

There is one shortcoming in the Waterfall Model (as is the case in many other systems development life cycles and/or software process models) and that is the difference between the way a user

understands a product as described by a specification document and the actual product. The user cannot understand a system which is described as a specification on paper. It is therefore possible that the product that is eventually delivered to the user might not be what the user really wanted. It will be much better if the user could be exposed to something more real - something like a prototype.

2.2.4 THE RAPID PROTOTYPING MODEL

The strength of the Rapid Prototyping Model is that it can help to ensure that the user's real needs are met. Prototyping enables the system to be developed from the users' perspective.

Vonk (1990:20) refers to a prototype as "a working model of (parts of) an information system, which emphasizes specific aspects of that system. Therefore prototyping refers to an approach for establishing a systems requirements definition which is characterized by a high degree of iteration, by a very high degree of user participation in the development process and by extensive use of prototypes".

The reason for a high degree of iteration is "that the user cannot indicate what requirements and wishes they have for a system

with no first-hand experience of it. In the process, the prototype is modified in an iterative fashion until the requirements and wishes of the users with respect to the modelled aspect/subsystem are clear. Each successive prototype forms a closer approximation to the real requirements. On the basis of the prototype, new requirements and wishes come to light, and it can be verified that the user and the developer have understood each other correctly" (Vonk, 1990:22).

"Participation in the prototyping process makes considerable demands on the user's time. In prototyping, the user is an active participant, evaluating prototypes, proposing improvements and, at the same time, continuing to obtain a deeper insight into his own requirements and wishes with respect to the information system. Because the user has only a limited understanding of the language in which system requirements definitions are drawn up, he will not often become actively involved in the definition phase and its conclusion" (Vonk, 1990:23).

The Rapid Prototyping Model of which a version appears as Figure 2-2, consists of a rapid prototyping phase as the first phase and then all the phases of the Waterfall Model from specification through to retirement. The only difference between the Rapid Prototyping and the Waterfall Model is that the user's requirements

are determined using prototypes in this model, while requirements are determined in a more traditional way (interviews, questionnaires, meetings) in the Waterfall Model.

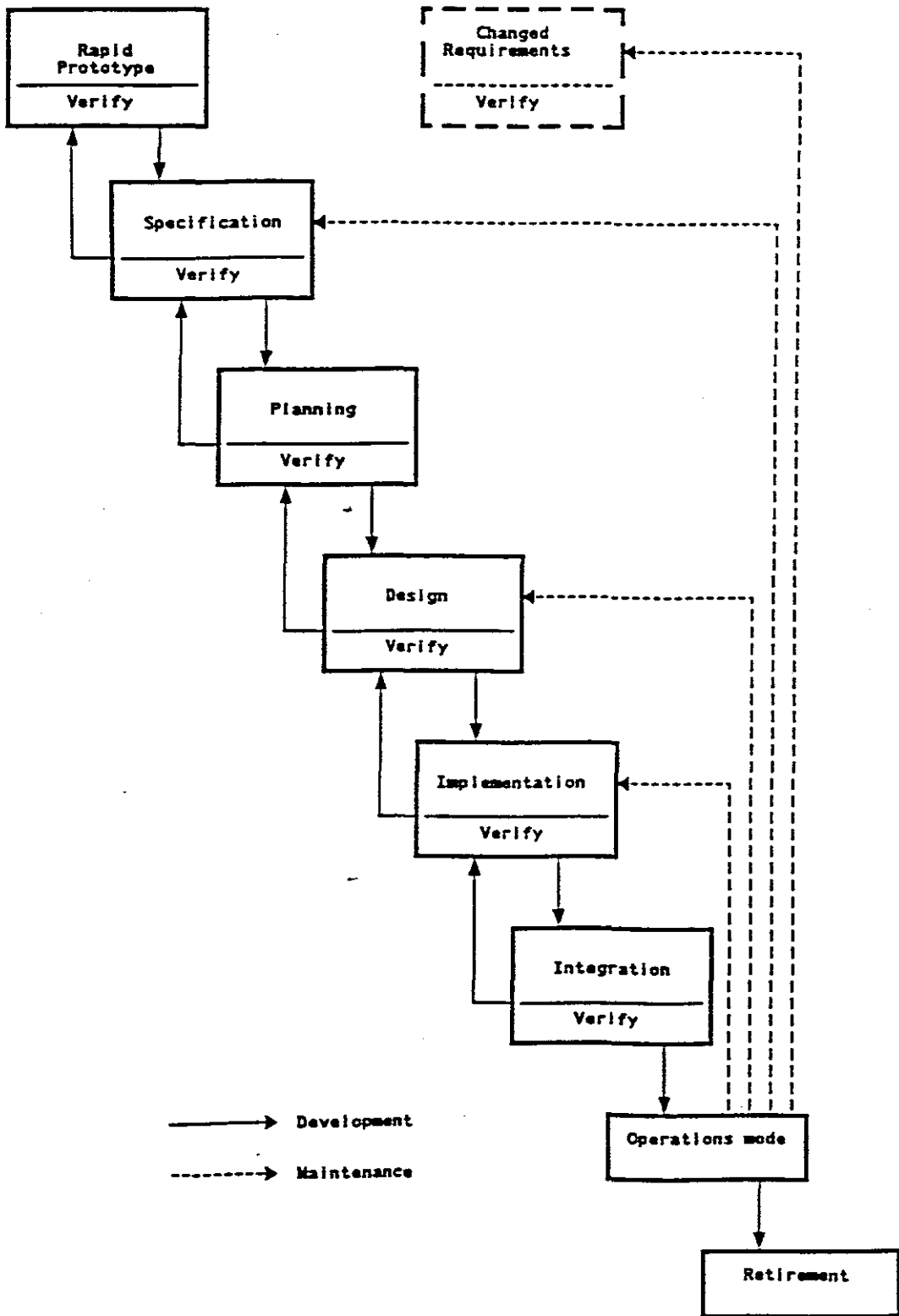


Figure 2-2: Rapid Prototyping model (Schach, 1993:55)

During the rapid prototyping phase a prototype is built, and the current and future users are allowed to interact and experiment with it. Once the user is satisfied that the rapid prototype does most of what is required, the developers can draw up the specification document knowing exactly what the users want.

What is very important in this phase, is that the prototype should be built as rapidly as possible so as to speed up the development process. It is also very important that the specifications produced during this phase are evaluated and checked by a group of users and systems analysts (called a quality assurance group) for quality.

Schach (1993:56) claims that rapid prototyping has had many successes, but it has not been proved beyond all doubt that it is the ultimate model for developing software. A solution could be to combine the Rapid Prototyping and the Waterfall approaches.

Rapid prototyping can be used as a requirements analysis technique to produce proper requirements that can serve as input to the Waterfall Model. It is also possible that the rapid prototype can serve as the specification. No time is then wasted drawing specifications, and the difficulties with specifications such as ambiguities, omissions and contradictions cannot arise. Instead, the rapid prototype which constitutes the specification states that the product should do what the prototype does. The only

additional requirements that need to be documented in a separate addendum to the specification are security, error handling, audit trails, file and database sizes.

There can be, however, major drawbacks as well. If there is a disagreement as to whether the developers have satisfactorily discharged their obligations, it is unlikely that the rapid prototype will be acceptable as an excuse for not producing the correct product. A second reason why rapid prototypes should not totally replace written specifications is potential problems with maintenance. It should always be kept in mind that systems analysts and programmers are very dependant on proper documented specifications during the maintenance phase. If the rapid prototype is the only 'specification' and it needs to be changed, corrected or updated (especially when functionality needs to be addressed), it might lead to a major effort in starting from scratch with the system.

2.2.5 INCREMENTAL MODEL

"Software is built, not written. That is to say, software is constructed step by step, in the same way that a building is constructed. With a software product in the process of development, each step consists of additional pieces that are

added to what has gone before. On one day the design is extended, the next day another module is coded. The construction of the complete product proceeds incrementally in this way until the product is finished. The basic reality is that a software product is built piecewise. The realization that in practice software is engineered incrementally has led to the development of a model that exploits this aspect of software development, the so-called incremental model" (Schach, 1993:57). A version of this model is shown in Figure 2-3.

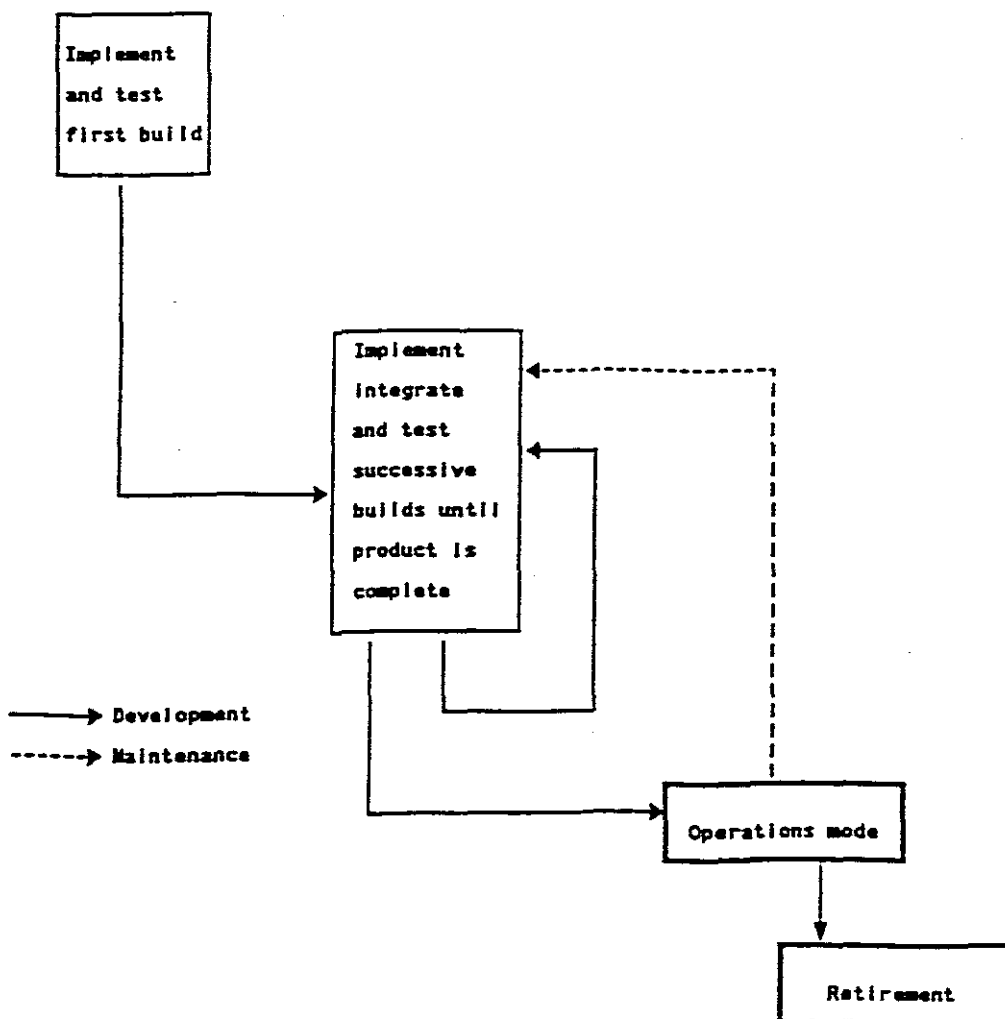


Figure 2-3: Incremental model (Schach, 1993:58)

From Figure 2-3 it is clear that this model does not really consists of phases. It is very important that a project should be broken down into manageable units or components, each with a specific deliverable product, when using this approach. Whereas the other software models concentrated on the whole system, this model concentrates more on individual units within the system. One can also assume that each unit may be approached differently in that each one can be developed using a different model (any of the above models).

The main advantage of this model is that more than one product are delivered to users at different times. Since at each stage the user has an operational quality product that does a portion of what is required, it is not necessary to wait for the final delivery date to be able to use the product. Instead, portions of the product can be available within days or weeks. This contributes towards a more gradual introduction of a changing environment for the user.

Although the user should be involved in the systems development and should be very well trained before a new system is commissioned, a whole new environment at once might be very traumatic for the user.

Another advantage is that a phased delivery does not require a large capital outlay at once. This contributes obviously to a better

cash flow situation. It is also not necessary to complete the whole project to get return on capital. It is possible for the user to stop the project at any time if it seems that further development would not render return on capital.

The major problem, however, with this model is that it can become 'untidy'. If separate units are not well incorporated into the already built structure, it can destroy the whole system. Furthermore, the existing structure should lend itself to extension in this way, and the addition of each succeeding build should be simple and straightforward.

One should be very careful not to degenerate this model into a 'build-and-fix' approach where a product is constructed without proper specifications or a design. Instead, the developers simply build a product which is reworked as many times as necessary until the user is satisfied. Obviously this will lead to problems related to project objectives such as final delivery date, the project budget and the quality of the product.

2.2.6 SPIRAL MODEL

Software development is expensive. It is therefore important to plan the development of software properly. Risk is one of the many aspects of software development that should be catered for in the planning of a software development project.

"There is almost always an element of risk involved in the development of software. Key personnel can resign before the product has been adequately documented. The manufacturer of hardware on which the product is critically dependant can go bankrupt. Too much, or too little, can be invested in testing and quality assurance." (Schach, 1993:60).

A possible solution to the above problem can be to develop a Rapid Prototype that satisfies the user's requirements and needs, and determine if the system is going to be viable and feasible. This is to minimize the risk.

"The idea of minimizing risk via the use of prototypes and other means is the concept underlying the spiral model. A somewhat simplistic way of looking at this process model is as a waterfall model with each phase preceded by risk analysis" (Schach, 1993:61). An example of this is shown in Figure 2-4.

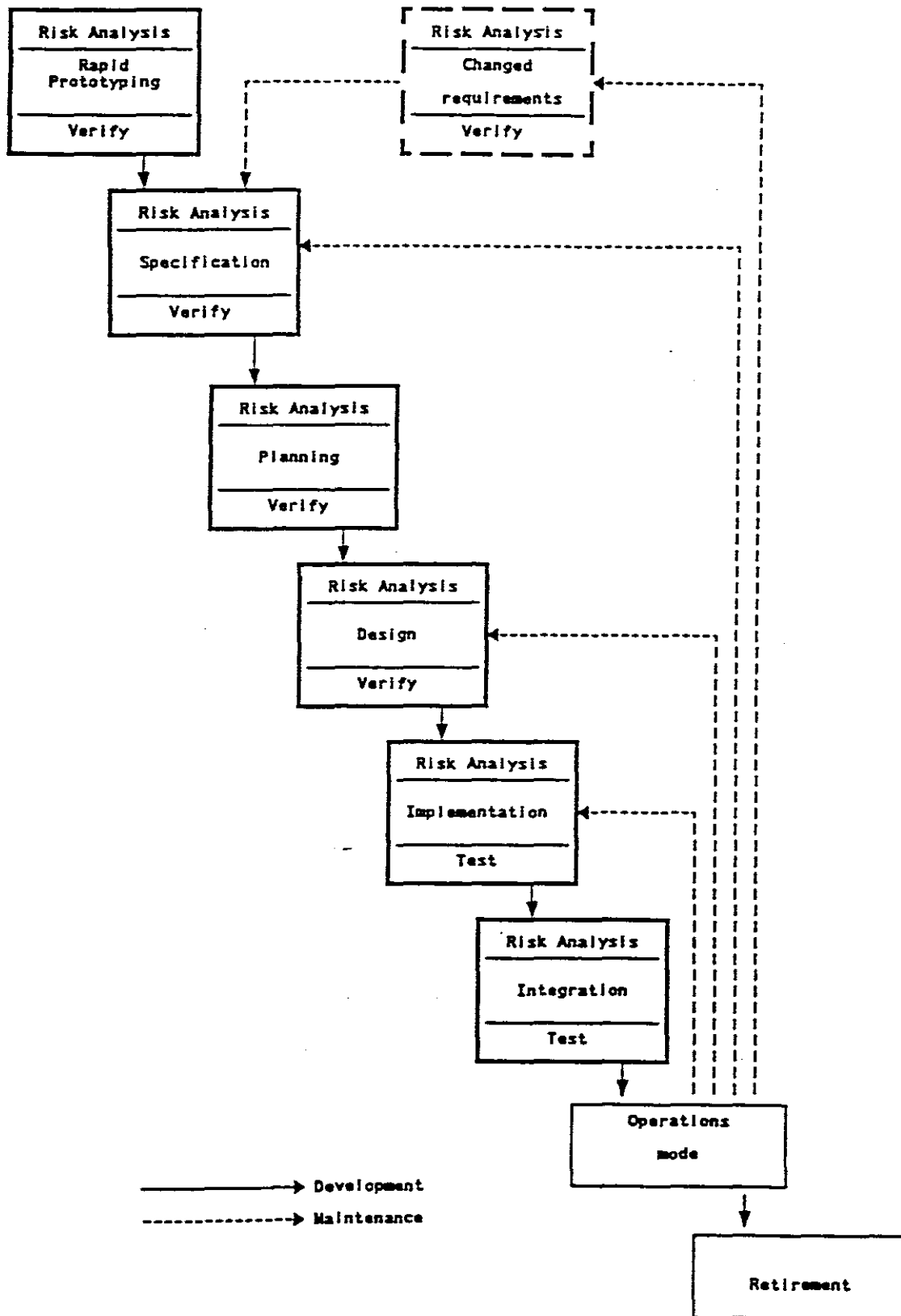


Figure 2-4: Simplistic version of spiral model (Schach, 1993:62)

The full Spiral Model is shown in Figure 2-5 where the radial dimension indicates cumulative cost to date and the angular dimension indicates progress through the spiral. Each cycle represents a phase which begins by determining the objectives, alternatives to achieve those objectives and the constraints imposed on the alternatives. The next quadrant represents the evaluation of alternatives, and the identification and resolution of the risks. This is followed by the development and verification of the product (via prototypes, if possible), and the planning of the next phase (if required to proceed).

Although this process model supports the re-use of existing software, it is clear that more than only systems analysis and design needs to be done. It is a process model that requires a variety of skilled people to perform all the necessary tasks and would as a result of this, take a long time to implement a system. It is therefore only suitable for the development of very large systems where the project can be terminated as soon as the risk analysis leads to such a conclusion.

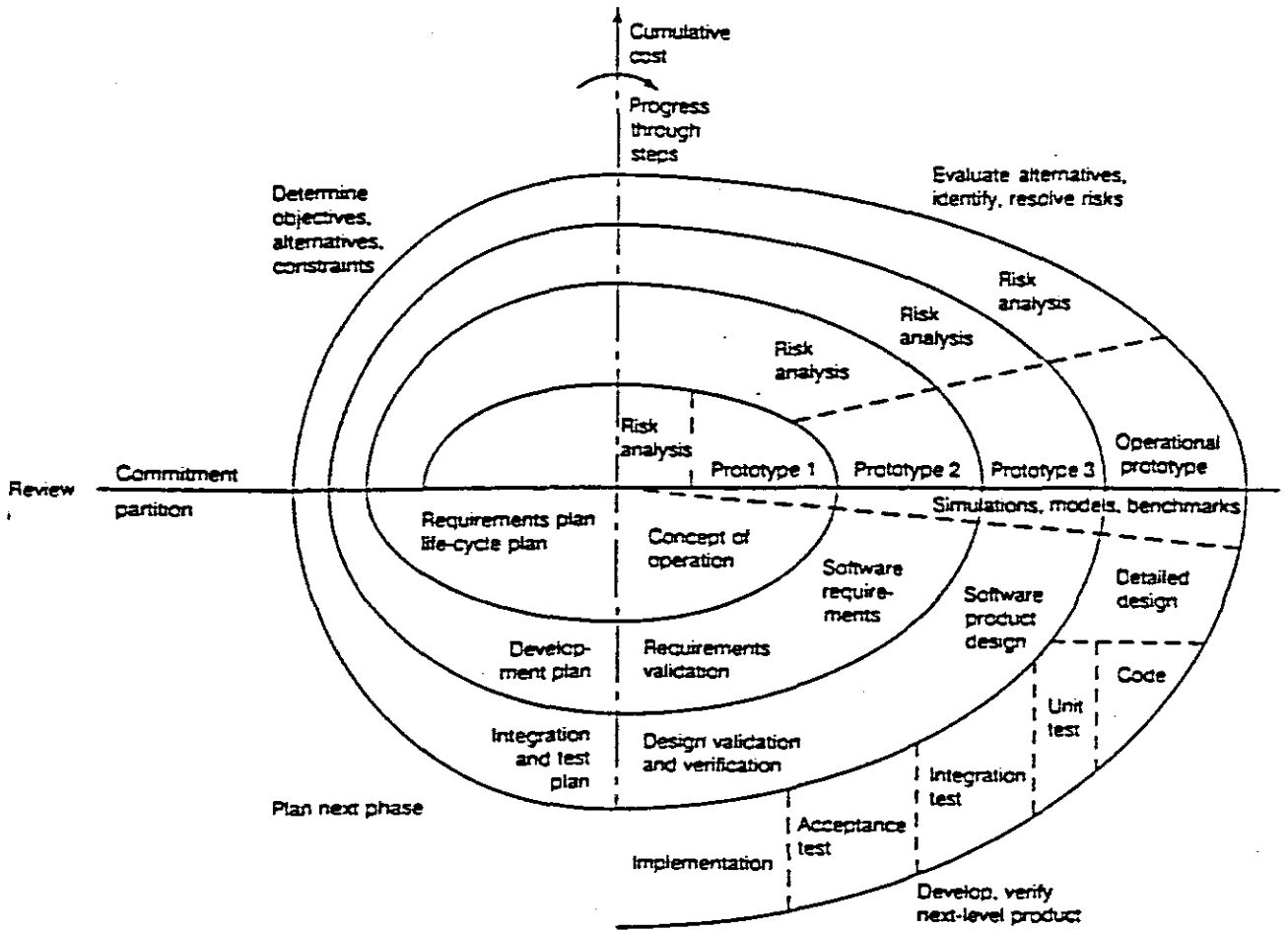


Figure 2-5: Full spiral model (Schach, 1993:64)

2.3 COMPARISON OF PROCESS MODELS

Six different process models have been described and examined.

The Classic Life Cycle, the Modern Life Cycle and the Waterfall Model are very similar. Most system developers use some or the other version of these three models because they are easy to understand. Each phase is clearly defined (especially when methodologies based on these models are used) and it is also clear that the output of the one phase serves as the input to the next phase. If something goes wrong in one phase, subsequent phases will be influenced by it. The problem is that a product that does not meet all the requirements of the user may, in other words, not be what the user really wants.

The Rapid Prototyping Model was developed to overcome the problem, namely that the delivered product may not be what the user really needs. The problem with this model, however, is that documentation can easily be neglected and it could become difficult to plan and control the project as it can lead to a repetitive trial-and-error process without knowing when the final version will be accepted. This model can be used as part of the phase where requirements are defined and finalised in the Waterfall and similar models.

The Incremental Model is useful if the total project can be broken down

into distinct deliverables or units. Each unit can then be developed using anyone of the more classic models. It is, however, important to keep in mind that the unit forms part of a broader system; therefore, interfaces must be planned meticulously.

The Spiral Model is basically an expansion of each phase of the Waterfall Model. More tasks were added to the existing phases. This proves to be a sound approach for very large systems only, because it is, judging by the tasks to be executed, by far the most expensive model.

2.4 CONCLUSIONS

The way in which systems development is taught depends on the environment in which it is taught. In industry, the prospective systems analyst is in the position to experience the use of process models in the real world. If the organization he works for, uses, for example, one of the classic approaches, he can be taught the tasks or activities of each phase just before he is required to perform those tasks.

This is not the case with a student at a tertiary institution where the focus is more on the theory. The student should be exposed to more than one model which he can apply once he starts working.

The fact that people are taught the process models in different ways,

constitutes a major problem. For the last 30 to 40 years almost every organization tended to implement its own version of one of the process models. This makes it almost impossible for the trainer to teach a life cycle that can be applied to the letter from day one of the prospective systems analyst's working career.

It is therefore necessary to establish from time to time the common parts of life cycles that are used by various leading organizations in the region being served by the training institution.

There are many systems development tools and techniques being used in the various phases of any SDLC. Some of these tools and techniques are discussed in the next chapter.

CHAPTER 3: SYSTEMS DEVELOPMENT TOOLS, TECHNIQUES AND METHODOLOGIES

3.1 INTRODUCTION

During the systems development process the systems analyst has to record an enormous amount of facts to enable him to analyze the current system and its problems, to identify the requirements for the new system and to design and implement a new system that will be accepted by the user. The specifications for the system and its programs should also be drawn up in such a way that it is clearly understood by the applications, software and database programmers.

It happens quite often that systems analysts try to remember important facts without properly documenting it. The reason for this is that they feel that the documentation of facts during systems analysis and design takes up valuable time. In short, some feel that it is a waste of time.

One has to appreciate their argument because the process of documenting facts can be very time-consuming. The problem, however, is that it is impossible for a person to remember all the facts that are required to develop a successful system. Some 'quicker ways' of documenting systems are therefore required.

Fortunately, during the early 1970's and 1980's, many systems experts

such as Yourdon, De Marco, Gane, Sarson and James Martin, identified and defined this problem. They developed some systems development tools, techniques and methodologies (SDTTM) in an attempt to minimise the time used to document systems.

Another problem that led to the software crisis was that the process of analysis and design was full of ambiguity. These SDTTM's had to cater for this problem as well.

It was also important that systems analysts had to use these SDTTM's within the systems development life cycle (SDLC) and not in place of the SDLC. These SDTTM's had to be applied in one or more phases of the SDLC.

There are many different SDTTM's being used by various organizations today. New and more modern SDTTM's are frequently developed by researchers, consultants and academicians. Some of these SDTTM's are obviously more popular than others, but the concept of using it to document systems (and even develop systems) has become widely accepted. In this chapter some of the more popular SDTTM's (according to the literature study) are described. It is, however, compulsory to establish which of these SDTTM's (and others not found in the literature) are mostly used in industry being served by the institution training systems analysts.

For the purpose of this chapter a systems development tool is something that is used to automate some of the activities that are involved in systems development (such as computer-aided software engineering), or something used to support a given activity (such as an editor to document specifications). A technique is "an approach that applies specific tools and rules to complete one or more phases of the SDLC" (Whitten, 1994:142). A methodology is "a comprehensive and detailed version of an entire SDLC that incorporates (1) step-by-step tasks for each phase, (2) individual and group roles to be played in each task, (3) deliverables and quality standards for each task and (4) development techniques to be used for each task" (Whitten, 1994:143).

3.2 SYSTEMS DEVELOPMENT TOOLS AND TECHNIQUES

Systems development tools and techniques (in the rest of this section referred to as tools) range from very simple 'pencil and paper' diagrams, charts or tables to very complex automated programs and hardware that generate specific products such as diagrams, charts, tables, reports, etc. Each tool has its specific objective and place in the SDLC. It is not necessary to use all tools to develop systems. It depends on the system itself, the size, the complexity, the technology to be used, etc. It is therefore compulsory to plan the use of the appropriate tools properly. A tool cannot be used just for the sake of using it - it must be applied in a

beneficiary way.

It is very important for the prospective systems analyst to know how a specific tool fits into the total process of systems development and why it is used. It is also necessary to be able to validate the correctness of each tool; therefore, the correct user skills for each tool should be acquired. Too many systems analysts depend on a very sophisticated CASE tool to do the work for them. It must be borne in mind that systems analyst do not always perform analysis and/or design in a computer-supported environment. Most systems analysts must be able to draw some of the diagrams, tables, etc. while interviewing system users. This chapter, therefore, concentrates only on those tools for which a CASE tool or hardware and software are not required.

The various tools investigated are briefly examined and discussed. Tables, diagrams and figures are again abundantly used to alleviate long narrative descriptions. In some of the discussions, case studies and possible solutions are used to clarify the use of the appropriate tool in more detail.

It is also necessary to mention that the methods used in the illustrations are the methods taught to the students at the institution where the researcher lectures. These methods were found to be the most effective in the application of the tools in students' projects.

3.2.1 DATA FLOW DIAGRAMS

Data flow diagrams are primarily used to model the flow of data into a system, within a system and from a system to its outside environment. It is also used as a communication tool to clarify ambiguities about a system's processes.

According to Whitten (1994: 350) a data flow diagram (DFD) is "a process modelling tool that depicts the flow of data through a system and the work or processing performed by that system. Synonyms include bubble chart, transformation graph, and process model".

There are various symbolic notations suggested by different authors and experts for drawing the DFD. The most popular and widely used notations are the Gane-Sarson and the De Marco-Yourdon symbol sets. The only difference between these two methods is the symbol set, but the approach is very much the same.

A completely different and more structured method of drawing DFD's was developed by Learmonth and Burchett Management Services (LBMS) in 1981 (Eva, 1992:3). This approach was accepted by the Government in the UK to be incorporated in the

Structured Systems Analysis and Design Methodology (SSADM) used by all its departments as the de facto standard. This method enforces certain strict rules that should be adhered to, as well as steps which must be executed in a specific sequence to ensure the correctness and acceptability of the eventual DFD.

The general rules for drawing DFD's are in all three cases the same. These rules are:

- * A process must have at least one input and at least one output.
- * A direct data flow between an external entity and a data store is invalid.
- * A direct data flow between two data stores is invalid.
- * It is not necessary to record a data flow between two external entities.

For the purpose of illustrating the differences in the three methods above in practice, a small case study is described and the different models are shown in Appendix A.

3.2.2 ENTITY RELATIONSHIP DIAGRAMS

Whereas DFD's are used to model the flow of data, entity

relationship diagrams (ERD's) are used to depict the structure of data. One can say that the process model (DFD) shows the system in motion, whilst the ERD (also known as the data model) shows the system's data structure at rest. Like the DFD, it is also used as a communication tool - but in this case, to clarify ambiguities about a system's structure.

An ERD is "a data modelling tool that depicts the associations among different categories of data within a business or information system - it does not imply how the data is implemented, created, modified, used or deleted. Synonyms include entity model or entity diagram, or entity relationship attribute diagram" (Whitten, 1994:307). SSADM refers to ERD's as Logical Data Structures (LDS).

In the case of the ERD or LDS, there are also various symbol sets to depict the system's data structure. To illustrate the use of these various symbol sets and approaches, the various data structure models of the system described in Appendix A are also shown.

3.2.3 DECISION TREES AND TABLES

Algorithms involving multiple nested decisions are very difficult to

describe when using standard English or any other language.

According to Whitten (1994:437) some of the problems that can be experienced when using for example English are -

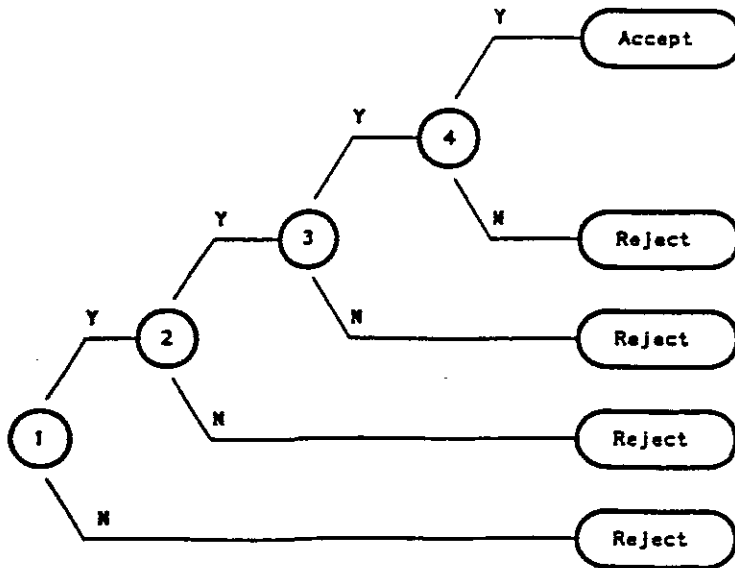
- * Statements tend to have an excessive or confusing scope,
- * Compound sentences cause interpretation problems,
- * Multiple definitions cause misunderstanding,
- * Undefined adjectives confuse readers and
- * Conditional instructions confuse readers. This problem is further complicated by combinations of conditions.

A decision table, consisting of a condition stub at the upper left, a condition entry at the upper right, an action stub at the lower left and an action entry at the lower right, is a tabular form of presenting the specification with its conditions and corresponding actions. A decision tree is a graphic representation of the specification.

To illustrate the use of these tools, a decision tree and a decision table are shown in Figures 3-1 and 3-2 for the following problem:

A basket ball coach is looking for full-time male students who are at least 2,1 meters tall and weigh at least 95 kilograms.

DECISION TREE



DECISIONS:

1. Is the student male?
2. Is it a full-time student?
3. Is the student taller than 2.0 meters?
4. Does the student weigh more than 94 kg?

Figure 3 - 1: Decision tree (Davis, 1983:493)

DECISION TABLE:

1. Identify the conditions and values

Data attributes or conditions	Possible values
Is the student male?	Y = Yes N = No
Is it a full-time student?	Y = Yes N = No
Is the student taller than 2.0 meters?	Y = Yes N = No
Does the student weigh more than 94 kg?	Y = Yes N = No

2. Determine the maximum number of condition entries or rules

Condition 1 offers 2 values	2
Condition 2 offers 2 values	x 2
Condition 3 offers 2 values	x 2
Condition 4 offers 2 values	x 2
Number of rules	16

3. Identify the possible actions

1. Accept student for basket ball
2. Reject student

4. Table all possible rules and resulting actions

Condition stub:	Condition entry/rules															
Student male?	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Full-time?	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
> 2.0 meters?	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
> 94 kg?	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N
Action stub:	Action entry															
Accept student	X															
Reject student	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure 3 - 2: Decision table (Adapted from Davis, 1983:406)

5. Simplify the table by eliminating and consolidating certain rules

Condition stub:	Condition entry/rules			
Student male?	Y	N		
Full-time?	Y		N	
> 2.0 meters?	Y			N
> 94 kg?	Y			N
Action stub:	Action entry			
Accept student	X			
Reject student		X	X	X

Figure 3 - 2 (continued)

3.2.4 INSPECTIONS AND WALKTHROUGHS

If the systems development process is to be effectively controlled, some or other management and technical review should occur at the end of each step in the process. Management reviews focus on aspects such as costs, schedule and return on investment, but lacks proper technical review. An inspection or walkthrough is a formal review of the exit criteria which is conducted by technical personnel. The intent of such an inspection or walkthrough is to determine the technical accuracy of the documentation produced during that step. Technical accuracy should be defined in the organization's standards.

A walkthrough can be regarded as an informal inspection. It usually takes place before the documentation is presented to an inspection team or management for official inspection. There are, however, major differences.

Firstly, for the inspection a formal team should consist of a moderator (chairperson), an author (scribe) and at least two inspectors (technical experts). For the walkthrough it is only necessary to consult several colleagues (usually peer analysts and/or programmers) to preview material for identifying "rough spots" mainly to save the analyst considerable embarrassment.

In the second place, the walkthrough (if formal) consists of only two steps, namely preparation and a group analysis. Davis (1983:258) describes the inspection as a six-step process consisting of:

- * Planning

The permanent personnel on the team, the moderator and the author select in consultation with management, the other members of the team. The next step is to distribute all relevant documentation to these elected members, and then schedule the series of meetings.

- * Overview

This is the only optional step of the whole process. If the material to be inspected entails very complex technical concepts, it is the moderator's duty to present a brief technical overview of the project and the documentation. Care must be taken that this step does not end up in a question-answer session. The presentation must stick to the facts. The members may tender questions during meetings after this session so that they have enough time to review, research and understand the documentation.

- * Preparation

This step requires of each member of the team to prepare himself properly for the meetings to follow. It is possible that members might confer with one another before meetings take place. This can save time, but it can also constitute a problem in that it might cause one member to convince another member to agree with his statement even before the problem is properly discussed at the meeting. Participants should be aware of potential bias and should avoid nonessential contact.

- * Inspection session

The objective of the inspection session is to find errors. The documentation should be scrutinized by the team in detail. A well-proven approach is that one of the members read the paragraphs of the documentation aloud. After each paragraph has been read, the team summarises the paragraph to ensure that no interpretation problems exist. All the errors found in the documentation should be logged officially on a standard error log sheet (refer Figure 3-3 for an example).

INSPECTION ERROR LOG					
PROJECT:					
MODULE/COMPONENT:					
INSPECTION LEVEL:					
SESSION DATE:					
TARGET DATE:					
MODERATOR:					
AUTHOR:					
INSPECTOR:					

	No Error description	Severity	Est	Act	Date Check
			time	time	

1					
2					
3					
4					
5					
6					
7					
8					
9					
INSPECTION TIME USED BY:					
	Moderator	-			
	Author	-			
	Inspector/s	-			
SIGNED BY:					
	Moderator			
	Author			
	Inspector/s			
				
				
DATE INSPECTION COMPLETED:					

Figure 3-3 : Inspection error log (Adapted from Davis, 1983:261)

It is very important to note that the inspectors are not allowed to correct the errors; merely to identify and report the errors. It is the duty of the analyst who produced the documentation to correct the errors.

There could be, of course, more than one inspection session. It all depends on the amount of work to be done. It is important that the work should be done properly and completely.

- * Rework

After the inspection sessions, the moderator and the author meet to discuss and estimate the rework time for each error recorded on the error log. As each error is corrected, the author should note the actual rework time. The purpose of this is to build a database of estimated and actual rework times with the view to improving the estimation process.

- * Follow-up

The moderator and author ensures that all the rework have been done and that it is correct.

3.2.5 FACT-FINDING TECHNIQUES

During the process of systems analysis and design, a systems analyst collects facts. This is one of the most important techniques that should be acquired.

Whitten (1994:792) defines fact-finding as "the formal process of using research, interviews, questionnaires, sampling and other techniques to collect information about systems, requirements and preferences. It is also called information gathering or data collection".

Whitten (1994:793 - 806) cites the most common fact-finding techniques as:

- * Study existing documentation, forms and files.

It is necessary for the systems analyst to collect facts about a system from existing documentation. It always gives a very good background about the history that led up to the systems that had been developed and implemented. At the same time the analyst also familiarises himself with the terminology of the system. The most important advantage of collecting facts from existing documentation is that the analyst can do research in his own time. It is usually not

necessary to involve other people. It therefore saves time and money.

The type of documentation needed to study are organization charts (to ensure that the most appropriate people are involved when meetings are arranged), minutes of meetings, interoffice memoranda, systems analyst's reports, information systems requests, systems, program and operations specifications and manuals, etc.

* Research and site visits.

This type of fact-collecting is necessary when a system which has never been developed before needs to be designed. It is possible to consult other companies having similar problems or study various computer trade journals.

* Observation of the work environment.

Observation is a fact-finding technique wherein the systems analyst either participates in or watches a person perform activities to learn about the system. When this technique is used, care should be taken that workers do not feel uncomfortable when being observed.

- * Questionnaires

Questionnaires are the best fact-finding technique to be used when information needs to be collected from a large number of users of the same system while maintaining uniform responses. A questionnaire is a special-purpose document with specific type of questions to achieve a specific objective. One of the major advantages of questionnaires is that they can be answered relatively quickly, and that the results can likewise be rapidly tabulated and analyzed. The problem is that it is not easy to prepare a questionnaire and that the response is sometimes very low.

- * Interviews

Interviews are generally recognized as the most popular and, provided that it is conducted in the correct way, the most effective way of collecting facts. Interviews give the analyst the opportunity to motivate the interviewee to respond freely and openly to questions. It also allows the analyst to probe for more feedback from the interviewee. If the interviewee does not understand the analyst's question, the analyst can reword the question without wasting much time. Another great advantage is that the analyst can confirm his understanding of the user's answer during the interview - in other words, while the user and the analyst are in the right

frame of mind. The main disadvantage of interviews is that it is time-consuming and therefore costly.

* Joint Application Design (JAD)

One shortfall of interviews is that an analyst sometimes has to consult more than one user at different times about the same matter. This usually leads to conflicting answers to the analyst's questions, which, in turn, causes the analyst to have a distorted idea about certain aspects of the system.

To overcome this problem, group work sessions, in which all the system's users (including management) and the systems development staff participate, are used. JAD is an example of such a group work session. JAD sessions which are highly structured usually take place over an extended period of time. During these sessions specific problems and requirements are discussed by selected participants in order to come up with a solution. The main advantages of JAD is that it reduces the amount of time it takes to get a system implemented eventually, it improves the quality of systems and it improves the relationships among system users, developers and management. However, due to the number of people involved at the same time, JAD is expensive. On the other hand, much time can be saved eventually if a matter gets solved during one session as compared to an

analyst having to conduct several interviews to get the same result.

3.2.6 FEASIBILITY ANALYSIS

An organization needs to know if a particular or proposed system is worth implementing. Systems are used to improve the business; it is therefore imperative to determine whether a system will be practical and beneficial to the organization. Feasibility "is the measure of how beneficial or practical the development of an information system will be to an organization. The feasibility study is the process by which feasibility is measured" (Whitten, 1994:812).

Whitten (1994: 816) groups feasibility into the following four categories:

- * **Operational feasibility:**

This is the measure of how well the proposed solution of the problem will work in the organization. It is very difficult to quantify and is therefore merely a list or report of various people's opinions.

- * **Technical feasibility:**

This is the measure of how well technically the solution will work. Some questions to answer here, are: Do users understand how to operate the system? Is there an experienced technical support person available?

- * **Schedule feasibility:**

This measures the time at the developer's disposal to complete the project. Is it reasonable?

- * **Economic feasibility:**

The bottom line of any project - is this going to be a good investment? This measures the cost-effectiveness of the solution. All the costs (development and operational) as well as the benefits (tangible and intangible) should be recorded and considered when a cost-benefit analysis (see 3.2.7) is done.

During the systems development process there are many points (called checkpoints) at which a feasibility analysis should be done. As one progresses with the development of a system, more and more information which has an effect on the feasibility of a system, becomes available. It is therefore very difficult to do a final feasibility analysis at the start of a project.

3.2.7 COST BENEFIT ANALYSIS

The cost-benefit analysis is generally used to compare costs and benefits in order to determine whether a system solution is economically feasible. Present and future costs (such as personnel costs, computer usage; training, hardware and software, etc.) and benefits (fewer processing errors, increased throughput, increased sales, improved customer goodwill, etc.) should be adjusted at a reasonable discount rate to present value so as to determine the economic feasibility. Three of the most popular techniques to determine economic feasibility are, the payback period (the period it will take the system benefits to break even with the amount invested), the net present value (the difference between the lifetime costs and benefits) and the return on investment (the profit on the investment). Figure 3-4 illustrates the method taught to students for using these three cost-benefit analysis techniques. From the table in this figure the results can be calculated as follows:

- * The Net Present Value (NPV) after 6 years can be read directly from the table, i.e. in column J. The figure is R8570.
- * The Return On Investment (ROI) after 6 years is equal to the NPV divided by the accumulated costs at present value, which is $8570 / 335940 = 2,55\%$

- * The payback period is after 5 years plus a portion of the 6th year. According to the table the NPV after 5 years was 26920- (column J). The gain during year 6 was 35490 (column K). The portion of the year it took to reach the break even point, is $26920 / 35490 = 0,76$. Therefore the payback period is 5,76 years.

A new production scheduling information system for the ABC company could be developed at a cost of R175 000,00. The estimated net operating costs and estimated net benefits over six years of operation would be as follows:

0	175000	0
1	50000	75000
2	45000	80000
3	40000	85000
4	35000	85000
5	30000	90000
6	25000	95000

Calculate the Net Present Value and the Return on Investment after 6 years, as well as the Breakeven point. Discount the cost and benefits at 14% interest rate.

A	B	C	D	E	F	G	H	I	J	K
0	1	1	175000	175000	175000	0	0	0	-175000	0
1	1.12	0.893	50000	44650	219650	75000	66975	66975	-152675	22325.
2	1.2544	0.797	45000	35865	255515	80000	63760	130735	-124780	27895
3	1.4049	0.712	40000	28480	283995	85000	60520	191255	-92740	32040
4	1.5735	0.636	35000	22260	306255	85000	54060	245315	-60940	31800
5	1.7623	0.567	30000	17010	323265	90000	51030	296345	-26920	34020
6	1.9738	0.507	25000	12675	335940	95000	48165	344510	8570	35490

{Column Headings are as follows:

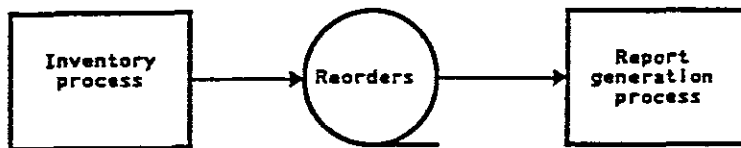
A = Year; B = Interest rate; C = Discount rate; D = Cost; E = Cost @ present value; F = Accumulated cost @ present value; G = Benefit; H = Benefit @ present value; I = Accumulated Benefit @ present value; J = Net Present Value; K = Improvement during year}

Figure 3-4: Cost-Benefit Analysis (own example)

3.2.8 SYSTEM FLOWCHARTS

System flowcharts are traditionally used to describe a physical system. A physical system contains many different components such as programs, files, databases, forms and procedures. Each component of the system should be depicted by a specific symbol. An example of a system flowchart is depicted in Figure 3-5.

A high-level system flow diagram of an inventory application



An exploded version of the above system flow diagram

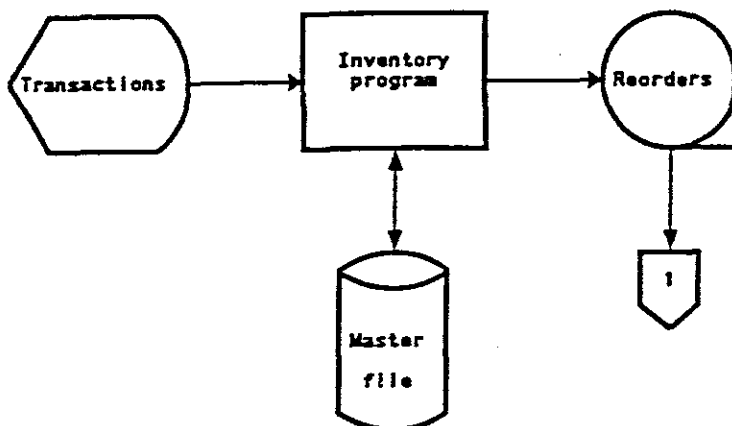


Figure 3 - 5: System flowchart (Davis, 1983:308)

3.2.9 PSEUDOCODE

Pseudocode is a tool to describe the logic of a procedure or an algorithm prior to real program coding. 'Pseudo' means similar to; thus pseudocode is similar to real code. Most of the various pseudocode structures are based on real program code such as COBOL, FORTRAN, BASIC, PASCAL, etc. The main purpose of using pseudocode is to describe in detail how the procedure or algorithm should be coded without concerning oneself about the syntax of a specific programming language. That means that pseudocode is used to describe the executable code in a form that a programmer can easily translate. It is therefore mostly used in program specifications. The following problem is used to illustrate the use of pseudocode:

The program has to read a file containing many student records. Each record consists of a student number, a student name, six occurrences of a subject taken by the student and its corresponding mark obtained in the examination. The purpose of the program is to calculate the student's average mark obtained for all his subjects.

The pseudocode for the above problem should be as follows:

Open input StudentFile.

Read StudentFile.

For each student record read:

 Compute AverageMark =

 (Mark1 + Mark2 + Mark3 + Mark4 + Mark5
 + Mark6) / 6.

 Display Student Name AverageMark (on screen).

At end: Close StudentFile.

If the programmer has to translate the above pseudocode into COBOL, it will be as follows:

A01-MainControl.

 Open input StudentFile.

 Read StudentFile at end

 move all 9 to StudNo.

 Perform B01-Process until StudNo = all 9.

 Close StudentFile.

 Stop run.

B01-Process.

 Compute AverageMark =

 (StudMark1 + StudMark2 + StudMark3 +
 StudMark4 + StudMark5 + StudMark6) / 6.

 Display StudName " " AverageMark.

Read StudentFile at end

move all 9 to StudNo.

3.2.10 CROSS-REFERENCE: DATA AND PROCESS MODEL

The Cross-Reference Listing (CRL) is used, especially in the SSADM method, to ensure that all data items required in a system are accommodated both on the process and the data model. If any discrepancies occur, it should be formally noted and solved before the new system can be designed and implemented.

To illustrate the use of this tool, refer to the CRL for the system described in Appendix A.

3.2.11 LOGICALIZATION

According to Eva (1992:3) Learmonth and Burchett Management Services (LBMS) invented a methodology called Structured Systems Analysis and Design (SSADM). In this methodology the technique, logicalization, is prescribed. Logicalization refers to the removal of all physical constraints from a set of DFD's. DFD is a powerful tool to analyze the processing and the data flow, but it is sometimes difficult to gain a true view of the functionality of a system. The reason for this is that the DFD sometimes has to show a lot of physical aspects so that the user can identify himself

with the system.

In order to design a system, it is important to view a system from a point of what the system must do irrespective of how it is done.

Logically, all information should be stored once, and all users should have access to this information at all times.

The main purposes of logicalization is:

- * To identify further problems, such as duplication of processes and information, in the current system.
- * To understand the basic functionality of the system.
- * To establish the system scope more precisely.
- * To force a switch in concentration away from the 'real world'.
- * To act as a basis for proper system specification.

The steps for logicalization are as follows:

- * Logicalize the data stores.

Each data store on the process model represents either data stored in a permanent base of data or transient data that is held for a short time before being used by a process and then deleted. Each of the main data stores should be related to the entities on the data model as this is the structure of the permanent data. It is therefore necessary to replace the

data stores with data model entity names and remove the transient data stores. This is illustrated in Figure 3-6.

- * Logicalize bottom-level processes and data flows.

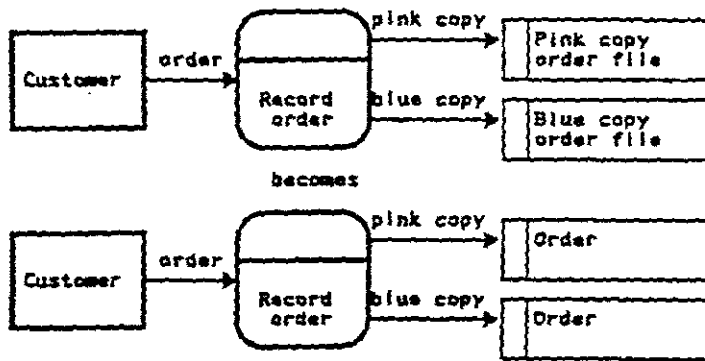
A logical process is one that transforms or uses data required for the business. A logical process also indicates only what is to be done, and not how it should be implemented. Therefore, processes that cannot be automated or processes that only reorganize data should be removed. All the activities for this step are illustrated in Figure 3-7.

- * Group bottom-level processes to form top-level DFD.

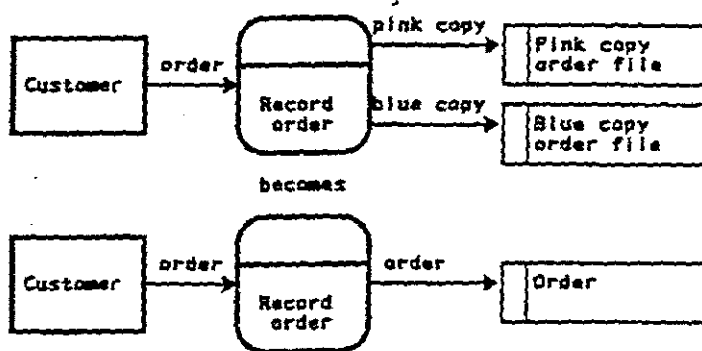
The top-level process model should show only the logical areas of a system. Useful guidelines for this step are to minimise the data flows between top-level processes and to group processes that use the same data together. Care should be taken not to change the top process model to such an extent that it becomes merely a process model copy of the organization chart. In the majority of cases it is not necessary to do this step as most of the requirements for logicalization were attended to in the previous steps.

Each data store stores permanent or transient data. Data stores should also be related to data entities which should indicate the permanent data structure. It is therefore necessary to:

1. Replace main data stores with data model entities



2. Remove duplicated data stores



3. Remove transient data stores

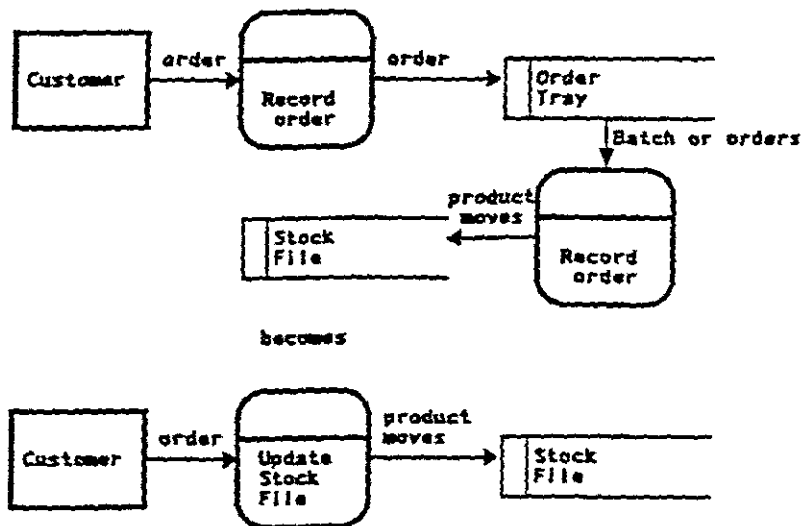
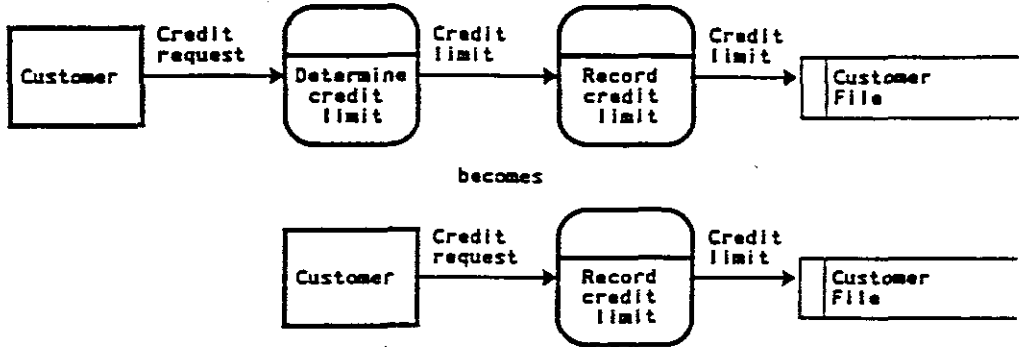


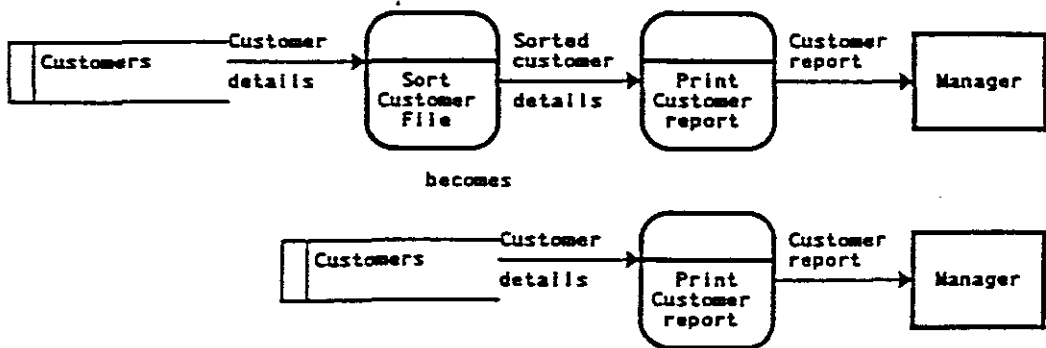
Figure 3 - 6: Logicalize data stores (own example)

A logical process is one that transforms or uses data because the business requires it to do so, independantly of how it is implemented. It is therefore necessary to:

1. Remove processes that cannot be automated



2. Remove processes that only re-organizes data



3. Merge processes that are joined by data flows only

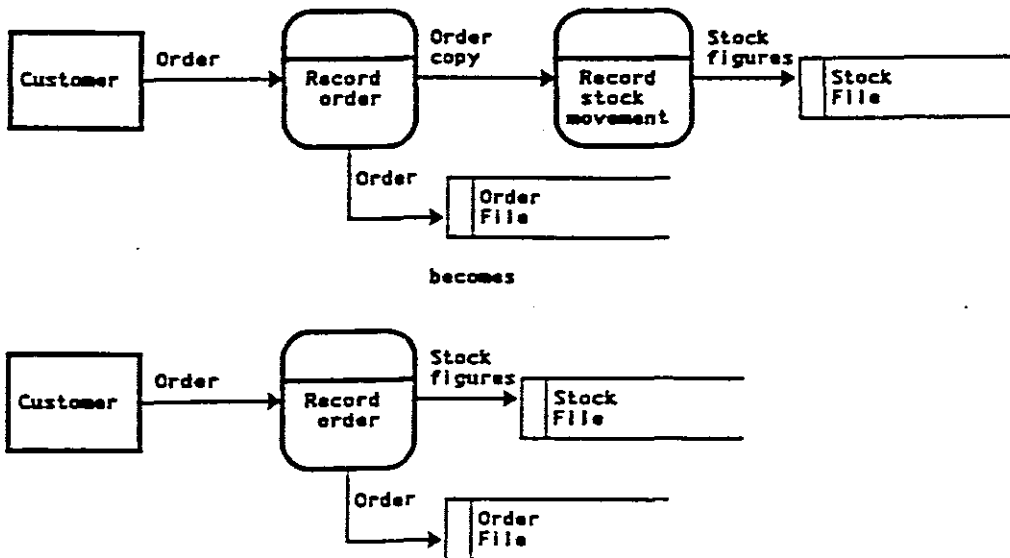
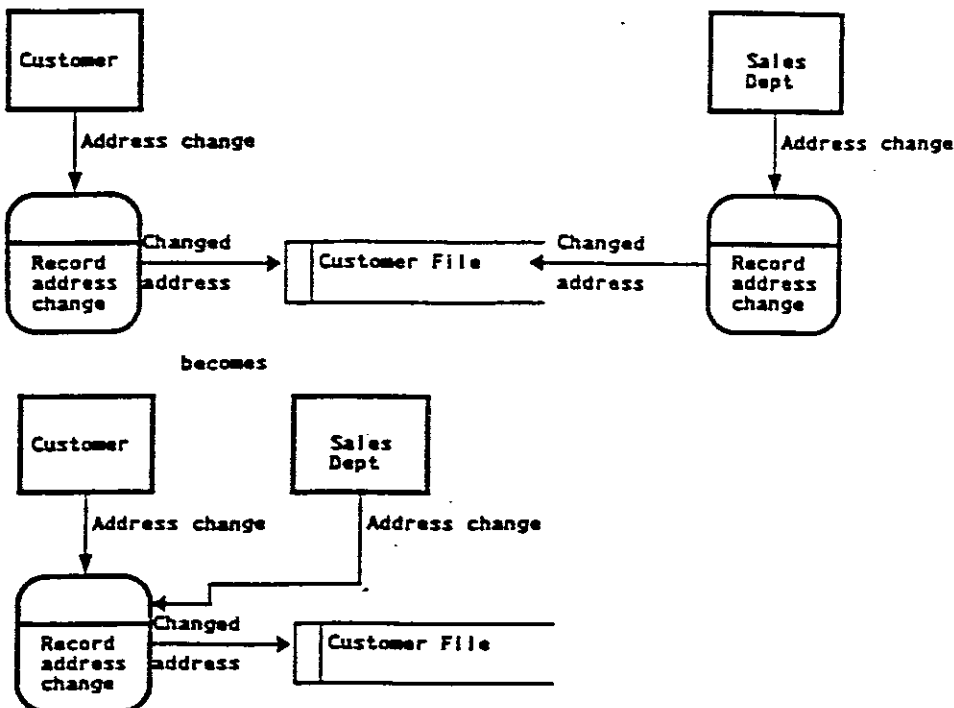
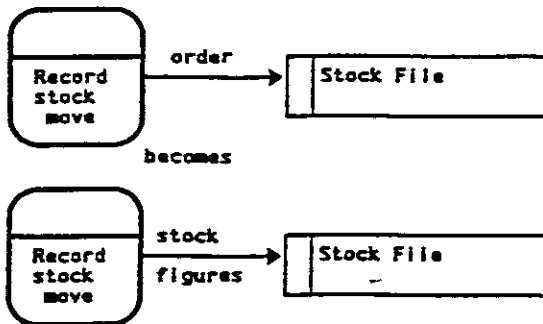


Figure 3 - 7: Logicalize bottom-level processes and data flows (own example)

4. Merge processes performing the same function



5. Minimise data flow content



6. Obtain data only when needed

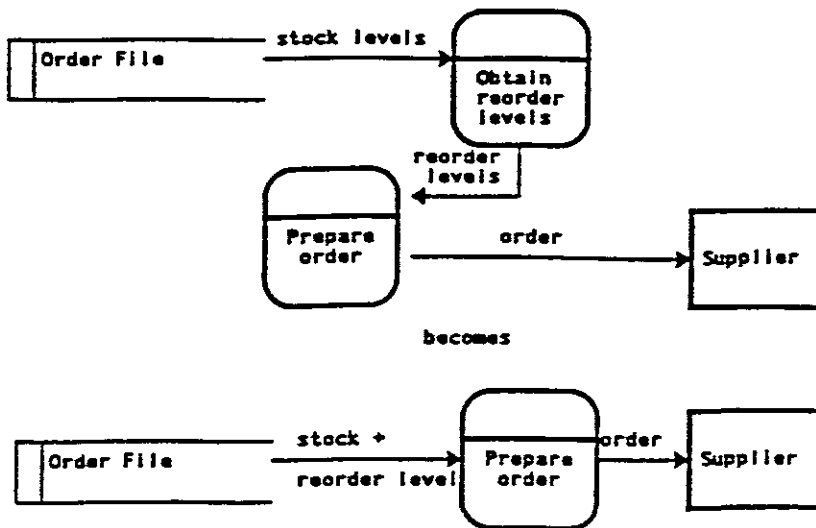


Figure 3 - 7 (continued)

3.2.12 LOCATION CONNECTIVITY DIAGRAM

Many companies today are dependent on proper networks for the effective running of business. It is important for the systems analyst to design an understandable network model. It is therefore necessary to have a tool to model the system geography independent of any possible implementation. According to Whitten (1994: 409) a location connectivity diagram (LCD) "is a network modelling tool that depicts the shape of a system in terms of its user, process and data locations and the necessary interconnections between those locations".

There are basically four types of locations that need to be depicted in a LCD. These types are:

- * Location with sublocations:

A building may have several floors, and a floor may have several rooms. To indicate this situation, a circle superimposed with a dotted plus sign is used.

- * Mobile locations:

Travelling agents of the business use the system while moving from one location to another. This is shown by a pair of concentric circles.

- * External locations:

Some companies link with external systems to reduce

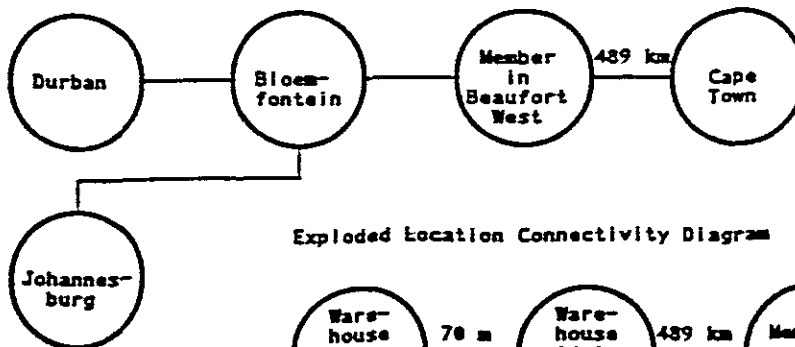
response time. This type of location is indicated with a circle superimposed with a dotted 'X'.

* Primitive locations:

This type of location is internal to the business and can not be divided into sublocations. This location is represented by a circle.

The connection between locations is represented by a line connecting the relevant locations. An LCD is illustrated in Figure 3-8.

System Location Connectivity Diagram



Exploded Location Connectivity Diagram

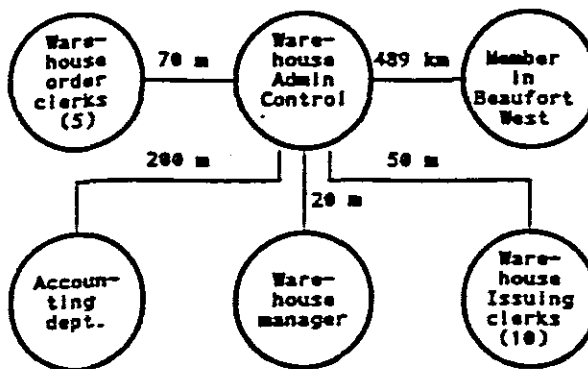


Figure 3 - 8: Location connectivity diagram (Adapted from Whitten, 1994:418)

3.2.13 DATA ANALYSIS

Once the initial data model has been finalised and balanced with the process model (using cross-referencing as described in 3.2.10), the data still needs to be normalized. This means that the data attributes must be grouped in such a way that it eliminates data redundancy and removes update anomalies.

Data analysis "is a procedure that prepares a data model for implementation as a non-redundant, flexible and adaptable database. Data analysis typically uses a procedure called normalization to simplify entities, eliminate redundancy and to build flexibility and adaptability into the data model. Normalization is a data analysis method that organizes data attributes such that they are grouped together to form stable, flexible and adaptive entities" (Whitten, 1994:507).

Up to the 1970's computers were mostly used to automate manual procedures. This resulted in a very ineffective way of storing data. This was recognized by, among other, Edgar Codd of IBM, who, based on the mathematical set theory, published material on Relational Data Analysis (RDA). Eva (1992: 334) defines RDA as "a technique for analysing data to its logical groupings. A set of 'raw', i.e. unstructured data, is reduced through a set of

transformations (normalization) to basic data groups in which each item is grouped with its sole and full determinant."

For the purpose of the discussion that follows, it is necessary to define relation. A relation is "a group of data items under its logical determinant. It is represented as a table of entries, each row being an occurrence of that relation. In RDA each relation is regarded as an entity for the purpose of validating the logical data model" (Eva, 1992:334).

Normalization consists of the following three steps:

- * First Normal Form (1NF)

A relation is in 1NF if it does not contain repeating groups.

- * Second Normal Form (2NF)

A relation is in 2NF if it is in 1NF and no non-key attribute is dependent upon only part of the key.

- * Third Normal Form (3NF)

A relation is in 3NF if, in addition to being in 2NF, there are

no non-key attributes which depend on other non-candidate key attributes. This is expected to be in the most logical grouping, not to be subject to update anomalies and to give the least data redundancy.

An example of normalization is shown in Figure 3-9.

3.2.14 EVENT ANALYSIS

Once the data model has been normalized, the effects of events that can influence the business need to be validated against the data model.

Event analysis is "a technique that studies the entities of a fully normalized data model to identify business events and conditions that cause data to be created, deleted or modified. A business event is something that 'happens' and that causes business data to change. Business events require processes to utilize or maintain the data. These processes may include:

- * Creating a new occurrence of an entity (sometimes called *add*)
- * Reading an occurrence of an entity (sometimes called *use* or *add*)
- * Updating an occurrence of an entity (sometimes called

change or modify)

- * Deleting or archiving and occurrence of an entity”
(Whitten, 1994:524).

An illustration of event analysis is explained and depicted in Figure 3-10.

The data on the following order must be analysed and normalized:

ORDER FORM				
ORDER NUMBER: 13546		DATE: 1995-05-29		
SUPPLIER: XYZ Company		SUPPLIER CODE: 34		
P O Box 123				
CAPE TOWN				
8000				
ITEM NUMBER	ITEM DESCRIPTION	UNIT PRICE	QUANTITY	AMOUNT
123456	Clutch pencil	4.00	25	100.00
234567	Parker pen	10.00	15	150.00
345678	Floppy disk	65.00	10	650.00
TOTAL:				900.00

1. STEP 1:

Write down the unnormalized entity as a relation in 0NF (zero normal form)

$R(\text{OrdNo, OrdDate, SuppCode, SuppName, SuppAddress, OrderEntries occurs } \theta \text{ to } n \text{ times: (ItemNo, ItemDesc, UnitPrice, Qty, Amt), OrdTotal})$

2. STEP 2:

Remove repeating groups to get relation in 1NF:

$R_0(\text{OrdNo, OrdDate, SuppCode, SuppName, SuppAddress, ItemNo, ItemDesc, UnitPrice, Qty, Amt, OrdTotal})$

3. STEP 3:

Remove partial dependencies to get to 2NF:

$R_1(\text{OrdNo, OrdDate, OrdTotal, SuppNo, SuppName, SuppAddress})$

$R_2(\text{ItemNo, ItemDesc, UnitPrice})$

$R_3(\text{OrdNo} \rightarrow \text{ItemNo, Qty, Amt})$

4. STEP 4:

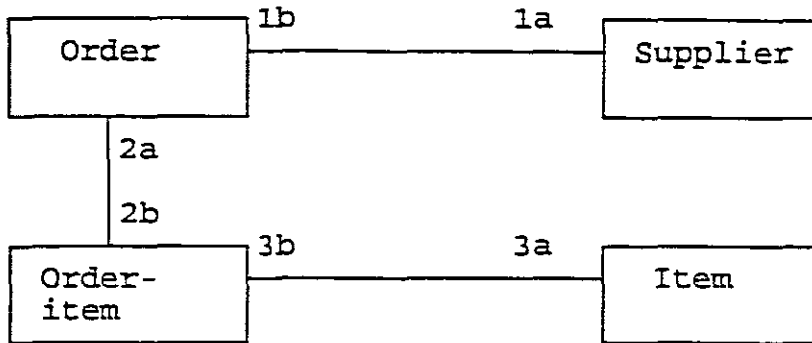
Remove attributes from relation that depend on a determinant that is not a candidate key, create a new relation with these attributes and make the determinant the primary key of this new relation to get to 3NF:

$R_{1..1}(\text{OrdNo, OrdDate, OrdTotal, SuppNo})$

$R_{1..2}(\text{SuppNo, SuppName, SuppAddress})$

Figure 3 - 9: Normalization (own example)

The normalized data model for the order relationship in figure 3-9 is as follows:



- 1a Each Supplier may have 0:m orders
- b Each Order must be for 1:1 supplier
- 2a Each order must contain 1:m order-items
- b Each OrderItem must appear on 1:1 order
- 3a Each Item may appear as 0:m order-items
- b Each OrderItem must belong to 1:1 item

The entity-event matrix for this model is as follows:
 (C = Create; D = Delete; U = Update; R = Read)

Event	Entity	Order	Supplier	Item	OrderItem
Create order		C			C
Create supplier			C		
Create item				C	
Delete order		D	U		D
Delete supplier		U	D		
Delete item		U	U	D	D
Add extra item to order		U	U		C
Item not supplied anymore		U	U	D	D
Supplier close business		D	D		D

Figure 3 - 10: Entity-Event Analysis (own example)

3.2.15 PERT CHARTS (DIAGRAMS)

It is an important part of the systems analyst's job to plan and schedule the various tasks to be performed when a new system should be developed. The systems analyst frequently has to assume the role of the project leader or coordinator. It is therefore important that the incumbent of such a role possesses the skills to plan and schedule properly. A proven technique for this purpose is the PERT (Program Evaluation and Review Technique) chart (diagram) which was developed during the late 1950's by the US Navy for controlling and scheduling the very high complex Polaris missile project (Davis, 1983:376). PERT charts are very helpful to:

- * Reduce the uncertainty about the completion dates of various tasks in a project,
- * Clearly indicate the interdependencies of the project tasks, and
- * Control the duration of project by increasing or decreasing resources where possible.

The following elements are found on a PERT chart:

- * Event.

An event is "a point in time that represents the start or completion of a task or a set of tasks" (Whitten, 1994:772).

For the purposes of this discussion, an event is indicated on

the PERT chart as a node.

- * Task.

A task is "a project activity (or set of activities)" (Whitten, 1994:772).

- * Optimistic Time (OT).

The optimistic time refers to the minimum time it takes to complete a task.

- * Pessimistic Time (PT).

The pessimistic time refers to the maximum time it takes to complete a task.

- * Most Likely Time (MLT).

The most likely time refers to the time it usually takes to complete a task.

- * Slack Time (ST).

It is possible to start late or exceed the expected duration of tasks not on the Critical Path without affecting the project's estimated completion time. This property is called slack.

- * Earliest Completion Time (ECT).

The ECT represents the earliest time (in terms of the event) that a task can be completed by and when the next task can be started.

- * Latest Completion Time (LCT).

The LCT represents the latest time (in terms of the event) that a task can be completed and that the next task can be

started. It is very important to note that the ECT and the LCT of the very last event (node) on the PERT chart should be equal.

- * Critical Path (CP).

The critical path/s are those route/s through the PERT chart on which all the ECT's and the LCT's are equal to each other and where the total of the ED's on that route is equal to the ECT and the LCT of the very last node.

- * Dummy tasks/activities.

A dummy task "represents a dependency between events.

However, because there is no activity to be performed, there is no duration between the events" (Whitten, 1994:772).

Figures 3-11 and 3-12 illustrate a method that can be used to produce a PERT chart. -

- Step 1: List all tasks to be performed in the sequence of occurrence (refer table 3 - 1)
- Step 2: Assign each task a task identity (an alphabetic character)
- Step 3: For each task, list the start event and the end event
- Step 4: Obtain from experts an optimistic, a pessimistic and a most likely estimation of the time it takes to complete the task.
- Step 5: Calculate the expected duration of each task. The formula for this calculation is

$$ED = (OT + (4 * MLT) + PT) / 6$$
- Step 6: Draw the project network containing the following:
1. Circles that represents the nodes or events
 2. Arrowed lines representing the tasks
 3. Dotted or dashed lines representing dummy activities. The task number for a dummy task always start with a "D" followed by a numeric.
- Step 7: Calculate the ECT for each event using the following three rules:
1. Consider each task entering the node
 2. For each one of these tasks, calculate an ECT:

$$ECT = ECT \text{ (start node)} + ED \text{ (task)}$$
 3. Select the highest ECT obtained.
- Step 8: Calculate the LCT for each event using the following three rules:
1. Consider each task leaving the node
 2. For each one of these tasks, calculate an LCT:

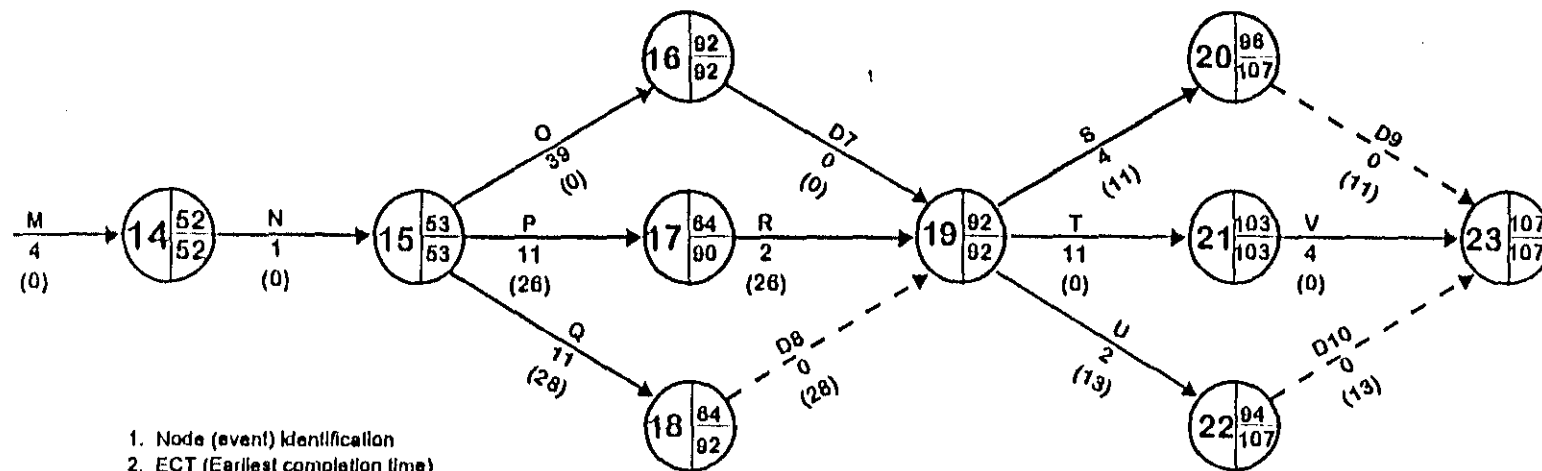
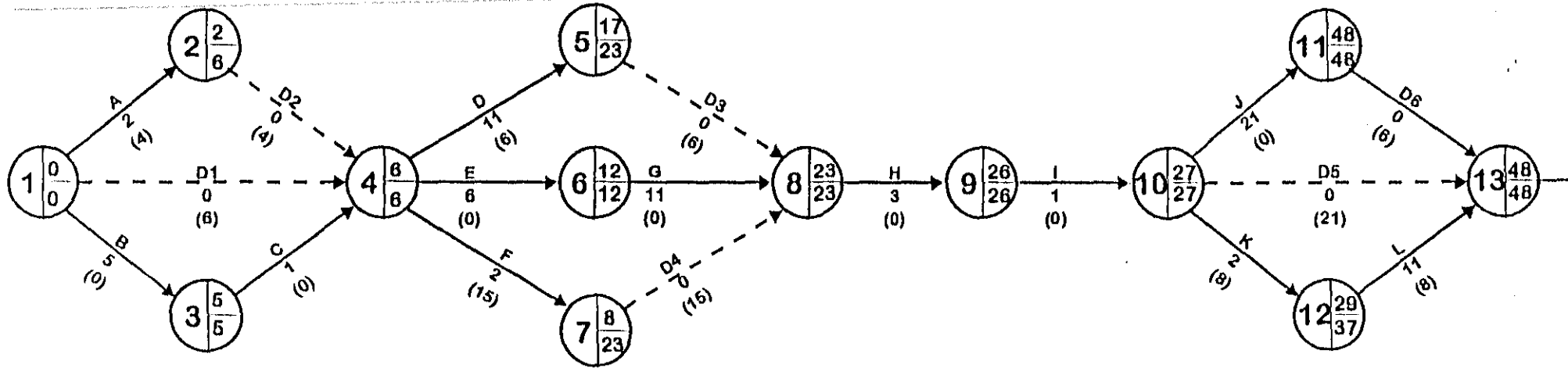
$$LCT = LCT \text{ (end node)} - ED \text{ (task)}$$
 3. Select the lowest LCT obtained.
- Step 9: Determine and mark the Critical Path or Paths
- Step 10: Calculate the Slack Time for the tasks not on the Critical Path/s using the following formula:

$$ST = LCT \text{ (end node)} - ED \text{ (Task)} - ECT \text{ (start node)}.$$

Figure 3 - 11: Steps: PERT diagram (own example)

Task	Task description	Start	End	OT	MLT	PT	ED
id		node	node				
A	Establish planning team	1	2	1	2	3	2
B	Develop a project plan	1	3	2	5	8	5
C	Review and agree plan with management	3	4	1	1	1	1
D	Model enterprise	4	5	6	10	16	11
E	Identify areas to be investigated	4	6	4	5	8	6
F	Establish analysis team	4	7	1	2	3	2
G	Model the business area	6	8	5	10	16	11
H	Plan development strategy	8	9	2	2	3	3
I	Review and agree plan	9	10	1	1	1	1
J	Collect relevant information	10	11	12	20	30	21
K	Define project scope	10	12	1	2	3	2
L	Classify problems and solutions	12	13	9	10	12	11
M	Establish project plan	13	14	1	4	5	4
N	Review and agree plan	14	15	1	1	1	1
O	Analyze current system	15	16	20	40	50	39
P	Model current system	15	17	9	10	12	11
Q	Analyze problems and opportunities	15	18	9	10	14	11
R	Review findings	17	19	1	2	3	2
S	Identify requirements	19	20	1	4	6	4
T	Model and specify requirements	19	21	4	10	19	11
U	Modify project plan	19	22	1	2	3	2
V	Review requirements specification	21	23	2	3	5	4

Table 3 - 1: Project tasks (own example)



1. Node (event) identification
2. ECT (Earliest completion time)
3. LCT (Latest completion time)
4. Task (Activity) identification
5. ED (Expected duration of task)
6. ST (Slack time)

CP (Critical Path) = B-C-E-G-H-I-J-D6-M-N-O-D7-T-V
 Total expected duration for these tasks:
 $5+1+8+11+3+1+21+0+4+1+39+0+11+4 = 107$

Figure 3-12: PERT diagram for the project tasks (own example)

From the PERT diagram in Figure 3 - 12 the following tasks must be closely monitored and controlled by project management so that the real duration of the project does not exceed the total expected duration. If any of the following tasks fall behind schedule, the whole project will be delayed:

- Develop a project plan
- Review and agree plan with management
- Identify areas to be investigated
- Model the business area
- Plan development strategy
- Review and agree plan
- Collect relevant information
- Establish project plan
- Review and agree plan
- Analyze current system
- Model and specify requirements
- Review requirements specification

To illustrate the existence of dummy tasks, focus on event 8 on the PERT chart (Figure 3-12), and note that events 5 and 8 are linked by a dashed line. Event 5 marks the end of task D (Model enterprise). Event 8 marks the end of task G (Model the business area), but also the start of task H (Plan development strategy).

Before the development strategy can be planned, both the

modelling of the enterprise and the business areas should be completed. Therefore, there is a dependency relationship between events 5 and 8. Task D3 shows this relationship, but consumes no resources.

It is, however, important to note that a dummy task can form part of the critical path.

3.3 SYSTEMS DEVELOPMENT METHODOLOGIES

Information technology departments can fully standardise by using a systems development methodology. This means that all systems can be developed in exactly the same way using the same tools, techniques and CASE tools (if part of the methodology). It also follows the same steps of a particular SDLC.

A "methodology is a collection of procedures, techniques, tools and documentation aids which help the systems developers in their efforts to implement a new information system" (Avison, 1990:4).

According to Whitten (1994:159) "commercial methodologies are off-the-shelf, step-by-step procedures, individual and group roles, deliverables, quality standards, preferred techniques and tools for completing the entire systems development life cycle".

Avison (1990:5) lists the objectives of methodologies as:

- * To record requirements for an information system in such a way that both developer and user understand.
- * To provide a systematic method of development in such a way that progress can be effectively monitored.
- * To provide an information system within an appropriate time limit and at acceptable costs.
- * To produce a system that is well documented and easy to maintain.
- * To provide an indication of any changes which need to be made as early as possible in the development process.
- * To provide a system which is liked by the people affected by that particular system.

A methodology becomes particularly useful when inexperienced systems analysts are employed. If these inexperienced systems analysts follow the step-by-step procedures and related aspects of the prescribed methodology to the letter, the risk of doing something wrong diminishes almost completely. Although some regard this as a 'monkey-see, monkey-do' method, the new systems analyst can only learn from the exercise and at the same time gain confidence in doing things right. When developing systems (especially commercial and administrative type of systems), there are so many variables that need to be considered in each and every system that time cannot be wasted on 're-inventing the

wheel'. Furthermore, it can only benefit the whole company if developer and user know exactly what to do, when to do it and how to do it.

Another very important benefit of methodologies is that it promotes consistent, standardised documentation. This is very beneficial when it comes to maintenance. The systems analyst responsible for the support of the system knows exactly what to look for, as well as where to look for it, when a problem needs to be identified, analyzed and corrected.

It is also important to note that the maintenance and improvement of a methodology are the responsibility of the vendor. It is therefore not necessary to allocate resources to keep up with the latest technology, techniques and trends used in the methodology.

There are a large number of commercial methodologies on the market today. Which one of these methodologies should a systems analyst recommend as a standard for the business? Avison (1990: 278) issues a list of features that need to be considered when comparing methodologies. These have been condensed to the following:

- * Does the methodology cover all aspects of systems development?
- * Are all the steps well defined?
- * Does each phase or stage require some understandable well documented deliverable?
- * To what types of applications is it suitable?

- * Are there built-in controls in the methodology and does each control measure success?
- * Will users be able to understand the steps they need to do?
- * Does the methodology improve quality of analysis, design and programming products and hence the overall quality of the information system?
- * Does the methodology provide an effective communication medium between analysts and users?

It is very important to choose the right commercial methodology, as it is an expensive product. Apart from acquiring the product, there are also additional costs of training developers and users in using the methodology properly. Careful evaluation of each product as well as a thorough comparison between products prove to be a worthwhile and useful exercise before buying a commercial methodology.

Correctly selected systems development life cycle, tools, techniques and methodologies are essential components of the systems development process. The most important component of systems development, however, is the people who combine all these other components to eventually produce a working and user-friendly computer-based system. The skills needed by these people to accomplish this challenging task, are discussed in the next chapter.

CHAPTER 4: SYSTEMS DEVELOPMENT SKILLS

4.1 INTRODUCTION

In the process of analysing, designing, implementing and supporting computer systems, the systems analyst does not only work with hardware and software, but also with people and managers at all levels, manual procedures, data and information as well as business networks. Systems analysis and design is therefore much more than tools, techniques, methodologies and systems development life cycles. There is a variety of general principles that underlie all systems development. A person also requires a large variety of skills to really become a successful systems analyst. These skills cannot be learnt, but are acquired through exposure, experience and exercise.

4.2 SYSTEMS DEVELOPMENT PRINCIPLES

Many experienced systems analysts still fail in the process of developing computer systems because they ignore the basic principles of systems development. Whitten (1994:91) states the following eight principles as the most important to consider when developing systems:

4.2.1 Get the user involved.

A systems builder should always remember that a system is built for the user. The user is the person who pays for the system and

has to use the system. It is always best to involve the user right from the beginning of the systems development project so that problems such as misunderstandings can be minimised.

4.2.2 Use a problem-solving approach.

This principle is taken into consideration when developing systems according to a systems development life cycle (SDLC).

4.2.3 Establish phases and tasks.

It is important to establish a program or process by which systems can be developed. Most SDLC's consists of phases and the phases are usually broken down into tasks.

4.2.4 Establish standards for consistent development and documentation.

If each user, analyst and programmer working for the same business had to use their own SDLC, tools, techniques and methodologies, systems development would be chaotic in that business. The reason for this is that employees come and go. Continuity would and could not prevail in such a business. This makes supporting systems almost impossible.

4.2.5 Justify systems as capital investments.

Systems should become an asset to a business. They should be regarded the same as a product that is supposed to be used for a

number of years, such as a motor car, furniture, etc.

4.2.6 Don't be afraid to cancel or revise scope.

A significant of a phased approach to systems development is that one can evaluate feasibility and viability and various stages of the development. If at any one of these checkpoints calculations show that it is not feasible to proceed with the project, one should not hesitate to cancel the project. It should also be accepted by users and management to change the scope of the project, if required.

4.2.7 Divide and conquer.

The old saying 'if you want to learn anything, you must not try to learn everything - at least not all at once' is the underlying statement for this principle. It is impossible to learn everything about one system at once. By dividing a large problem into smaller manageable units, the problem-solving can be simplified.

4.2.8 Design system for growth and change.

Businesses operate in a changing environment. This fact should always be considered and catered for as far as possible, when designing computer systems.

4.3 SYSTEMS DEVELOPMENT SKILLS REQUIRED

Many experienced systems analysts and information technology managers have different views on the importance-rating of skills required by prospective systems analysts.

Whitten (1994:19) lists the following skills as very important for becoming a systems analyst:

4.3.1 Working knowledge of information systems and technology.

It is important for the systems analyst to keep abreast with the latest technology so that the user can eventually reap the benefits of having a more efficient environment. This is not a stagnant process. The student has to develop a discipline and habit of reading various periodicals on computers and information systems.

4.3.2 Computer programming experience and expertise.

The systems analyst is the link between the user and the programmers. It is not necessary for the systems analyst to be an expert, but he should be able to appreciate the problems of both the user and the programmer when it comes to solving a business problem by means of a computer program. The systems analyst has to write program specifications in such a way that the programmer can understand and interpret it correctly. It is

therefore necessary for the systems analyst to have at least some programming background. Today's systems analysts should be familiar with at least one fourth generation programming language (CLARION, DBASE, etc) and at least one third generation programming language (COBOL, C, Pascal, etc).

4.3.3 General business knowledge.

The systems analyst should be able to communicate with a variety of business users. It is therefore imperative that prospective systems analysts learn the basic principles of business subjects such as Financial and Cost Accounting, Economics, Human Resources, Inventory control and Management.

4.3.4 Problem-solving skills.

The systems analyst should break a problem down into components, analyze each of these components, identify and correct the problems in each component and eventually assemble an improved system to solve the global problem. Systems analysts should also be able to define various solutions to a problem.

4.3.5 Interpersonal communication skills.

The systems analyst has to convey complex information to various levels of users as well as different types of technical specialists. A misinterpreted message can be very costly to the enterprise. It is

therefore important that the systems analyst is able to communicate effectively, both orally and in writing.

4.3.6 Interpersonal relations skills.

Individual users have conflicting goals and needs. They have personality clashes. The systems analyst has to mediate such problems and achieve benefits for the business as a whole.

The analyst is also sometimes labelled as 'an agent of change'. As most users are comfortable with the status quo of systems, they feel threatened when approached by systems analysts for the purpose of analysing and designing a new or changed system. The analyst should be able to put them at ease and persuade them that the exercise is for the benefit of the business.

4.3.7 Flexibility and adaptability.

There is no single approach to the development of systems. Due to systems being different, the systems analyst should learn to be flexible and adapt to special challenges or situations presented by specific systems development projects.

4.3.8 Character and ethics.

Systems analysts are sometimes exposed to sensitive and confidential information. They sometimes write programs that contain algorithms unique to the business enterprise. It is therefore

expected from the analyst to be reliable, a person of strong character and a sense of ethics.

4.3.9 Systems analysis and design skills.

Systems analysis and design skills are factored into three subsets, which are:

concepts and principles,
tools and
techniques.

Systems analysts can work with highly sophisticated technology, such as costly CASE tools which can produce models, project repositories, programs, etc at an alarming rate. However, all these wonderful tools can fail, and it is then expected of the analyst to be able to continue with the project. The analyst who remembers the basic principles and concepts of systems work, which include systems analysis and design techniques, will be able to continue. It is sometimes expected from the analyst to draw process and data models with only a pencil, paper and flowchart template. It is therefore necessary that the prospective analyst must take note of all three subsets.

Kendall (1992:27) states that an analyst should enjoy working with people, as he or she serves as a translator and a buffer between technical staff, such as programmers and nontechnical staff, or the managers and

users. Furthermore, an analyst should be a good diplomat and a good motivator. An analyst should also be able to work in a team and should therefore learn to 'check their egos at the door'. As a team leader, the analyst coordinates the efforts of programmers, estimators, testers and other analysts, as well as resolves conflicts and monitors the result of team effort. In order to do this, the analyst has to be a good manager.

According to Whitten (1994:7), the systems analyst also needs to train end-users and managers to use new and unfamiliar computer-based systems. Whitten (1994:10) states further that a systems analyst should estimate personnel requirements, budgets and schedules (planning skills), document current system operations (administrative skills) and select hardware and software purchases (negotiating skills).

CHAPTER 5: EMPIRICAL RESEARCH

5.1 INTRODUCTION

The research done in this study can be divided into two distinct areas, namely a literature study and an empirical research.

* Literature study:

The relevant literature is described in chapters 2 to 4 and also includes the way in which some of the systems development principles, tools and techniques are taught to the students at the institution where the researcher lectures (hereafter called the Technikon).

* Empirical research:

This research and analysis of the results are discussed in this chapter.

5.2 RESEARCH METHODOLOGY

One of the purposes of this research was to establish which systems development approaches, principles, tools and techniques should be taught to students so as to equip them sufficiently for a career as a systems analyst in the industry. It was therefore necessary to consult businesses who will most probably employ qualified students from the Technikon.

One of the major problems was to determine which companies to consult. It was finally decided by the researcher, study leader and employer to consult those companies who participate in the co-operative education program of the Technikon, as well as those companies who have indicated in the past that they are interested in joining this program. It was also suggested that a number of ex-students from the Technikon, who are permanently employed, be included in the study population.

5.2.1 DEVELOPMENT OF QUESTIONNAIRE

A questionnaire, consisting of eleven sections and a total of 80 questions, was developed. Some of the questions were divided in a number of sub-sections which resulted in a total number of 160 variables to be analyzed. A summary of the sections and the number of questions and variables per section, is shown in Table 5.1. A copy of the questionnaire is also included as Appendix B of this thesis.

SECTION	PURPOSE OF SECTION	NUMBER OF QUESTIONS	NUMBER OF VARIABLES
1	To obtain general information about respondent and company	5	5
2	To justify training at tertiary institution	2	2
3	To determine the types of problems experienced in industry	1	8
4	To determine how important the industry rate the SDLC	11	11
5	To determine how important the industry rate the use of tools and techniques	25	25
6	To determine what skills are required in the industry	12	12
7	To determine the popularity of tools and techniques (how often are they used)	8	21
8	To determine the popularity of various programming languages, methodologies, computer-aided software tools, systems development methodologies and prototyping tools	5	66
9	To determine the popularity of the new Object-oriented approach to systems development	4	3
10	To determine in which areas of Information Technology staff is needed	3	6
11	To evaluate the textbook currently used at the Technikon and to determine in which area the industry can support or assist the Technikon in training	4	1
		80	160

Table 5.1 Summary of questionnaire sections

The questionnaire consisted of different sections containing questions on the following topics:

* GENERAL INFORMATION

This section consisted of five questions in which the respondent had to give information such as job title, the sector of the industry to which the institution belongs, the number of Information Technology staff employed by the institution and the number of analysts reporting to him.

* TRAINING

Only two questions were asked in this section. The respondent had to confirm if the employer trains programmers and/or analysts.

* TYPE OF PROBLEMS EXPERIENCED IN INDUSTRY

This section consisted of only one question (but eight sub sections) to which the respondent had to answer with a Yes or a No whether the institution experiences a specific problem or not.

* **IMPORTANCE OF THE SYSTEMS DEVELOPMENT LIFE CYCLE**

For this section the respondent had to rate the importance of the SDLC as well as the various phases of the SDLC on a 7 point scale. This section contained eleven questions. The following scale was used:

Irrelevant	Not important	Important	Of Critical Importance			
1	2	3	4	5	6	7

For the purpose of this study a rating of 5, 6 or 7 was regarded as very important.

* **IMPORTANCE OF TOOLS AND TECHNIQUES IN SYSTEMS DEVELOPMENT**

There were 25 questions in this section where the importance of various tools and techniques also had to be rated on the same 7 point scale used in the previous section.

* **SKILLS REQUIRED TO BECOME A SUCCESSFUL SYSTEMS ANALYST**

This section consisted of twelve questions also to be rated according to the 7 point scale of importance. In this section the importance of certain essential systems development skills was determined.

* **POPULARITY AND USAGE OF SPECIFIC TOOLS AND TECHNIQUES**

This section contained eight questions. The purpose was to determine the popularity of, or the rate at which the industry uses specific tools and/or techniques. The respondent had to rate the popularity according to the following 7 point scale:

Never	Sometimes	Mostly	Always			
1	2	3	4	5	6	7

For research purposes a rating of 5, 6 or 7 was regarded as very popular.

* POPULARITY OF PROGRAMMING LANGUAGES, COMPUTER-AIDED DEVELOPMENT TOOLS, SYSTEMS DEVELOPMENT METHODOLOGIES AND PROTOTYPING TOOLS

This section consisted of five questions where the respondent had to indicate whether the products listed were used in the company or not.

* NEW APPROACH TO SYSTEMS DEVELOPMENT

This section dealt with the new object-oriented approach to systems development and consisted of only four questions. The respondent had to answer with a Yes or a No to three of the questions while the fourth question expected a product name as an answer.

* INFORMATION TECHNOLOGY STAFF NEEDED

For this section the respondent had to prioritize the need of the various staff categories. The job titles of the first three indicated by the respondent were included in the analysis of the results. The respondent also had to indicate whether students from the Technikon were employed by his institution, and if so, in which capacity. This section contained three questions.

* EVALUATION OF TEXTBOOKS AND INDUSTRY SUPPORT

The respondent had to answer four open questions for this section. The purpose of this section was to determine the popularity of textbooks in the industry and also to investigate the possibility of, and to what extent, industry support can be applied to Technikon needs.

5.2.2 SAMPLING PROCEDURE

A questionnaire consisting of 80 questions that required approximately 30 minutes to complete, was distributed to all the companies in the Western Cape who participate in the Technikon's co-operative education program. It was also distributed among 30 post-diploma students which represented more than 20 companies. Copies of the questionnaire were therefore distributed to more than 35 companies in the various sectors of the industry. Where possible, an employee of the company (hereafter referred to as an agent) was requested to distribute the questionnaire to the key personnel in Information Technology or Information Systems Services departments of the company. For the purpose of this study information technology managers, systems development project managers, systems analysts, analyst/programmers, systems designers, programmers, database administrators, network

administrators or any persons who are employed with the purpose of developing systems, were regarded as key personnel.

It was not possible to keep a proper record that reflects the distribution of questionnaires 100% correctly because -

- * agents were allowed to make photocopies of the questionnaire if more copies were required,
- * some agents requested students to collect questionnaires directly from the researcher,
- * some questionnaires were sent via e-mail to various businesses and private persons,
- * the sectors in which the post-diploma students work is not known to the researcher,
- * full-time under graduate students offered to request family members and friends working in the information technology industry to complete questionnaires and
- * most of the respondents insisted on, and were assured of total confidentiality; therefore company names could not be obtained.

According to the records kept of questionnaires distributed by the researcher the following sectors were asked to participate:

SECTOR	%
Financial services	36.4
Commercial trading	22.7
Business services	17.2
Manufacturing	13.1
Public sector	10.6

Table 5.2: Sectors requested to participate

5.2.3 EVALUATION OF RESPONSES TO THE SURVEY

For the purpose of this research the response sample was analyzed with the aid of BMDP Statistical Software (Dickson, 1981) in order to determine -

- * the problems that are mostly experienced by the industry,
- * the most important tools and techniques,
- * the most important phases of the SDLC,
- * the most wanted skills of an IT person in industry,
- * which tools, techniques and programming languages are mostly used in the industry,
- * which textbooks are used by the industry and
- * to confirm the necessity for the Technikon to train

prospective systems analysts.

5.2.3.1 RESPONSE RATE

A total of 250 copies of the questionnaire were distributed, of which 112 were returned. Therefore, an overall response rate of 44% was obtained. In this response the various sectors of the industry were represented as follows:

SECTOR	%
Financial services	33.9
Other	19.6
Business services	16.1
Commercial trading	12.5
Public sector	10.7
Manufacturing	6.2
Construction	0.9

Table 5.3: Response rate

From many personal and telephone conversations with some of the respondents who were prepared to reveal their identity, the researcher could deduct that the reason for the relatively high percentage of 'other' sector is that a large portion of the respondents regarded consultancies, systems development contractors, computer distributors and publishing not as business services or commercial trading. The intention of listing business

services as a sector in the questionnaire was to include these type of businesses. Unfortunately, the majority of respondents in this sector indicated the sector as 'other'. As the respondents were assured that their responses would be treated confidentially, it was impossible to determine exactly how many of the four sectors mentioned were recorded as 'other'.

5.2.3.2 DURATION OF SURVEY

The objective was to have all the questionnaires returned within a period of 6 weeks. Questionnaires were sent out and distributed during May 1996 and were expected to be returned by the end of June 1996. The majority of the completed questionnaires were received towards the end of July 1996.

5.3 ANALYSIS OF THE DATA

5.3.1 FREQUENCY ANALYSIS

The software application that was used to do the frequency analysis is BMDP2D - Frequency Analysis (Dickson, 1981). Appendix C contains the detailed results of this analysis.

* GENERAL INFORMATION

Table 5.4 shows clearly that more than 50% of the respondents were in the job categories that are really involved in systems analysis and design phases more than systems implementation.

JOB TITLE	%
Project manager	22.3
Analyst/programmer	18.8
IT Manager	14.3
Systems Analyst	14.3
Programmer	12.5
Other	7.1
Software Engineer	4.5
Database Administrator	3.6
Systems Designer	1.8
Network Administrator	0.9

Table 5.4: Job titles of respondents

It was also necessary to determine whether the respondent supervised systems analysts and/or analyst programmers. Table 5.5 shows the answer to the question whether the respondent has any systems analysts reporting to him. The majority indicated that they did not have any systems analysts reporting to them, but from Table 5.4 it is clear that more than 30% of the respondents were

actually systems analysts and/or analyst/programmers themselves. The 'Yes' and the 'Sometimes' groups in Table 5.5 indicates that almost 50% of the respondents manages systems analysts which automatically involves them in the analysis and design phases as well.

Answer to question	%
No	52.7
Yes	34.8
Sometimes	12.5

Table 5.5: Supervision of systems analysts

The majority of the respondents worked in businesses with more than 100 Information Technology employees. Table 5.6 shows the distribution of the various sizes of the Information Technology departments.

NUMBER OF IT STAFF	%
101-200	29.5
201-500	17.9
1-20	16.1
21-50	13.4
51-100	12.5
> 500	10.7

Table 5.6: Number of IT staff

* TRAINING

The questions that were asked to give the results shown in Table 5.7 were:

- do you train programmers and
- do you train systems analysts.

According to Table 5.7 the majority of businesses is not prepared to train programmers and, especially, systems analysts.

QUESTION	% YES reply
Train programmers?	45.5
Train analysts?	28.6

Table 5.7: Need for systems analyst training

* PROBLEMS EXPERIENCED BY INDUSTRY

It is clearly indicated by Table 5.8 that the types of problems that are experienced by the majority of respondents, are, in order of percentage positive responses to the question -

- the increasing rate of need for systems
- the lack of proper communication between user and developer

- the lack of user understanding
- the lack of understanding of systems development principles

PROBLEM	%
Rate of increasing need for systems	72.3
Lack of communication between user and developer	71.4
Lack of user understanding	65.2
Lack of understanding systems development principles	50.9
Lack of proper training	40.2
Too many approaches to systems development	33
Too many skills required	23.2
Other	17

Table 5.8: Systems development problems

* **IMPORTANCE OF THE SYSTEMS DEVELOPMENT LIFE CYCLE**

The implementation phase is regarded by the majority of respondents as the most important phase of the SDLC. It is clear, as indicated by Table 5.9 that all the phases are actually very important.

SDLC PHASE	%
Implementation	89.3
Analysis	88.5
Planning	86.6
Design	83.4
Support	77.7

Table 5.9: Importance of the SDLC phases

* **IMPORTANCE OF TOOLS AND TECHNIQUES IN SYSTEMS DEVELOPMENT**

The importance of the various tools and techniques as rated by the respondents is shown in Table 5.10. The majority of respondents regarded most of the tools and techniques listed as very important. Only three techniques were not rated as very important. These techniques were:

- decision trees and tables,
- PERT diagram, and
- network modelling.

TOOLS AND TECHNIQUES	
Data Analysis	84.8
Entity-Event Analysis	81.3
Data Modelling	80.7
Walkthroughs	79.5
Process Modelling	78.6
Presentations	75.9
Output Design	75.8
Input Design	73.2
Project Management	71.4
Report Writing	70.5
Fact-Finding	69.9
Cross Reference	63.4
Cost Benefit Analysis	60.9
Decision Trees & Tables	43.8
PERT diagram	39.3
Network Modelling	36.1

Table 5.10: Importance of tools and techniques

For the purpose of the table interviewing, observation, questionnaires and joint-application-design (JAD) were grouped together as fact-finding. The questionnaire, however, listed these four techniques separately and it was therefore possible to obtain the rating for each one of these techniques. It is very interesting to note that interviewing scored the highest in the importance rating of all the techniques. The ratings for these four fact-finding techniques are individually shown in Table 5.11.

FACT FINDING TECHNIQUE	%
Interviewing	88.4
Observation	76.8
JAD	70.6
Questionnaire	43.8

Table 5.11: Fact finding techniques

* **SKILLS REQUIRED TO BECOME A SUCCESSFUL SYSTEMS ANALYST**

Table 5.12 gives the rating of the skills needed to become a successful systems analyst as listed in the questionnaire by the

respondents as very important. It is very interesting to note that, although implementation is regarded as the most important phase of the SDLC (refer Table 5.9), both programming skills in a third or fourth generation language scored less than 70% very important rating. Interpersonal communications, interpersonal relations and general business knowledge are the three skills that are clearly mostly required.

SKILL REQUIRED	%
Interpersonal communications	92.9
Interpersonal relations	92.9
General business knowledge	89.3
Basic understanding of methodologies	78.6
Basic understanding of prototyping	74.1
Working knowledge of prototyping	72.3
Working knowledge of methodologies	69.6
3GL programming skills	61.6
Basic understanding of CASE tools	58.0
4GL programming skills	53.7
Working knowledge of CASE tools	53.4

Table 5.12: Skills required

* POPULARITY AND USAGE OF SPECIFIC TOOLS AND TECHNIQUES

According to the responses for this section reflected in Table 5.13, program design is the technique that is used the most in the industry. This technique is followed by data modelling, project repository and prototyping for verification purposes. The rest of the techniques are not so popular as they scored less than 50%. It is interesting to note that the respondents rated the importance of the use of these tools and techniques (refer Table 5.10) much higher than the actual usage itself.

TOOLS AND TECHNIQUES USED	%
Program design	74.1
Entity Relationship Diagram /Logical Data Structure	55.3
Project repository	53.5
Prototyping for verification purposes	52.7
Data Flow Diagrams	48.5
Prototyping for implementation	39.9
Structured English	33.1
CASE tools	28.8

Table 5.13: Popularity of tools and techniques

* POPULARITY OF PROGRAMMING LANGUAGES, COMPUTER-AIDED DEVELOPMENT TOOLS, SYSTEMS DEVELOPMENT METHODOLOGIES AND PROTOTYPING TOOLS

Table 5.14 reflects the popularity of the different programming languages used in industry. Cobol is still rated as the most popular third generation language. It is also noticeable that C++ is very rapidly gaining popularity if taking into account that the majority of the study population was in the financial, commercial and business services sectors. These sectors usually make use of business-oriented programming languages, whereas C++ is accepted in both the business and the scientific disciplines. The group 'other' was listed in the questionnaire mainly to collect information about the usage of fourth generation languages. The reason for not listing fourth generation languages by name was that there were too many different products on the market in this category and it was therefore up to the respondent to supply the name of the product used (if used) in his business. As expected, 28 different products were supplied of which the following seven (representing 65% of all the fourth generation languages used in this survey) are the most popular:

- Visual Basic (14%)
- Clipper (10%)

- Oracle (9%)
- Informix (8%)
- Delphi (8%)
- Natural (8%)
- SQL Windows (8%)

PROGRAM LANGUAGE	%
Other	73.2
Cobol	57.1
C + +	33.9
C	33
Assembler	17
Pascal	9.8
Basic	9.8
Fortran	5.4
RPGII	1.8
ADA	0.9
ALGOL	0
MAGIC	0

Table 5.14: Popularity of programming languages

Some other very interesting statistics about the usage of programming languages, are the following:

- 55% of the respondents use at least one 3rd generation programming language and at least one 4th generation language,
- 19% of the respondents use only a 4th generation language, and
- 26% of the respondents do not use 4th generation languages at all.

Although the majority of the respondents felt that a basic understanding, as well as a working knowledge, of CASE tools is an important skill, it is not used a great deal in the industry. According to Table 5.15, more than 50% of the respondents indicated that their business do not use any CASE tool at all. Due to the fact that there are so many specific CASE tool products being used by various businesses in the industry, it was impossible to list all the possibilities; therefore the category 'other' had to be used. Fourteen different products were supplied of which the following four (representing 69% of all the other CASE tool products used in this survey) are the most popular:

- Oracle (38%)
- LBMS System Engineer (19%)
- Erwin (6%)
- Construct (6%)

CASE TOOL	%
Do not use CASE	52.7
Other	30.4
Systems Architect	11.6
Cobol CICS	5.4
Bachman	4.5
ADW	3.6
PowerBuilder	3.6
IE	2.7

Table 5.15: Popularity of CASE tools

Table 5.16 indicates that SSADM scored the highest popularity-rating amongst commercial systems development methodology products for this survey. Here again (as was the case for CASE tools), it was just impossible to list all the various products being used in the industry; therefore the category 'other'. Nine different products were supplied of which the following four (representing 75% of all the other methodology products used in this survey) are the most popular:

- Rambach OMT (25%)
- Tetrach (20%)
- Prince (20%)
- Booch (10%)

METHODOLOGY	%
SSADM	31.3
Other	25
Infomet	10.7
IE	4.5
JSD	2.7
STRADIS	1.8
ISAC	1.8
SSM	1.8
ETHICS	0.9
Multiview	0.9

Table 5.16: Popularity of systems development methodology

The larger portion of the industry uses prototyping (refer Table 5.17) for either developing and implementing systems or verifying the user's requirements. This result is more or less in line with the requirement (indicated in Table 5.12) that a good understanding and a working knowledge of prototyping are requirements for becoming a successful systems analyst. Here again, due to the

large variety of products on the market, the questionnaire had to include a group "other" which obtained the highest rating.

Thirteen different products were supplied of which the following five (representing 75% of all the other prototyping products used in this survey) are the most popular:

- Oracle (22%)
- SQL Windows (19%)
- Delphi (16%)
- Informix (9%)
- SE (9%)

PROTOTYPING TOOL	%
Other	28.6
Do not use prototyping	28.6
Visual Basic	26.8
Clipper	13.4
Natural	9.8
Clarion	7.1
PowerBuilder	6.2
SAS	4.5
ADS/ON-LINE	4.5
Progress	4.5
Paradox	1.8
IDEAL	1.8
dBASE IV	1.8

Table 5.17: Popularity of prototyping tools

* NEW APPROACH TO SYSTEMS DEVELOPMENT

Object orientation is a relatively new approach to systems development that has not yet been used by many information technologists. Table 5.18 clearly confirms this statement although the interest in this approach is definitely growing. Almost half of the respondents indicated that they are interested in this approach, however, only 33% has successfully developed systems using this new approach. It is interesting to note that there is a close relationship between the percentage respondents that has developed object-orientated systems successfully and the use of C++ as a programming language (refer Table 5.14). It can be deducted from this relationship that most people developing object-orientated systems use C++ as the programming language.

QUESTION	% YES reply
Do you use object oriented approach for systems development	46.4
Have you developed any systems using Object Oriented approach	33
Do you use an Object oriented methodology	17

Table 5.18: Popularity of Object Orientation

* INFORMATION TECHNOLOGY STAFF NEEDED

Respondents had to prioritize the need for information technology staff and according to Table 5.19 the industry needs programmers, analyst/programmers and systems analyst the most. A large portion of the respondents (78.6%) have unfortunately only indicated the priority 1 need. Despite this, it can still be deducted from the results that the industry needs qualified programmers and systems analysts.

TYPE OF PERSON REQUIRED	%
Priority 1	
Programmer	34.8
Analyst/programmer	19.6
Systems analyst	10.7
Priority 2	
Analyst/programmer	26.8
Systems analyst	14.3
Database Administrator	8.9
Priority 3	
Analyst/programmer	8.9
Systems analyst	8.9
Project Manager	8.9

Table 5.19: Information technology staff needs

Table 5.20 suggests that a Technikon student who has qualified as an information technologist from the Technikon has a better chance of getting a permanent job than being employed on a temporary basis. It is, however, a hopeful sign that the training at the Technikon can be regarded as helpful to the industry.

EMPLOYMENT TYPE	%
As part of the co-operative program	56.2
On a temporary basis	54.5
On a permanent basis	60.7

Table 5.20: Employment: Technikon students

* EVALUATION OF TEXTBOOKS AND INDUSTRY SUPPORT

Textbooks are not very popular in the industry. This was clearly illustrated by the fact that only 32.1% of the respondents used textbooks. Furthermore, the majority of these respondents actually used manufacturer's or supplier's manuals which cannot be regarded as textbooks. It was therefore not possible to evaluate the applicability of the textbooks prescribed to students.

5.3.2 REGRESSION ANALYSIS

In the further analysis of the data, a component of a restricted linear regression model method was used (Bruwer, 1991). By means of all possible subset multiple linear regression analysis, a subset of the data of three population groups were analyzed to determine the most important aspects contributing to the respondents' view of the importance of the SDLC. The three population groups were:

- the full population
- systems development project and information technology managers
- systems analysts and analyst/programmers

In the regression analysis the importance of the SDLC indicated by X(16) was used as the dependent variable while variables X(17) to X(35), X(37) to X(50) and X(52) to X(62) were used as independence variables.

* FULL POPULATION

With the regression analysis of the data for the full population, seven variables were found in the 'best' subset of independent variables that declared 52% of the variation of the dependent variable. The results of this regression are given in Table 5.21.

VARIABLE	REGRESSION COEFFICIENT	STANDARD ERROR	T-STAT	CONTRIBUTION TO R ²
X(17)	0.357	0.09	3.96	0.065
X(23)	0.314	0.098	3.19	0.042
X(19)	0.241	0.087	2.78	0.032
X(31)	0.172	0.083	2.08	0.018
X(40)	0.151	0.073	2.08	0.018
X(28)	-0.142	0.074	-1.91	0.015
X(33)	-0.131	0.072	-1.81	0.013

$$R^2_a = 0,52$$

The regression equation being the following:

$$X(16) = 0.154 + 0.357 X(17) + 0.314 X(23) + 0.241 X(19) + 0.172 X(31) + 0.151 X(40) - 0.142 X(28) - 0.131 X(33)$$

Table 5.21: Results of regression analysis: Full population

If the contribution of the variables to the multiple regression coefficient is used as criteria for variable-importance, then Table 5.22 contains this variable-importance in priority-order.

PRIORITY	VARIABLE	DESCRIPTION
1	X(17)	Planning phase
2	X(23)	Implementation phase
3	X(19)	Analysis phase
4	X(31)	Balancing of data and process models
5	X(40)	Decision tables
6	X(28)	Data modelling
7	X(33)	Observation for fact finding

Table 5.22: Variable importance in priority order

* SYSTEMS DEVELOPMENT PROJECT AND INFORMATION
TECHNOLOGY MANAGERS

With the regression analysis of the data for the IT and project manager population nine variables were found in the 'best' subset of independent variables that declared 74% of the variation of the dependent variable. The results of this regression are given in Table 5.23.

VARIABLE	REGRESSION COEFFICIENT	STANDARD ERROR	T-STAT	CONTRIBUTION TO R ²
X(21)	0.335	0.093	3.61	0.083
X(43)	-0.948	0.263	-3.6	0.083
X(42)	0.889	0.258	3.44	0.076
X(38)	-0.222	0.069	-3.21	0.066
X(17)	0.303	0.096	3.17	0.064
X(20)	0.291	0.095	3.06	0.06
X(22)	0.297	0.112	2.65	0.045
X(24)	-0.193	0.088	-2.18	0.03
X(40)	0.113	0.06	1.88	0.022

$$R^2_a = 0,74$$

The regression equation being the following:

$$\begin{aligned}
 X(16) = & 0.523 + 0.335 X(21) - 0.948 X(43) + \\
 & 0.889 X(42) - 0.222 X(38) + 0.303 X(17) \\
 & + 0.291 X(20) + 0.297 X(22) - 0.193 X(24) + \\
 & 0.113 X(40)
 \end{aligned}$$

Table 5.23: Results of regression analysis: IT and project managers

Again if the contribution of the variables to the multiple regression coefficient is used as criteria for variable-importance, then Table 5.24 contains this variable-importance in priority-order.

PRIORITY	VARIABLE	DESCRIPTION
1	X(21)	Systems Design phase
2	X(43)	Output design
3	X(42)	Input design
4	X(38)	Location connectivity diagrams
5	X(17)	Planning phase
6	X(20)	Systems Analyst key facilitates Systems Analysis
7	X(22)	Systems Analyst key facilitates Systems Design
8	X(24)	Systems Analyst key facilitates Systems Implementation
9	X(40)	Decision tables

Table 5.24: Variable importance in priority order

* **SYSTEMS ANALYSTS AND ANALYST/PROGRAMMERS**

With the regression analysis of the data for the systems analysts and analyst/programmers, nine variables were found in the 'best' subset of independent variables that declared 87% of the variation of the dependent variable. The results of this regression are given in Table 5.25.

in Table 5.25.

VARIABLE	REGRESSION COEFFICIENT	STANDARD ERROR	T-STAT	CONTRIBUTION TO R ²
X(19)	0.467	0.087	5.35	0.1
X(38)	-0.523	0.103	5.1	0.09
X(49)	-0.438	0.092	-4.77	0.079
X(17)	0.402	0.095	4.23	0.062
X(57)	0.694	0.17	4.09	0.058
X(29)	0.244	0.077	3.15	0.034
X(55)	-0.299	0.097	-3.08	0.033
X(58)	-0.459	0.163	-2.8	0.027
X(53)	0.251	0.132	1.9	0.012

$$R^2_a = 0,87$$

$$\begin{aligned}
 X(16) = & -1.534 + 0.467 X(19) - 0.523 X(38) - \\
 & 0.438 X(49) + 0.402 X(17) + 0.694 X(57) \\
 & + 0.244 X(29) - 0.299 X(55) - 0.459 X(58) + 0.251 \\
 & X(53)
 \end{aligned}$$

Table 5.25: Results of regression analysis: Systems analysts and analyst/programmers

For this regression analysis Table 5.26 contains this variable-importance in priority-order.

PRIORITY	VARIABLE	DESCRIPTION
1	X(19)	Systems Analysis phase
2	X(38)	Location connectivity diagrams
3	X(49)	Presentations
4	X(17)	Planning phase
5	X(57)	CASE tool working knowledge
6	X(29)	Data Flow Diagrams
7	X(55)	Programming in high-level language
8	X(58)	CASE tool understanding
9	X(53)	Interpersonal relations

Table 5.26: Variable importance in priority order

5.4 CONCLUSION

The responses to the questionnaire were analyzed according to frequency and regression. It is, however, important to draw some conclusions from these analyses so that the essential elements for efficient training of systems analysts can be identified and described. This is exactly what the purpose of this research is. The conclusion, identification and description of these elements are discussed in the next chapter.

CHAPTER 6: INTERPRETATION OF RESULTS AND CONCLUSIONS

6.1 INTRODUCTION

In this chapter the results of the research are discussed with a view to including the essential elements of a systems development course that can be offered to prospective systems analysts, so as to equip them for a career in Information Technology.

In this research the importance of various systems development approaches, principles, programming languages, methodologies, tools and techniques as regarded by industry were determined. The popularity of some of the elements were also determined.

The main purpose of this research was to determine which approaches, principles, skills, tools and techniques should be taught to students. To this end, the frequency and regression analyses need to be discussed.

6.2 FREQUENCY ANALYSIS

For the discussion of this section, only the elements scoring more than a 50% importance rating are considered. The assumption is made that more than 50% of the industry rate the element as so important that the

prospective systems analyst should have at least a sound basic knowledge of it.

6.2.1 SYSTEMS DEVELOPMENT LIFE CYCLE

The SDLC is -

- * a systematic approach to systems development
- * a process whereby systems can be developed
- * a project management tool to plan, execute and control systems development

The SDLC contains many different phases and these phases were rated as shown in Figure 6.1.

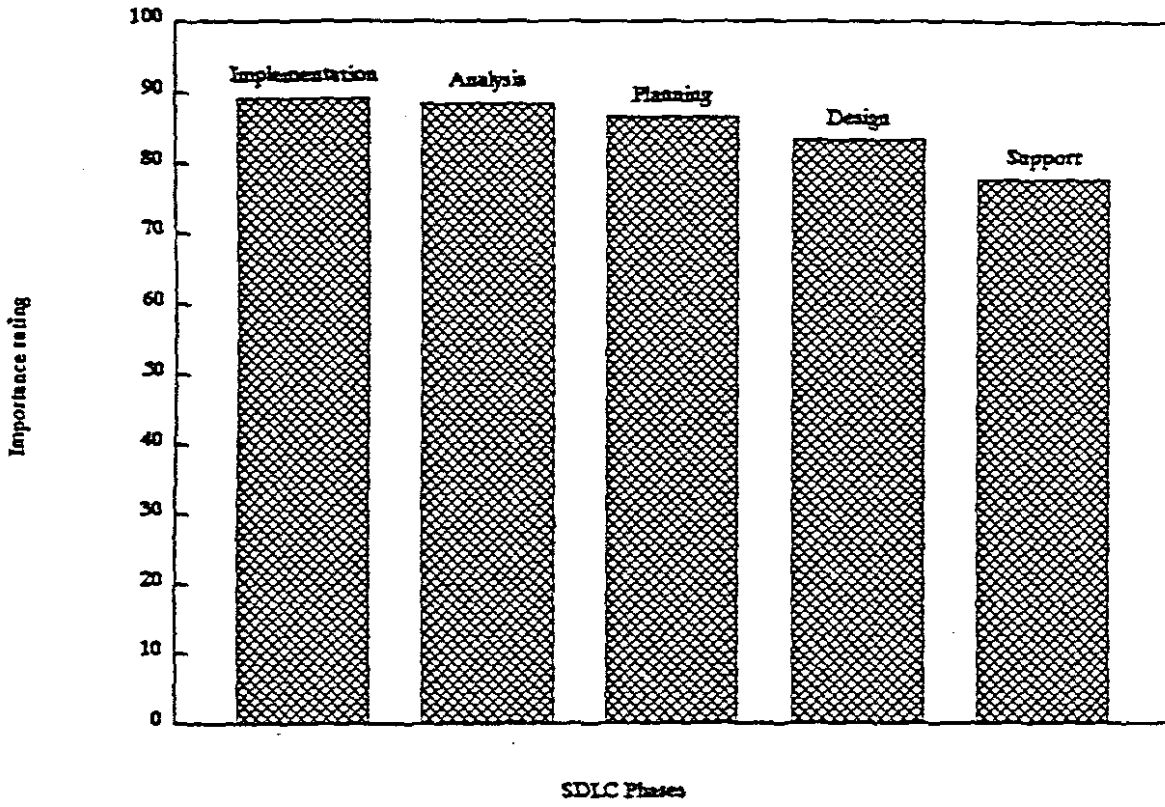


Figure 6.1: Importance of the SDLC phases

From the bar chart in Figure 6.1, it can clearly be deduced that all the phases of the SDLC are so important that a systems analysis and design course cannot be successful unless these phases are included in the course.

6.2.2 TOOLS AND TECHNIQUES

For the purpose of this research a technique is a systematic procedure whereby a certain objective can be achieved. A tool is merely the automation of the technique.

The results of the importance rating (in percentage) of the tools and techniques are depicted in the bar chart in Figure 6.2.

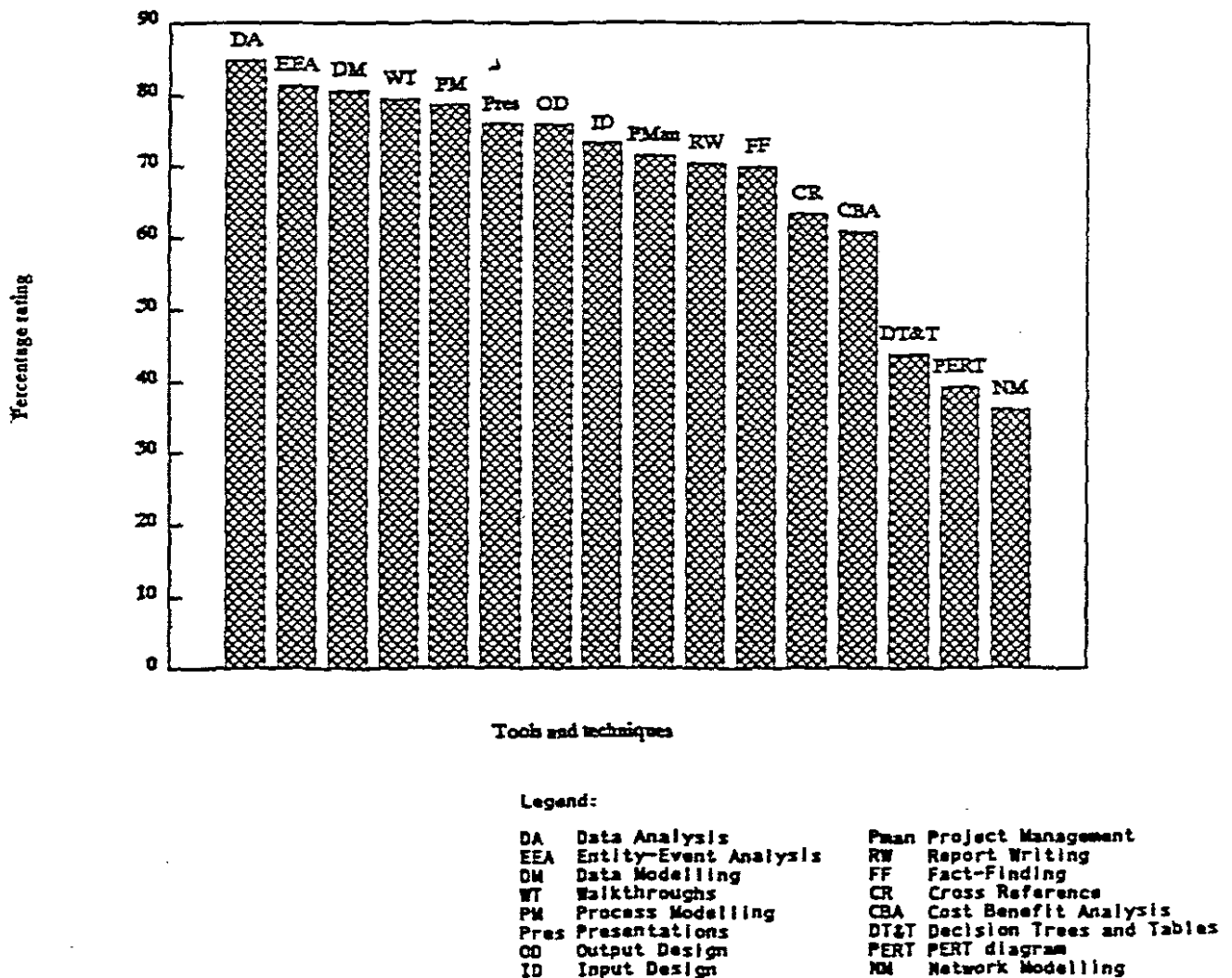


Figure 6.2: Importance of tools and techniques

The bar chart in Figure 6.2 indicates that any systems analysis and design course should include the following tools and techniques that scored an importance rating of more than 50%:

* **Data Analysis (84.8%)**

This is a procedure that is used to prepare a data model for implementation as a nonredundant, flexible and adaptable database.

The success of a computer system is very much dependent on the way its data structure is organized. It should be organized in such a way so that information can be stored and retrieved in the minimum time at the minimum cost by the right people at the right time. It is therefore necessary to consider all the requirements that can have an influence on each piece of data used by the system.

Object orientation approach is already used by almost 50% of the respondents in this survey. This fact emphasizes the importance of data analysis. If it is necessary to convert to object orientation and data analysis was done properly, the conversion should be easier for both user and developer. Why? Each and every piece of information requirement was considered carefully before organized and documented

properly.

* **Entity-Event Analysis (81.3%)**

This is also called the event analysis and refers to the study of the entities of a fully normalized data model with the purpose of identifying those business events and conditions that can possibly cause data to be created, deleted or modified.

An event can cause the attributes of an entity to be changed. It is therefore imperative to cater for these changes. For example, when a new record is created for a student enrolling for the first time, the database (the implemented version of the data model) should be able to store information about this new student in a separate record. This record should be linked to a specific course and to a specific school or faculty. Event analysis seeks to eliminate (if possible) or at least minimise the system failures caused by events not catered for.

* **Data Modelling (80.7%)**

This is a technique used to organise and document a system's data.

User and developer should have the same understanding of a system's data structure. Any piece of related data grouped

together is called an entity, and each occurrence of the entity has certain characteristics that makes it different from the other occurrences of the same type of entity. Furthermore, one entity can be associated with another entity or even with more than one other entity. It is very difficult to describe these entities, their attributes and especially relationships with other entities in words only. The only method that minimises misinterpretation by user and developer is by means of a picture or diagram. This diagram is called the data model.

* Walkthroughs (79.5%)

This is a peer group review to verify the validity and correctness of a particular systems development project's documentation. Although the ideal situation would be that the members participating in this review are all on the same level, it is acceptable that higher and/or lower level employees take part, but each and every member should then be treated as a peer for the purpose of the review.

The systems development process should be controlled; it is therefore necessary to have milestones or events where the accuracy of the exit documentation of the various phases or stages of the total systems development process is reviewed.

The formal review (usually by supervisors and management) is

called an inspection, but the informal review with peers is referred to as a walkthrough. The benefit of this technique is that it can save systems analysts and programmers considerable embarrassment.

* Process Modelling (78.6%)

This is a technique used to organise and document a system's processes, inputs, outputs and data stores in such a way that everybody involved (user and developer) has the same understanding of the system.

As for data modelling, the user and developer should have the same understanding of a system's processes, inputs, outputs and data stores. There are many processes that have an influence on the system, but only some of them might be of importance to the specific user for whom the system has been developed. This means that the systems analyst needs to determine the project scope (which processes should be included and which should be excluded for the purpose of a specific project). Again, it is very difficult to describe this type of situation in words only. The only method that minimises misinterpretation by user and developer is by means of a picture or diagram. In this case the diagram is called the process model.

- * **Presentations (75.9%)**

A presentation is actually a meeting, but with the sole purpose of selling new ideas (such as a new computer system) and gaining approval for it. A presentation is usually arranged after a systems developer has presented a report about the design or implementation of a new system to management and users, and if some queries need to be resolved.

Time is of the essence when computer systems are developed. Users usually have questions to ask if a system needs to be developed or, more specifically, a new system be implemented. Developers cannot explain the design and/or implementation of a system to each user individually at different times - it will waste too much time. It is much quicker and more efficient to put the users and developers together in one meeting where the users can put all their questions to the developer and the developer can explain to all at the same time what the design or implementation is all about.

- * **Output Design (75.8%)**

This is a technique used to invent, describe, depict and document methods to present the information produced by a computer system to the users of that system.

The output of any computer system is 'the system itself' to users. If the output is user friendly, the system is good, else, the system has not achieved what it was intended for. Output design, with the total involvement of the users, is thus of vital importance.

* **Input Design (73.2%)**

Whereas output design refers to the output produced by the system, input design invent, describe, depict and document methods to capture data and convert the captured data into a format suitable for the system to perform the necessary processes on it.

Captured data that is not acceptable to a computer system has no value. No processes can be performed on it and therefore no information can be derived from it.

It is also necessary to make the data capturing process as user friendly as possible. Data capture starts long before it needs to be physically entered into a computer system. Users need input forms and input screens to enable them to get the data captured. They also need user friendly procedures to capture data. All these forms, screens and procedures should be designed with the user's requirements and computer literacy

level in mind.

* **Project Management (71.4%)**

This management skill and systems development technique is a process of planning, directing, controlling and organizing the development of an acceptable system at the minimum cost within a specified period.

Managing a complex project can be very difficult. The best approach is to break the project down into a series of smaller tasks that can be easier planned and controlled. However, such subdivision may cause the whole project to collapse. It is therefore necessary that a systems analyst (who usually becomes the project leader or project manager of a systems development project) is to apply project management technique/s to secure the successful completion of a computer system project.

* **Report Writing (70.5%)**

This technique or skill is used to communicate information about the system and the development of it to various levels of users and developers in writing. The readers of these reports include managers at all levels, ordinary users, technical programmers and professional staff.

This is one of the most common and standard methods of conveying information within an organization. Writing a report is supposed to force the author of the report to explain each and every fact clearly and to verify the correctness of the facts. The reader, on the other hand, can read the report in his own time without affecting other people, and also formulate any queries or uncertainties about the report. A report can also be retained for further reference. Reports, such as feasibility reports, requirements statements and acquisition of hardware motivations are an integral part of a system's documentation.

* **Fact-Finding (69.9%)**

Fact-finding is the systematic approach and procedure to collect relevant information about systems, requirements, problems and preferences.

Fact-finding includes specific techniques such as questionnaires, interviews, observations and joint-application-designs. These techniques are used to collect valuable information from the user about the system so as to ensure the correctness of the requirements and new computer system design specifications. It is important for a systems analyst to apply these techniques not only correctly, but also at the right time and under the right circumstances. Many systems

analysts rely too much on interviews without realizing that interviews could be regarded as, although very efficient, still the most expensive fact-finding technique. Interviews are often not even planned well enough to justify the cost incurred by them. Project budgets are sometimes overspent due to this type of ruthless use of good techniques. A prospective systems analyst should always be made aware of this problem.

* **Cross Reference (63.4%)**

This technique is used to balance the individual items stored in the data store elements of a process model and the attributes catered for in the data structure of the data model for the same system.

It can be disastrous if a database cannot accommodate the input data required by a process, or cannot store the output data from a process. On the other hand, if a process which should store data in the database is not reflected on the process model, it can be equally disastrous. The data and process model should therefore be balanced by making use of this cross-referencing technique.

* **Cost Benefit Analysis (60.9%)**

This technique is used to measure the cost-effectiveness of a

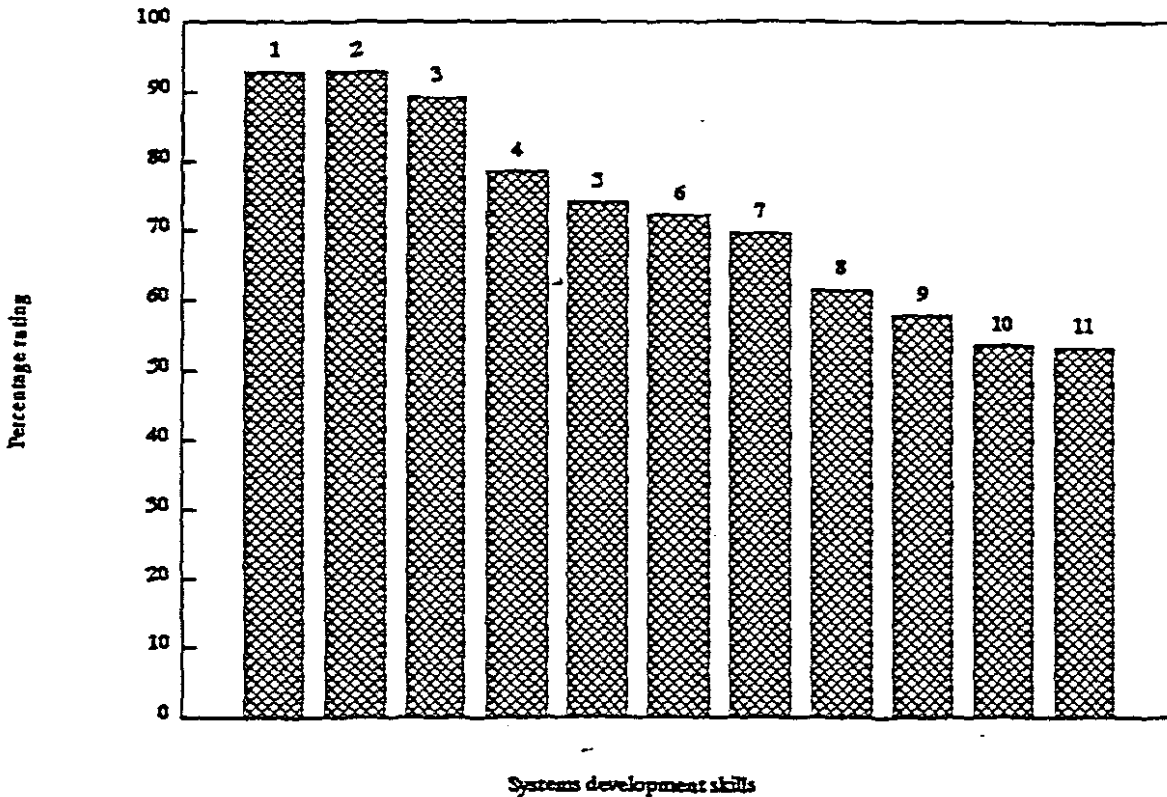
project or a solution to a systems problem.

The bottom line when determining the feasibility of a project, is money. If not enough funds are available for the development, implementation and day-to-day running of a computer system, the system can simply not be developed.

Although many information technologists regard this analysis as part of an accountant's duty, the systems analyst should be able to evaluate systems development and operational costs against the benefits that can be derived from the system. The systems analyst should be able to justify costs and benefits. He should therefore have a good general business knowledge (one of the required skills) as well as sufficient technical knowledge about software and hardware to be used in the system.

6.2.3 SKILLS

Skills refer to a person's ability to do something specifically, for example, communicating, reading and speaking.



Legend:

1. Interpersonal communications
2. Interpersonal relations
3. General business knowledge
4. Basic understanding of methodologies
5. Basic understanding of prototyping
6. Working knowledge of prototyping
7. Working knowledge of methodologies
8. 3GL programming skills
9. Basic understanding of CASE tools
10. 4GL programming skills
11. Working knowledge of CASE tools

Figure 6.3: Importance of skills

Figure 6.3 reveals the results of the importance rating of skills by the respondents.

* **Interpersonal communications (92.9%)**

This skill refers to the ability to communicate effectively, both orally and in writing.

* **Interpersonal relations (92.9%)**

For the systems analyst it is important to be able to work with people, to settle conflicting ideas, goals and needs from users and to convince people to change procedures and methods to make the organization more effective.

* **General business knowledge (89.3%)**

Systems analysts should be able to communicate with business experts to gain knowledge of business problems and needs.

* **Basic understanding of methodologies (78.6%)**

This skill refers to the ability to understand the philosophy behind commercial systems development methodologies which are step-by-step procedures, individual and group roles, tools, deliverables and quality standards for completing the systems development life cycle.

- * **Basic understanding of prototyping (74.1%)**

Prototyping is the process of building small-scale, working models of a user's requirements for the purpose of discovering or verifying the user's requirements. It can also be used to build complete systems. The systems analyst should understand when and why to use prototyping, what benefits can be derived from it and how to apply it effectively.

- * **Working knowledge of prototyping (74.1%)**

The working knowledge of prototyping refers to the real using of any specific prototyping tool. It is therefore necessary for the systems analyst to acquire enough knowledge of the prototyping tool so that he can develop small-scale, working models of the system and even implement a complete system with this prototyping tool. It is obvious that the systems analyst should know how to apply this prototyping tool within the constraints of the SDLC.

- * **Working knowledge of methodologies (69.6%)**

Whereas the basic understanding of methodologies refers to the understanding of the philosophy behind methodologies, the working knowledge of methodologies requires from the system's analyst to be able to develop and implement computer systems using the procedures, tools, documentation,

deliverable standards and quality standards of a specific methodology. This, of course, depends on the availability of the methodology product.

* **3GL programming skills (61.6%)**

The third generation programming languages include languages such as COBOL, FORTRAN, C, C++ and ALGOL. A systems analyst with a background of at least one 3GL programming language usually has an appreciation of the problems that can occur when implementing systems. This systems analyst is able to manage these tasks with more control.

* **Basic understanding of CASE tools (58.0%)**

This skill refers to the ability to understand the philosophy behind CASE tools which is the application of computer technology to systems development activities, techniques and methodologies. The systems analyst should be able to justify the use of CASE tools taking into account costs, benefits, technical skills and user computer literacy.

* **4GL programming skills (53.7%)**

If programs need to be developed quicker than usual, 4GL programs can be ideal to cater for this requirement, provided that it can generate correct and maintainable object codes.

The systems analyst should be able to identify situations where 4GL programming should be applied.

* Working knowledge of CASE tools (53.4%)

Whereas the basic understanding of CASE tools refers to the understanding of the philosophy behind CASE tools, the working knowledge of CASE tools requires from the systems analyst to be able to develop and implement computer systems using the CASE tool. This, of course, depends on the availability of the CASE tool product. It is obvious that the systems analyst should know how to apply CASE tools within the constraints of the SDLC.

From the bar chart it is clear that all the skills included scored a importance rating of more than 50%. It is therefore obvious that these skills should be taught to prospective systems analysts.

6.3 REGRESSION ANALYSIS

The regression analysis was based on the importance of the SDLC. Being the systematic approach to systems development, a process whereby computer systems can be developed as well as a project management tool to plan, execute and control systems development, the SDLC should be

labelled as the most important aspect of systems development. It is therefore assumed that the SDLC is so important that all other aspects of systems development are dependent on it.

6.3.1 FULL POPULATION

The regression analysis for the full population resulted in the following six aspects being the most important to be considered for systems development:

- * Planning phase

The planning phase deals with the ongoing study of the whole business to identify possible problem and enhancement areas. The ultimate aim of this phase is to provide long-term benefits for the business.

- * Implementation phase

During this phase, the detailed specifications (usually the technical specifications) are drawn up and the computer programs are constructed. All the systems development planning, analysis and design work done prior to this phase culminate in this - the building of the final product. Each and every piece of information or fact related to the system should be taken into account during this phase. The end-product of this phase, the computer-based system, is what the user will use and would therefore only then be in a position to really

evaluate the system.

* **Analysis phase**

The systems analyst has to study the current system to identify and define the real problems that triggered off the systems development project. It is of vital importance that the systems analyst familiarises himself with the system's user concerns, problems and requirements. Based on this information collected, the systems analyst can, with the involvement of the user, define the requirements in such a way that an acceptable technical specification can be drawn up from this requirements statement (the end-product of the Analysis phase). The Analysis phase is therefore regarded by many as the most important phase of the SDLC.

* **Balancing of data and process models**

This process is done using the Cross-Reference technique described in a previous section. Based on the fact that data and processes are two of the most important building blocks of any computer-based system, the balancing between the two models representing these building blocks should be regarded as very important.

- * Decision tables

This technique can be used as a vehicle to specify a set of conditions in such a way that it can form part of a very technical document such as a programming specification. On the other hand, managers can also use it to help with decision making in respect of non-technical, business-related matters. It is therefore not surprising that respondents from all positions regard this technique as important.

- * Data modelling

This technique is described in a previous section.

6.3.2 INFORMATION TECHNOLOGY AND PROJECT MANAGER POPULATION

In the regression analysis on the responses of information technology and project managers, the following six aspects were listed as the most important for systems development:

- * Design phase

The design phase is the evaluation of alternative solutions to a problem and the business specification of the selected solution. For information technology and systems development managers, this phase is important because this is really the first time that the new product can be 'sold' to the user. This is for the user the first real step towards solving his problem.

His requirements are to be transformed into something tangible
- a computer-based system.

- * Output design

This technique is described in a previous section.

- * Input design

This technique, which is closely related to output design, is also explained in the previous section.

- * Location connectivity diagrams

The definition attached to the questionnaire for this technique, was as follows:

A network modelling tool that depicts the 'shape' of a business or information system in terms of its user, data and processing locations.

It is possible that respondents could have interpreted this as a modelling tool that depicts the organization structure with additional facilities that show not only the job title, but also the processes done by the person in that job as well as the data kept at that location. If it were the correct interpretation, one can accept that this could be a very important technique for managers and analysts.

- * **Planning phase**

The Planning phase is described in the previous section. This phase is indeed important for managers as well.

- * **Systems analyst as the key facilitator of Analysis**

As mentioned in the previous section, the development of a successful system is highly dependent on the proper completion of the Analysis phase. For the manager, the key person to Analysis is the systems analyst. The manager is totally dependent on this person to ensure the validity of the requirements statement (the end-product of this phase).

6.3.3 SYSTEMS ANALYSTS AND ANALYST/PROGRAMMER POPULATION

A regression analysis was also done on the responses from systems analysts and analyst/programmers, and this analysis revealed the following six aspects to be the most important for systems development:

- * **Analysis phase**

This phase is described in a previous section. For the systems analyst, this is the most important aspect of systems analysis and design.

- * *Location connectivity diagrams*

The possible misinterpretation is referred to in the previous section.

- * *Presentations*

During the entire systems development process, the systems analyst has to present his findings, recommendation or new ideas to the users from time to time. It is therefore important to be able to present these aspects in such a way that users, managers, programmers and other technical staff understand what it entails.

- * *Planning phase*

Although this phase is also listed as important for the full population, the systems analyst is dependent on the products of this phase. These products usually determine which area of the business needs in-depth investigation which, in turn, affects the systems analyst's plan and schedule directly.

- * *CASE tool working knowledge*

Although the survey shows that the industry does not really use CASE tools very often, many systems analysts feel that a CASE tool can assist them effectively in dealing with volumes of repetitive work (such as project repository entries, drawing

of data and process models).

- * **Data Flow Diagrams**

This technique is used by the majority of businesses developing their own computer-based systems. The reason for this is that it depicts the flow of data to, from and within a system as well as the processing done on that data.

6.4 MODEL FOR EFFICIENT SYSTEMS ANALYSIS AND DESIGN TRAINING

Figure 6.4 depicts a model that highlights the most important systems development phases, skills, tools and techniques that should be included in a systems analysis and design course. It is a combination of the frequency analysis and the regression analyses on the full population (indicated by FP), the information technology and systems development managers (indicated by MP) and the systems analysts and analyst/programmers (indicated by AP). The aspects, each indicated by a variable (Xnn), are elucidated in Tables 6.1 (tools and techniques), 6.2 (SDLC phases) and 6.3 (skills).

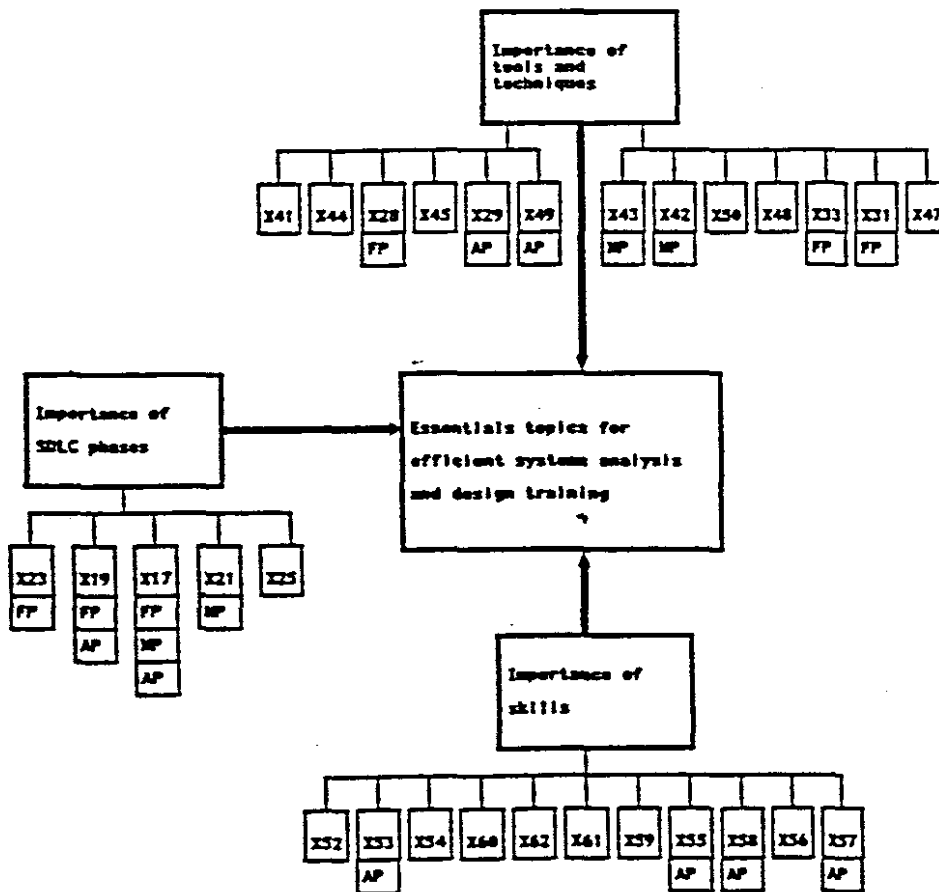


Figure 6.4: Model for efficient systems analysis and design training

VARIABLE NUMBER	VARIABLE DESCRIPTION	PERCENTAGE IMPORTANCE RATING
X(41)	Data Analysis	84.4
X(44)	Entity-Event Analysis	81.3
X(28)	Data Modelling	80.7
X(45)	Walkthroughs	79.5
X(29)	Process Modelling	78.6
X(49)	Presentations	75.9
X(43)	Output Design	75.8
X(42)	Input Design	73.2
X(50)	Project Management	71.4
X(48)	Report Writing	70.5
X(33)	Fact-Finding	69.9
X(31)	Cross Reference	63.4
X(47)	Cost Benefit Analysis	60.9

Table 6.1: Tools and Techniques

VARIABLE NUMBER	VARIABLE DESCRIPTION	PERCENTAGE IMPORTANCE RATING
X(23)	Implementation	89.3
X(19)	Analysis	88.5
X(17)	Planning	86.6
X(21)	Design	83.4
X(25)	Support	77.7

Table 6.2: SDLC Phases

VARIABLE NUMBER	VARIABLE DESCRIPTION	PERCENTAGE IMPORTANCE RATING
X(52)	Interpersonal communications	92.9
X(53)	Interpersonal relations	92.9
X(54)	General business knowledge	89.3
X(60)	Basic understanding of methodologies	78.6
X(62)	Basic understanding of prototyping	74.1
X(61)	Working knowledge of prototyping	72.3
X(59)	Working knowledge of methodologies	69.6
X(55)	3GL programming skills	61.6
X(58)	Basic understanding of CASE tools	58.0
X(56)	4GL programming skills	53.7
X(57)	Working knowledge of CASE tools	53.4

Table 6.3: Skills

The model shown in Figure 6.4 contains all the essential topics for systems analysis and design training taken separately from both the frequency and regression analysis. If only the aspects identified by both analyses as important are included in a model, the result will be as depicted in Figure 6.5.

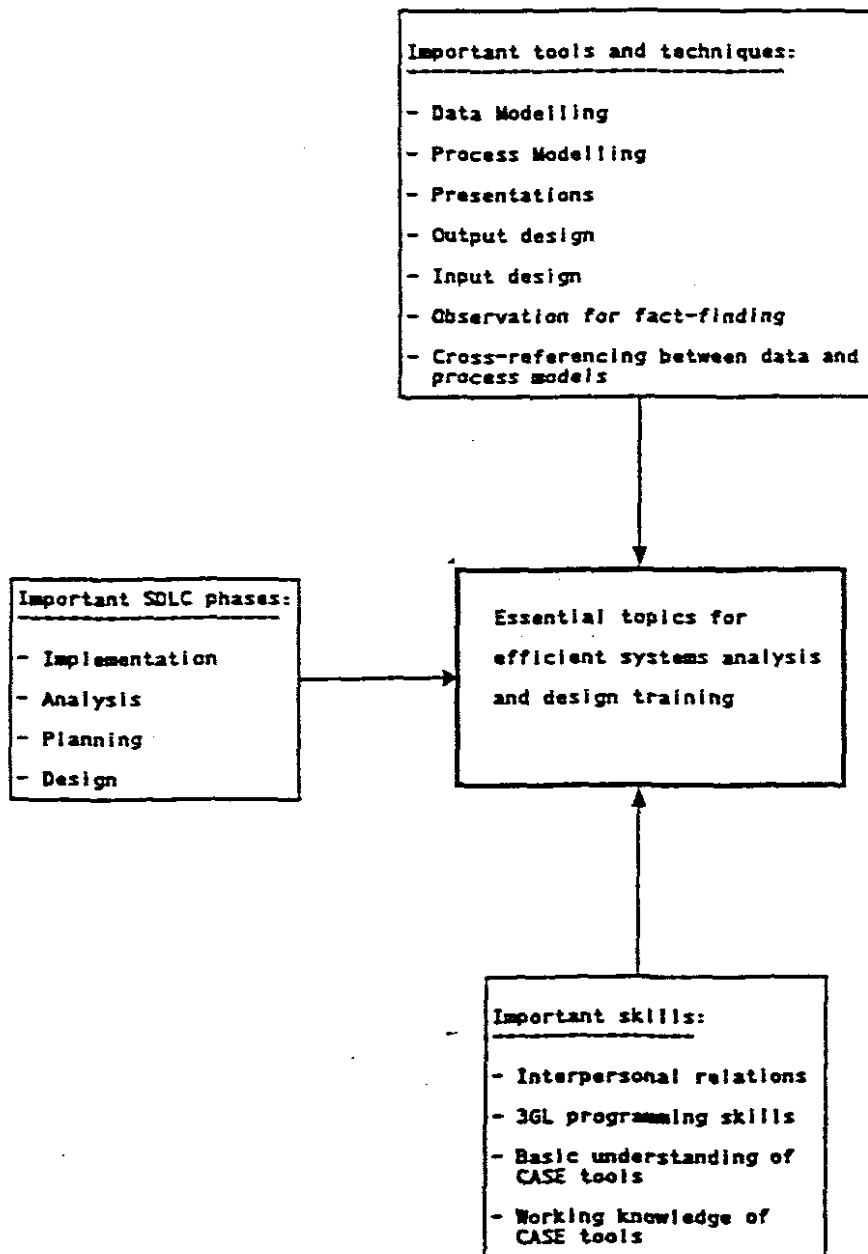


Figure 6.5: Summarised model for efficient systems analysis and design training

6.5 ADDITIONAL CONSIDERATIONS

Some additional aspects that need to be considered when compiling a course for systems analysis and design, are as follows:

- * Principles of systems development

Students should be made aware that systems are not merely developed without considering certain good systems development principles. When systems are developed, the following principles (among other) should always be considered:

- * Involve the user always
- * Approach the problem-solving systematically
- * Design the system for growth and change
- * Break the whole project up into phases and tasks
- * Schedule checkpoints throughout the process

- * Systems development problems

There are many problems that can be related to systems development. According to this survey, industry listed the following systems development problems as the most severely experienced:

- * Rate of increasing need for systems
- * Lack of communication between user and developer
- * Lack of user understanding
- * Lack of understanding systems development principles

Prospective systems analysts should be made aware that these problems can have a negative effect on their work environment and should therefore endeavour to find acceptable solutions to it.

* Textbooks

The survey done for this research, clearly emphasized the fact that the industry does not really use textbooks for reference or even in-house training. The product manuals from suppliers of various hardware and software usually replace textbooks. The systems analysis and design training official or lecturer should consider the following when prescribing a textbook:

- * Does it contain all or most of the required topics
- * Is it readily available
- * Can it be used as a complete guide to systems analysis and design.

* Industry support

According to the majority of the respondents industry is prepared to support the Technikon in so far as participating in the co-operative education program. Both parties (industry and the Technikon) benefit from it. A small group indicated that industry is also prepared to grant bursaries to students, make donations for specific purposes (such as acquisition of computer equipment) and allow an employee to give one or two lectures on specific topics to students, but industry cannot commit itself on a permanent basis. It is the lecturer

who has to identify opportunities to get industry involved.

6.6 CONCLUSIONS

The course in *Systems Analysis and Design* offered at the Cape Technikon at present, does include most of the required aspects highlighted by this research. Table 6.4 contains the current as well as the proposed course content.

The sequence in which the various topics are offered should be changed to the order as indicated in Table 6.4 under the heading Proposed Course content. The appropriate principles, tools and techniques should form part of the SDLC-phase content, or should at least be the very next topic to be covered after the phase.

Level	Current course content	Proposed course content
I	Introduction to the SDLC	Introduction to the SDLC
II	Functions of a systems analyst	Functions of a systems analyst
	Information Systems building blocks	Information Systems building blocks
	Introduction to Project Management	Systems Analysis
	PERT diagram	Process Analysis (including process modelling)
	Cost-benefit analysis	Data Analysis (including data modelling)
	Systems Analysis	IT Communications (including fact-finding and report-writing)
	Process Analysis	Feasibility analysis (including cost-benefit analysis)
III	Data Analysis	Project Management (including project management tools and techniques)
	Balancing data and process models	Systems Design
	IT Communications	Input-output design
	Project management	CASE philosophy
		Methodologies

Table 6.4: Current and proposed course content

It is also necessary to pay more attention to the following topics currently excluded from the course, or not covered in enough detail, be included or covered in much more detail than before:

* **Data modelling**

This technique should be taught to students as early as possible in the second year of Systems Analysis and Design, so as to lay a good and solid foundation for Database Management Studies in the later part of the second year.

* **Presentations**

Presentations is covered currently, but not in enough detail. This technique should be included in a separate module called Communications for Information Technologists.

* ***Input and Output design***

These two techniques are excluded from the course at the moment. These should be included in the section where Systems Design is done.

* **Interpersonal relations**

This topic should also be included in the Communications

module.

* CASE tools

The principles of CASE tools should be covered during Analysis or Design. It is, however, a problem to get students to work physically on CASE tools due to the cost of the CASE tools themselves. This could be an area where the industry can play *a major role in making their equipment, time and locations available to the students.* It will unfortunately not be easy to accommodate 150 to 200 students simultaneously.

APPENDICES

APPENDIX A: Case study: Textbook Inventory System

APPENDIX B: Questionnaire

APPENDIX C: Results of frequency analysis

APPENDIX A**CASE STUDY: TEXTBOOK INVENTORY SYSTEM**

(adapted from Whitten, 1994:397)

The purpose of the Textbook inventory system at a campus bookstore is to supply textbooks to students for classes at a local technikon. The technikon's academic departments submit initial data about courses, instructors, textbooks and projected enrolments to the bookstore on a textbook master list. This list is examined and validated by the order clerk and stored in a filing cabinet.

From the textbook master list, the order clerk then generates purchase orders which are sent to suppliers. Copies of the orders are filed in the Order file.

Book orders arrive at the bookstore accompanied by packing slips, which are checked and verified with the copies of the orders by the receiving clerk. If the ordered number of books agrees with the number of books received, the packing slips are filed with the copies of the orders.

Students fill out a book request that includes course information and hand it over to the sales clerk. The sales clerk checks if the books are in stock and writes out an invoice which is given to the student together with the books. Copies of the invoices are filed in the Invoice file.

When students pay for their books, the students are given a sales receipt and the Invoice file is updated with the payment by the accounts clerk.

This system excludes processes performed by students, suppliers and the academic departments.

Data Flow Diagram

For the Gane-Sarson and the De Marco-Yourdon method, the following steps are required:

1. List the processes with their respective processors:
 1. Order clerk: Examine and validate textbook master list
 2. Order clerk: Generate purchase order
 3. Receiving clerk: Check and verify packing slips and book orders
 4. Sales clerk: Check stock and write out invoice
 5. Accounts clerk: Record payment

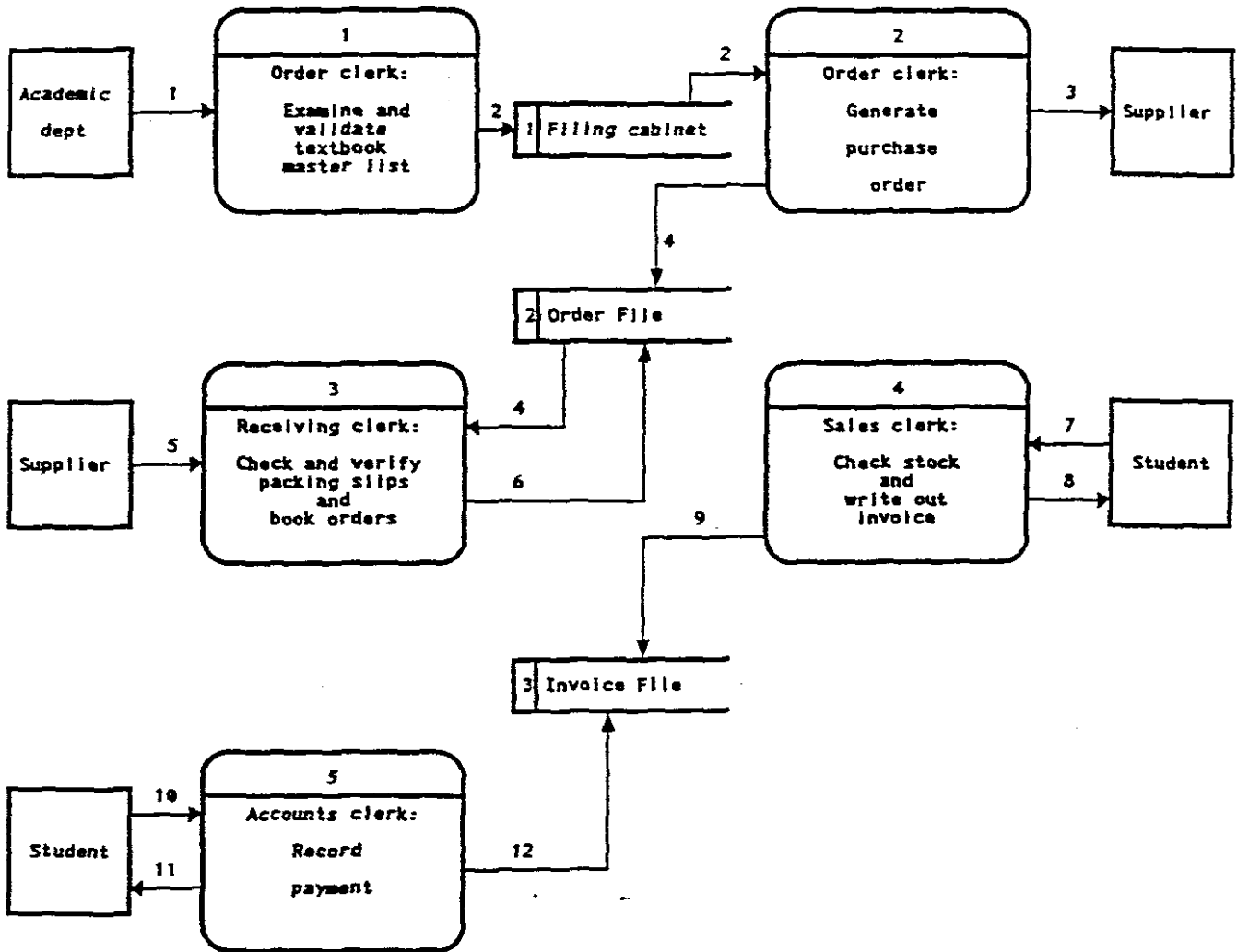
2. List the data stores:
 1. Filing cabinet
 2. Order File
 3. Invoice File

3. List the external entities:
 1. Student
 2. Academic department
 3. Supplier

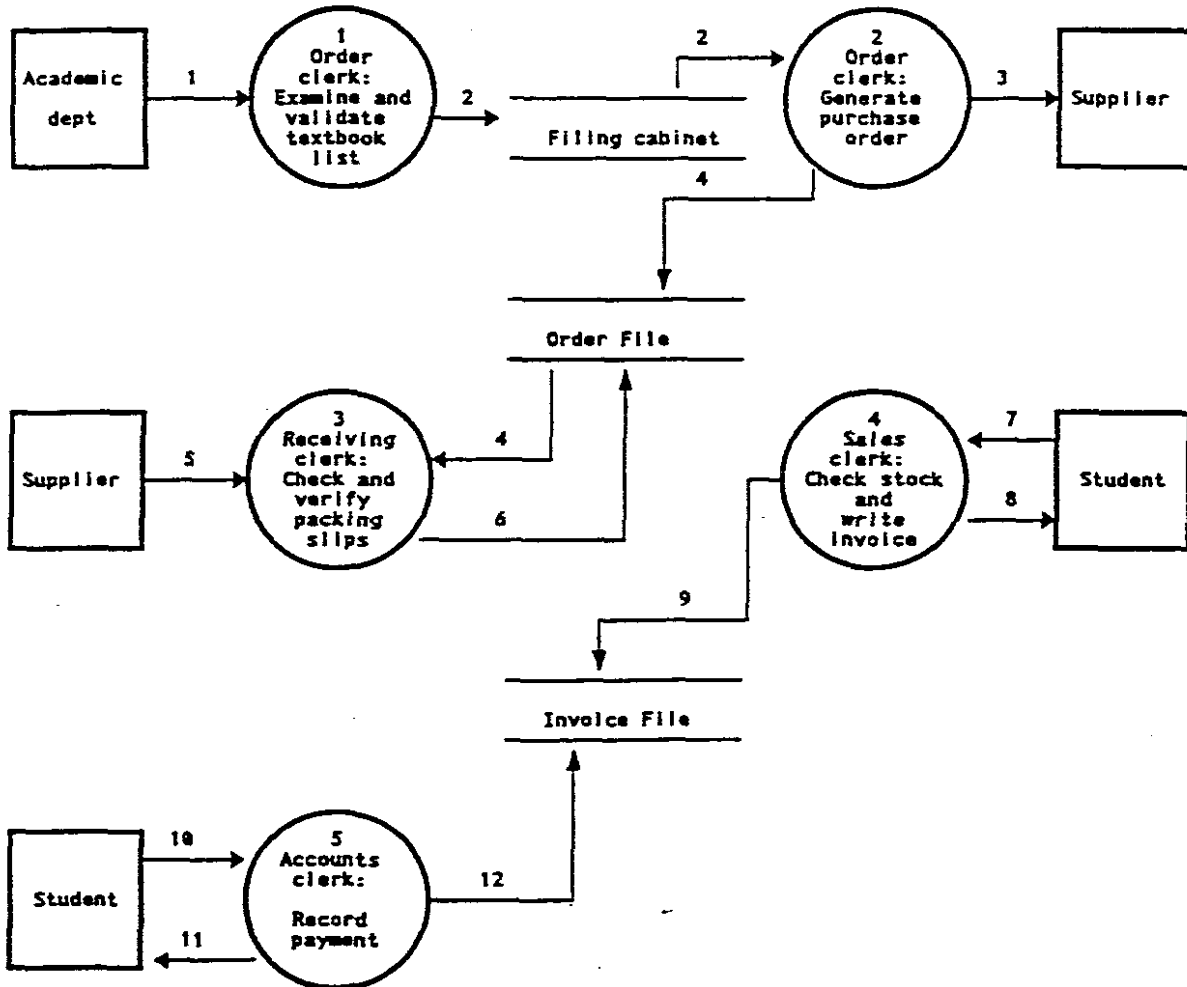
4. Draw the Data Flow Diagram (DFD)
Refer the next two pages

5. List the data flows (referenced by the allotted number on the DFD)
 1. Textbook master list
 2. Validated textbook master list
 3. Purchase order
 4. Copy of purchase order
 5. Book order and packing slip
 6. Copy of purchase order plus packing slip
 7. Book request
 8. Books and invoice
 9. Copy of invoice
 10. Payment
 11. Sales receipt
 12. Payment notification

Data Flow Diagram: Gane and Sarson symbol set



Data Flow Diagram: De Marco/Yourdon symbol set



Data Flow Diagram: SSADM method

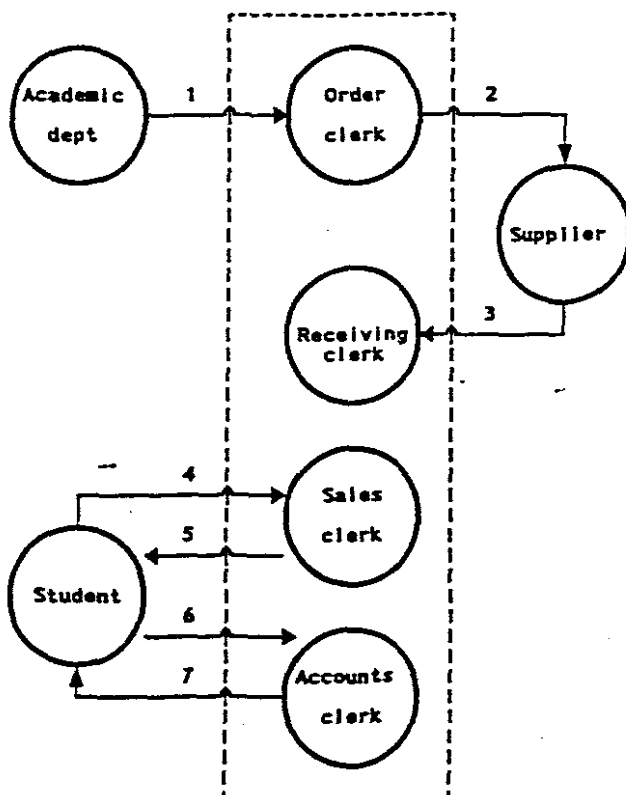
STEP 1:

List the key documents with their respective sources and recipients

No of document	Source of document	Document	Recipient of document
1	Academic dept	Textbook master list	Order clerk
2	Order clerk	Purchase order	Supplier
3	Supplier	Packing slip	Receiving clerk
4	Student	Book request	Sales clerk
5	Sales clerk	Invoice	Student
6	Student	Payment	Accounts clerk
7	Accounts clerk	Sales receipt	Student

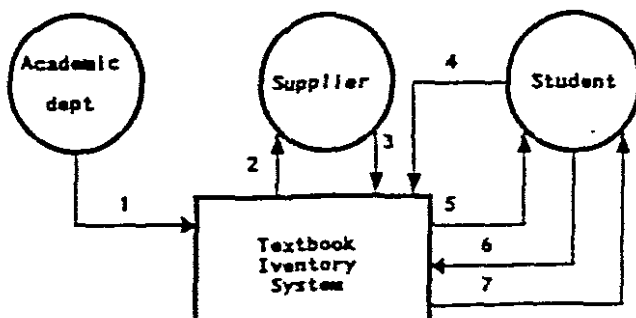
STEP 2:

Draw the physical document flow diagram, and determine the boundaries of the system



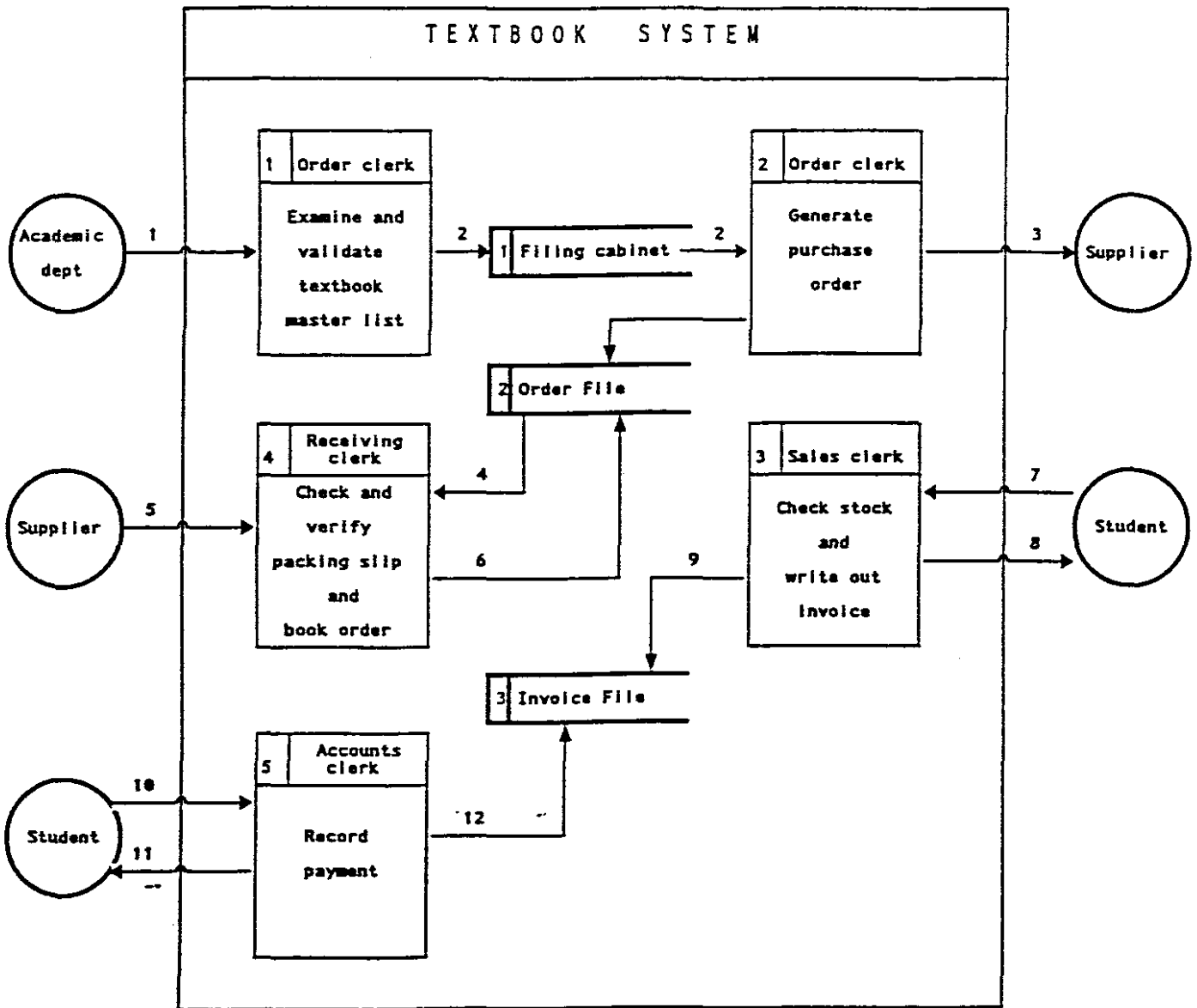
STEP 3:

Draw context (level-0) diagram



STEP 4:

Draw the level-1 Data Flow Diagram



DATA FLOWS:

1. Textbook Master List
2. Validated Textbook Master List
3. Purchase Order
4. Copy of Purchase Order
5. Book order and Packing Slip
6. Copy of Book Order and Packing Slip
7. Book Request
8. Books and Invoice
9. Copy of Invoice
10. Payment
11. Sales Receipt
12. Payment Notification

STEP 5:

Record the contents of each data store

FilingCabinet(DeptNo, DeptName, DeptAdd, DeptTel, ISBNNo, BookTitle, Author, QtyReq'd, SubjectCode, SubjectName)

OrderFile(OrdNo, OrdDate, SuppNo, SuppName, SuppAddr, ISBNNo, BookTitle, Author, QtyOrdered, DeptNo, DeptName, DeptQty)

InvoiceFile(InvNo, InvDate, StudNo, StudName, StudAdd, ISBNNo, BookTitle, QtyBought, UnitPrice, InvAmt, ReceiptNo, ReceiptDate, ReceiptAmt)

Entity Relationship Diagram

STEP 1:

Identify entities and their identifiers

Department(DeptNo)

Student(StudNo)

Order(OrdNo)

Supplier(SuppNo)

Book(ISBNNo)

Invoice(InvNo)

Payment(ReceiptNo)

STEP 2:

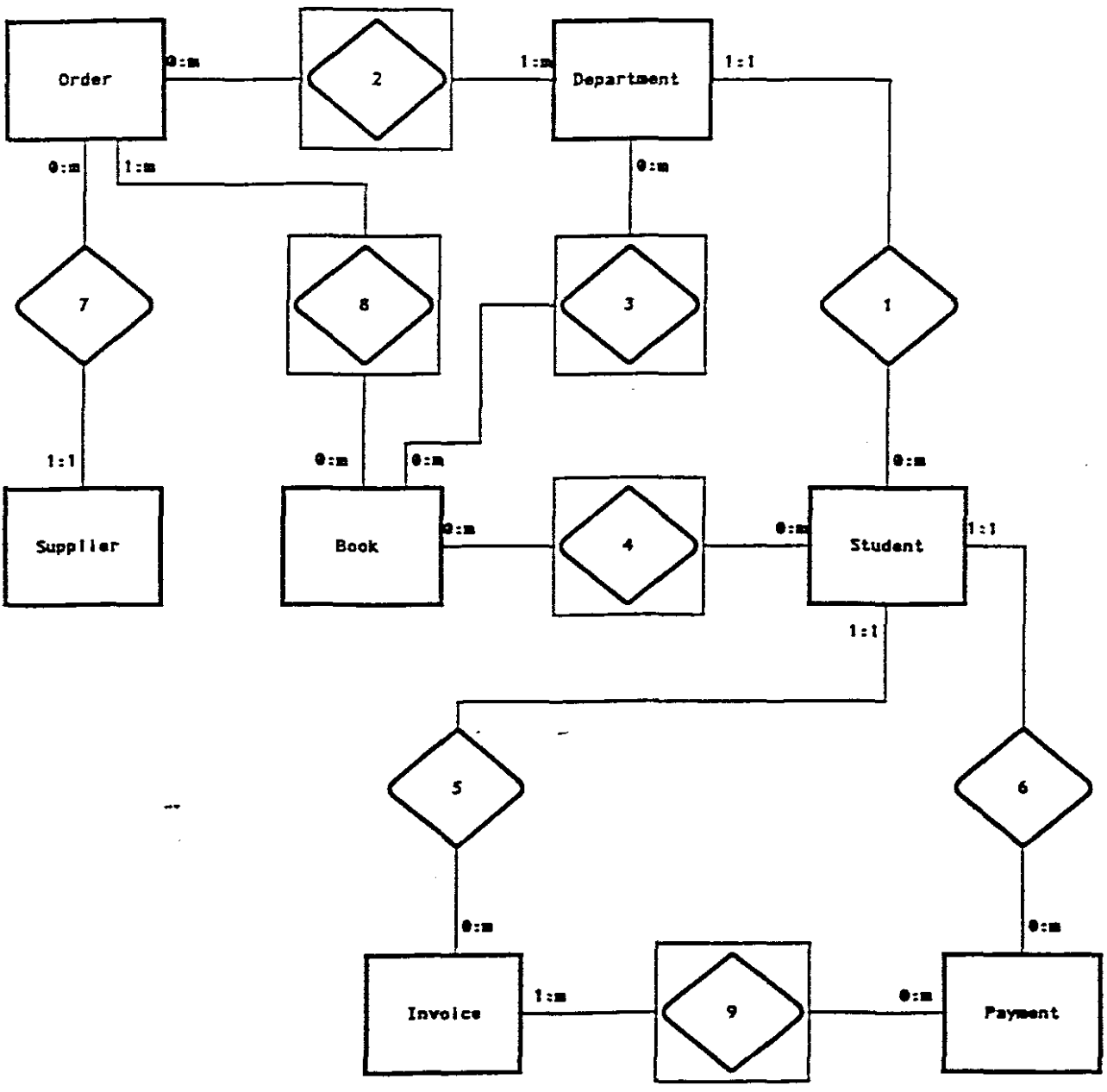
Determine relationships

	D e p a r t m e n t	S t u d e n t	O r d e r	S u p p l i e r	B o o k	I n v o i c e	P a y m e n t
Department		1a	2a		3a		
Student	1b				4a	5a	6a
Order	2b			7a	8a		
Supplier			7b				
Book	3b	4b	8b				
Invoice		5b					9a
Payment		6b				9b	

- 1a teaches zero or more
b belongs to one and only one
- 2a causes zero or more
b is generated for one or more
- 3a prescribes zero or more
b is prescribed by zero or more
- 4a requests zero or more
b is requested by zero or more
- 5a receives zero or more
b is issued to one and only one
- 6a submits zero or more
b is submitted by one and only one
- 7a is sent to one and only one
b receives zero or more
- 8a contains one or more
b appears on zero or more
- 9a results in zero or more
b is for one or more

STEP 3:

Draw Entity-Relationship diagram



Logical Data Structure

STEP 1:

Identify entities and their identifiers

Department(DeptNo)

Student(StudNo)

Order(OrdNo)

Supplier(SuppNo)

Book(ISBNNNo)

Invoice(InvNo)

Payment(ReceiptNo)

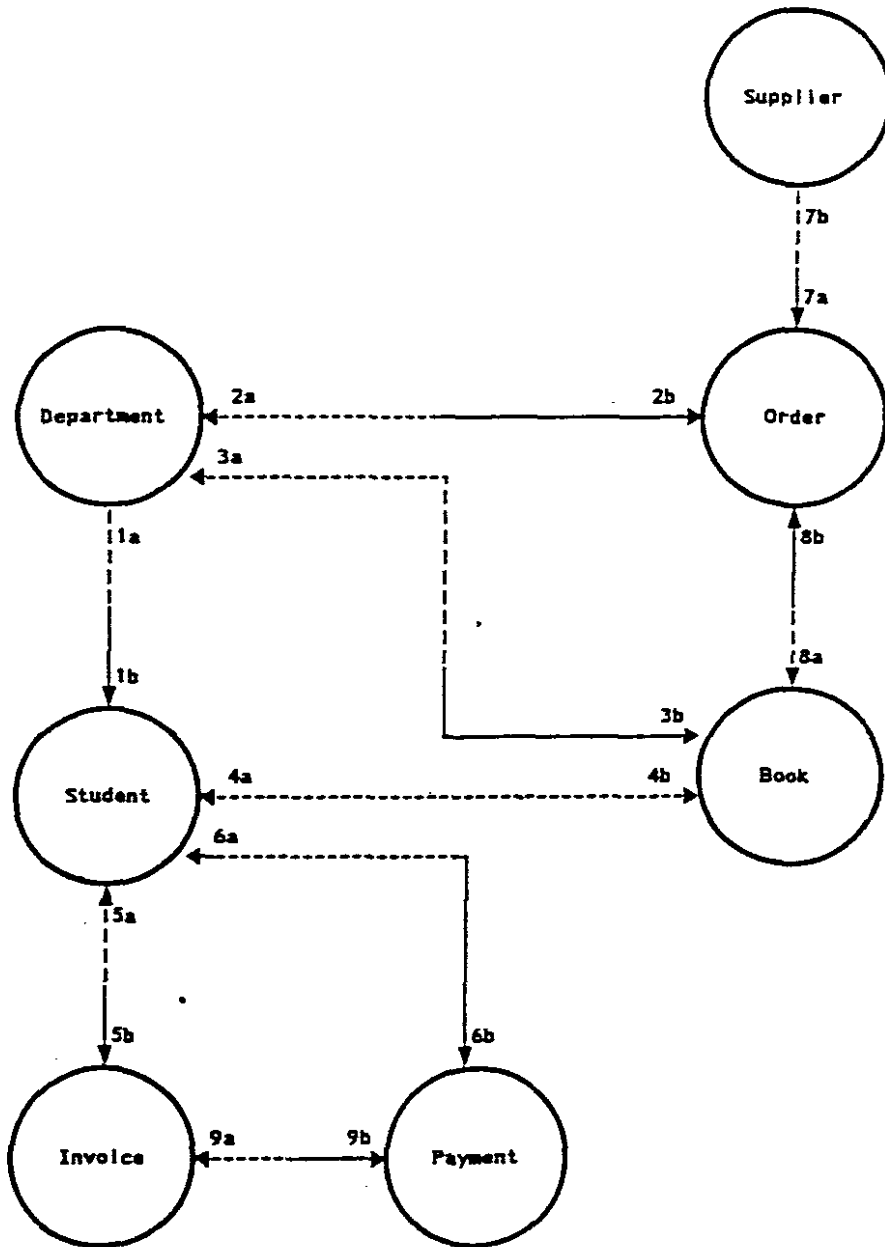
STEP 2:

Determine relationships

	D e p a r t m e n t	S t u d e n t	O r d e r	S u p p l i e r	B o o k	I n v o i c e	P a y m e n t
Department		***	***	***	***	***	***
Student	1		***	***	***	***	***
Order	2			***	***	***	***
Supplier			7		***	***	***
Book	3	4	8			***	***
Invoice		5					***
Payment	--	6				9	

- 1a Each department may have 0:m students
b Each student must belong to 1:1 department
- 2a Each department may cause 0:m orders
b Each order must be for 1:m department
- 3a Each department may prescribe 0:m books
b Each book must be prescribed by 1:m departments
- 4a Each student may order 0:m books
b Each book may be ordered by 0:m students
- 5a Each student may receive 0:m invoices
b Each invoice must be for 1:m students
- 6a Each student may submit 0:m payments
b Each payment must be made by 1:m students
- 7a Each order must be sent to 1:1 supplier
b Each supplier may receive 0:m orders
- 8a Each order must contain 1:m books
b Each book may be an item on 0:m orders
- 9a Each invoice may trigger off 0:m payments
b Each payment must be in respect of 1:m invoices

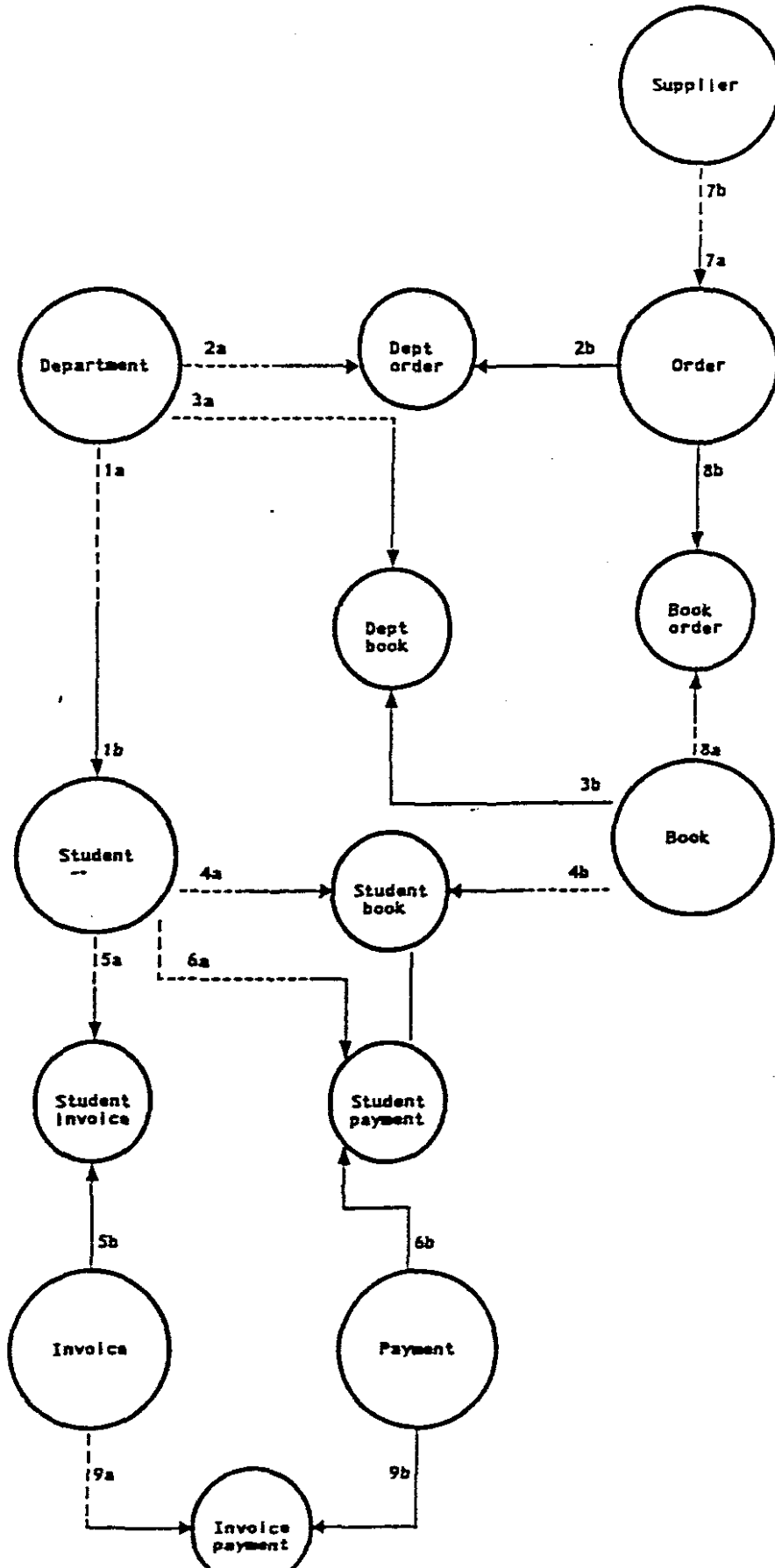
STEP 3:

Draw Draft LDS (Logical Data Structure)

Logical Data Structure

STEP 4:

Remove many-to-many relationships from the draft LDS



STEP 5:

Record all the key attributes of each entity

Department(DeptNo)

Student(StudNo)

Order(OrdNo)

Supplier(SuppNo)

Book(IsbnNo)

Invoice(InvNo)

Payment(ReceiptNo)

DeptOrder(DeptNo+OrdNo)

DeptBook(DeptNo+IsbnNo)

StudentBook(StudNo+IsbnNo)

StudentInvoice(StudNo+InvNo)

StudentPayment(StudNo+ReceiptNo)

BookOrder(IsbnNo+OrdNo)

Cross reference listing

STEP 1: List all the individual items of information stored in the datastores of the DFD.

Filing cabinet:

DeptNo, DeptName, DeptPhoneNo, BookAuthor/s, BookTitle, IsbnNo, BookQty

OrderFile:

OrderNo, OrderDate, SupplierNo, SupplierName, SupplierAddress, BookQty,
BookAuthor/s, IsbnNo, BookTitle, OrderPrice

InvoiceFile:

InvoiceNo, InvoiceDate, StudentAccountNo, StudentName, StudentAddress, BookQty,
BookAuthor, IsbnNo, BookTitle, BookPrice, AccountBalance, ReceiptNo, ReceiptDate,
ReceiptAmount.

STEP 2: List all the attributes of the entities on the LDS.

Department(DeptNo, DeptName, DeptPhoneNo
Student(StudNo, StudName, StudAddress
Order(OrdNo, OrdDate,
Supplier(SuppNo, SuppName, SuppAddress, SuppPhoneNo
Book(IsbnNo, BookTitle, BookAuthor/s, BookPublisher
Invoice(InvNo, InvDate
Payment(ReceiptNo, ReceiptDate, ReceiptAmount
DeptOrder(DeptNo, OrdNo,
DeptBook(DeptNo, IsbnNo, DeptQty
StudentBook(StudNo, IsbnNo,
StudentInvoice(StudNo, InvNo, StudInvAmount
StudentPayment(StudNo, ReceiptNo, StudPayAmount
BookOrder(IsbnNo, OrdNo, OrdQty

STEP 3: For each datastore, list all the data entities (on the LDS) containing items stored in this datastore.

Filing cabinet:

Department, Book

OrderFile:

Order, Supplier, Book

InvoiceFile:

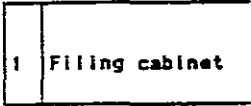
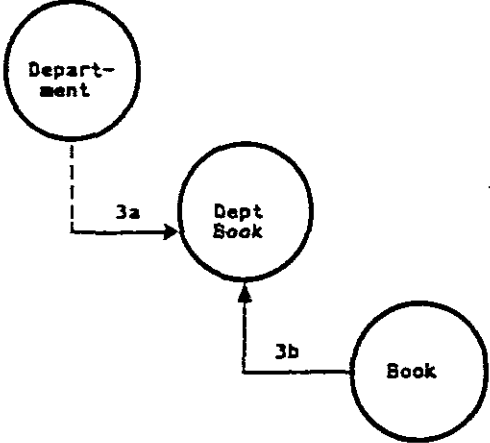

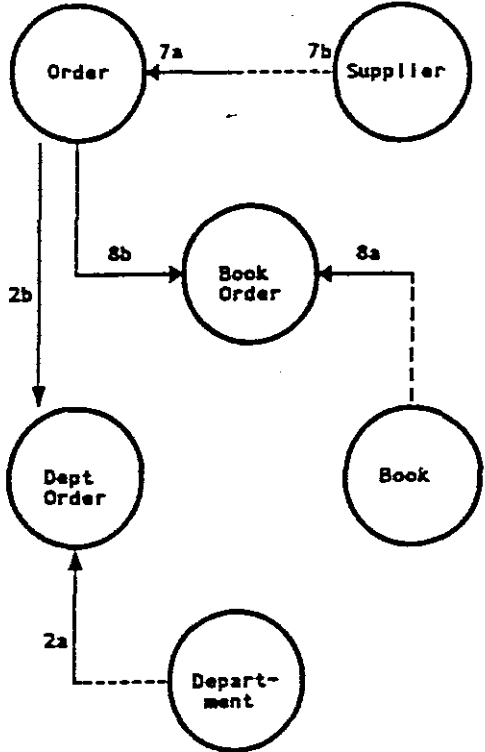
Invoice, Student, Book, Payment


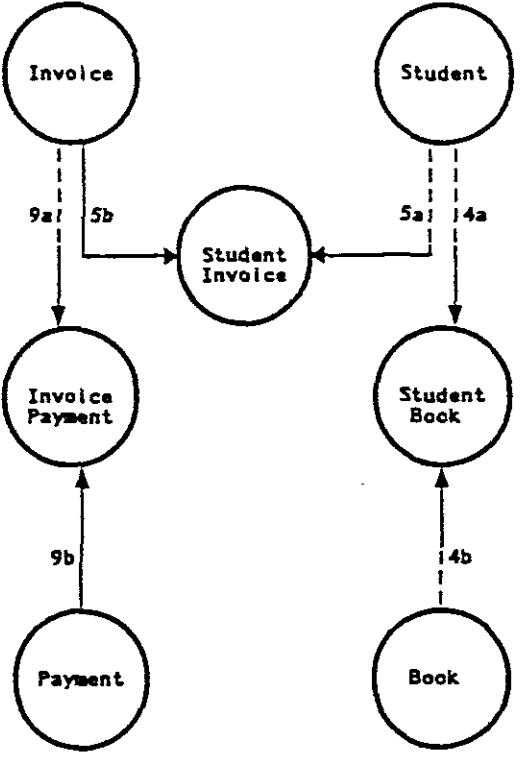
STEP 4: Draw cross-reference list (refer next page).

STEP 5: Notify the following on the CRL if encountered:

1. Any data entity (LDS) not listed on the CRL
2. Any additional data items required on data entities
3. Any superfluous data items on data entities and/or in datastores.

Cross Reference Listing

Entry number	Data Flow Diagram data stores	Logical Data Structure data entities	Notes/ Adjustments
1			<ol style="list-style-type: none"> 1. Add attribute QtyReq'd to entity DeptBook. 2. SubjectCode and SubjectName not required for the purpose of this system.
2			<ol style="list-style-type: none"> 1. Add attribute QtyOrdered to entity BookOrder 2. Add attribute DeptQty to entity DeptOrder

Entry number	Data Flow Diagram data stores	Logical Data Structure data entities	Notes/ Adjustments
3		 <pre> graph TD Invoice((Invoice)) Student((Student)) StudentInvoice((Student Invoice)) InvoicePayment((Invoice Payment)) StudentBook((Student Book)) Payment((Payment)) Book((Book)) Invoice -.-> 9a InvoicePayment Student -.-> 5a StudentInvoice Student -.-> 4a StudentBook StudentInvoice --> 5b InvoicePayment StudentBook -.-> 4b StudentBook Payment --> 9b InvoicePayment Book -.-> 4b StudentBook </pre>	<ol style="list-style-type: none"> 1. Add attribute QtyBought to entity StudentBook 2. Add attribute UnitPrice to entity Book

APPENDIX B: QUESTIONNAIRE

School of Business Informatics
CAPE TECHNIKON
P O Box 652
CAPE TOWN
8000

12 April 1996

Sir/Madam

RESEARCH: SYSTEMS DEVELOPMENT SKILLS

Please complete this questionnaire as objectively as possible, and return it within 14 days to Mr R Terblanche at the School of Business Informatics. The information is required to evaluate the current syllabus for the subject Information Systems which forms part of the National Diplomas for Information Technology and Financial Information Systems.

Please refer to annexure A containing definitions, interpretations and explanations of the terminology used in this questionnaire.

Thank you very much for your co-operation.

Dr P J S Bruwer

Please complete the following questions by making a (X) in the appropriate block, or circling the correct answer.

- 1) To which sector of the industry does your institution belong? X(1)

1	Mining		2	Business services	
3	Manufacturing		4	Hotels and Off-sales	
5	Construction		6	Commercial trading	
7	Financial services		8	Import and Export	
9	Public sector		10	Other	

If Other indicated, please specify

- 2) Does your institution usually develop its own software? X(2)
- 2.1 Yes
 2.2 No
 2.3 Develop and purchase software packages
 2.4 Contracting software development

3. Which of the following is the nearest to the description of your job title? X(3)

1	Systems analyst		2	Analyst/programmer	
3	Systems development project manager		4	Information Technology manager	
5	Systems designer		6	Software engineer	
7	Database Administrator		8	Network Administrator	
9	Programmer		10	Other	

If Other indicated, please specify

4. Do you have systems analysts reporting directly to you? X(4)
- 4.1 Yes
 4.2 No
 4.3 Sometimes

5. How many Information Technology staff members are employed by your institution? X(5)

1	1 - 20		2	21 - 50	
3	51 - 100		4	101 - 200	
5	201 - 500		6	More than 500	

6. Do you employ people with no systems development experience and/or qualifications, and then train them to become programmers? (Y/N) X(6)
7. Do you employ people with no systems development experience and/or qualifications, and then train them to become systems analysts? (Y/N) X(7)
8. What problems do you experience in systems development?
- 8.1 Lack of user understanding (Y/N) X(8)
- 8.2 Lack of understanding of systems development principles, procedures, techniques and tools (Y/N) X(9)
- 8.3 Lack of proper communication between user and developer (Y/N) X(10)
- 8.4 Rate of increasing need for systems (Y/N) X(11)
- 8.5 Too many different approaches to systems development (Y/N) X(12)
- 8.6 Too many different skills required to become a systems analyst (Y/N) X(13)
- 8.7 Very difficult to train prospective systems analysts (Y/N) X(14)
- 8.8 Other (please specify) (Y/N) X(15)

Please rate the following on a scale 1 to 7 by drawing a circle (O) around the number of your choice.

Irrelevant	Not important	Important	Of Critical Importance										
1	2	3	4	5	6	7							
How important is:													
1.	The development of computer-based systems according to the Systems Development Life Cycle (SDLC).					1	2	3	4	5	6	7	X(16)
2.	Systems Planning as part of the SDLC.					1	2	3	4	5	6	7	X(17)
3.	The systems analyst as key facilitator of Systems Planning.					1	2	3	4	5	6	7	X(18)
4.	Systems Analysis as part of the SDLC.					1	2	3	4	5	6	7	X(19)
5.	The systems analyst as key facilitator of Systems Analysis.					1	2	3	4	5	6	7	X(20)
6.	Systems Design as part of the SDLC.					1	2	3	4	5	6	7	X(21)
7.	The systems analyst as key facilitator of Systems Design.					1	2	3	4	5	6	7	X(22)
8.	Systems Implementation as part of the SDLC.					1	2	3	4	5	6	7	X(23)
9.	The systems analyst as key facilitator of Systems Implementation.					1	2	3	4	5	6	7	X(24)
10.	Systems Support must form part of the SDLC.					1	2	3	4	5	6	7	X(25)
11.	The systems analyst as key facilitator of Systems Support.					1	2	3	4	5	6	7	X(26)

Please rate the following on a scale 1 to 7 by drawing a circle (O) around the number of your choice.

Irrelevant	Not important		Important		Of Critical Importance								
1	2	3	4	5	6	7							
How important is it that the systems analyst must be able to apply the following technique/s in Systems Development:													
12.	Data Flow Diagrams (process modelling) in Systems Analysis.					1	2	3	4	5	6	7	X(27)
13.	Entity Relationship Diagrams (data modelling) in Systems Analysis.					1	2	3	4	5	6	7	X(28)
14.	Data Flow Diagrams in Systems Design.					1	2	3	4	5	6	7	X(29)
15.	Entity Relationship Diagrams in Systems Design.					1	2	3	4	5	6	7	X(30)
16.	The balancing of data and process models using cross-referencing.					1	2	3	4	5	6	7	X(31)
17.	Interviewing for fact-finding purposes.					1	2	3	4	5	6	7	X(32)
18.	Observation for fact-finding purposes.					1	2	3	4	5	6	7	X(33)
19.	Questionnaires for fact-finding purposes.					1	2	3	4	5	6	7	X(34)

Please rate the following on a scale 1 to 7 by drawing a circle (O) around the number of your choice.

Irrelevant	Not important	Important	Of Critical Importance										
1	2	3	4	5	6	7							
<p>How important is it that the systems analyst must be able to apply the following technique/s in Systems Development:</p>													
20.	Joint Application Design for fact-finding purposes.					1	2	3	4	5	6	7	X(35)
21.	Other fact-finding techniques (please specify):												X(36)
					1	2	3	4	5	6	7	
					1	2	3	4	5	6	7	
					1	2	3	4	5	6	7	
					1	2	3	4	5	6	7	
					1	2	3	4	5	6	7	
22.	Network modelling.					1	2	3	4	5	6	7	X(37)
23.	Location connectivity diagrams for network modelling purposes.					1	2	3	4	5	6	7	X(38)
24.	Decision trees for the purpose of documenting conditions and related actions.					1	2	3	4	5	6	7	X(39)
25.	Decision tables for the purpose of documenting conditions, rules and related actions.					1	2	3	4	5	6	7	X(40)
26.	Data analysis such as normalization.					1	2	3	4	5	6	7	X(41)

Please rate the following on a scale 1 to 7 by drawing a circle (O) around the number of your choice.

Irrelevant	Not important	Important	Of Critical Importance			
1	2	3	4	5	6	7
How important is it that the systems analyst must be able to apply the following technique/s in Systems Development:						
27.	Input design.					1 2 3 4 5 6 7 X(42)
28.	Output design.					1 2 3 4 5 6 7 X(43)
29.	Process Analysis (Entity-event analysis).					1 2 3 4 5 6 7 X(44)
30.	Walkthroughs.					1 2 3 4 5 6 7 X(45)
32.	PERT charts.					1 2 3 4 5 6 7 X(46)
33.	Cost-benefit analysis.					1 2 3 4 5 6 7 X(47)
36.	Report-writing.					1 2 3 4 5 6 7 X(48)
38.	Presentations.					1 2 3 4 5 6 7 X(49)
39.	Project management.					1 2 3 4 5 6 7 X(50)
40.	Other technique/s: (please specify)					X(51)
					1 2 3 4 5 6 7
					1 2 3 4 5 6 7
					1 2 3 4 5 6 7
					1 2 3 4 5 6 7

Please rate the following on a scale 1 to 7 by drawing a circle (O) around the number of your choice.

Irrelevant	Not important		Important		Of Critical Importance		
1	2	3	4	5	6	7	
How important are the following skills to become a successful systems analyst							
41.	Interpersonal communication.					1	2 3 4 5 6 7 X(52)
42.	Interpersonal relations.					1	2 3 4 5 6 7 X(53)
43.	General business knowledge.					1	2 3 4 5 6 7 X(54)
44.	Computer Programming in a high-level programming language.					1	2 3 4 5 6 7 X(55)
45.	Computer programming in applications generators or fourth generation language.					1	2 3 4 5 6 7 X(56)
46.	Working knowledge of CASE tools.					1	2 3 4 5 6 7 X(57)
47.	Understanding of the basic principles of CASE tools.					1	2 3 4 5 6 7 X(58)
48.	Working knowledge of commercial methodologies.					1	2 3 4 5 6 7 X(59)
49.	Understanding of the basic principles of methodologies.					1	2 3 4 5 6 7 X(60)
50.	Working knowledge of prototyping.					1	2 3 4 5 6 7 X(61)

Please rate the following on a scale 1 to 7 by drawing a circle (O) around the number of your choice.

Irrelevant	Not important	Important	Of Critical Importance			
1	2	3	4	5	6	7

How important are the following skills to become a successful systems analyst

51. Understanding of the basic principles of prototyping. 1 2 3 4 5 6 7
X(62)

52. Other skill/s: (please specify) X(63)

..... 1 2 3 4 5 6 7

..... 1 2 3 4 5 6 7

..... 1 2 3 4 5 6 7

..... 1 2 3 4 5 6 7

Please rate the following on a scale 1 to 7 by drawing a circle (O) around the number of your choice.

Never	Sometimes			Mostly		Always							
1	2	3	4	5	6	7							
To what extent do you:													
53.	Document systems requirements in a project repository?					1	2	3	4	5	6	7	X(64)
54.	Use Structured English in the project repository?					1	2	3	4	5	6	7	X(65)
55.	Use CASE tools in:												
55.1	Systems Analysis					1	2	3	4	5	6	7	X(66)
55.2	Systems Design					1	2	3	4	5	6	7	X(67)
55.3	Systems Implementation					1	2	3	4	5	6	7	X(68)
56.	Use Data Flow Diagrams in:												
56.1	Systems Analysis					1	2	3	4	5	6	7	X(69)
56.2	Systems Design					1	2	3	4	5	6	7	X(70)
56.3	Systems Implementation					1	2	3	4	5	6	7	X(71)
57.	Use Entity Relationship Diagrams in:												
57.1	Systems Analysis					1	2	3	4	5	6	7	X(72)
57.2	Systems Design					1	2	3	4	5	6	7	X(73)
57.3	Systems Implementation					1	2	3	4	5	6	7	X(74)
58.	Use prototyping in:												
58.1	Systems Analysis					1	2	3	4	5	6	7	X(75)
58.2	Systems Design					1	2	3	4	5	6	7	X(76)
58.3	Systems Implementation					1	2	3	4	5	6	7	X(77)
59.	Use prototyping for the purpose of discovering or verifying user requirements?					1	2	3	4	5	6	7	X(78)

Please rate the following on a scale 1 to 7 by drawing a circle (O) around the number of your choice.

Never	Sometimes		Mostly		Always	
1	2	3	4	5	6	7

To what extent do you:

60. Use the following diagramming techniques for program design:

60.1 Jackson hierarchical diagrams 1 2 3 4 5 6 7
X(79)

60.2 Yourdon 1 2 3 4 5 6 7
X(80)

60.3 Warnier-Orr 1 2 3 4 5 6 7
X(81)

60.4 Flowcharts 1 2 3 4 5 6 7
X(82)

60.5 Nassi-Schneidermann 1 2 3 4 5 6 7
X(83)

60.6 Other (please specify) X(84)

..... 1 2 3 4 5 6 7

..... 1 2 3 4 5 6 7

..... 1 2 3 4 5 6 7

..... 1 2 3 4 5 6 7

61. Which of the following programming languages do you use in systems development:

1	COBOL	X(85)		2	RPGII	X(86)	
3	ASSEMBLER	X(87)		4	FORTRAN	X(88)	
5	ADA	X(89)		6	C	X(90)	
7	C++	X(91)		8	PASCAL	X(92)	
9	BASIC	X(93)		10	ALGOL	X(94)	
11	MAGIC	X(95)		12	OTHER	X(96)	

If OTHER indicated, please specify:

.....
.....
.....

Please note that the following abbreviations for the various phases of the Systems Development Life Cycle are used in questions 62 to 64:

Systems Planning: Plan
 Systems Analysis: Anal
 Systems Design: Dsgn
 Systems Implementation: Impl
 Systems Support: Supp

62. Which of the following methodologies do you use in the various phases of the systems development life cycle:

	Methodology	Plan	Anal	Dsgn	Impl	Supp
1	STRADIS X(97)					
2	SSADM X(98)					
3	ISAC X(99)					
4	SSM X(100)					
5	Infomet X(101)					
6	IE X(102)					
7	JSD X(103)					
8	ETHICS X(104)					
9	Multiview X(105)					
10	Other X(106)					

If Other indicated, please specify:

	Methodology	Plan	Anal	Dsgn	Impl	Supp
1						
2						
3						
4						

63. Which of the following CASE tools do you use in the various phases of the systems development life cycle:

	CASE tool	Plan	Anal	Dsgn	Impl	Sup
1	ADW X(107)					
2	Systems Architect X(108)					
3	IE X(109)					
4	Visible Analyst X(110)					
5	DevelopMate X(111)					
6	PC/Prism X(112)					
7	APS X(113)					
8	CSP X(114)					
9	EASEL X(115)					
10	ObjectView X(116)					
11	PowerBuilder X(117)					
12	Bachman X(118)					
13	Excelerator X(119)					
14	Synon X(120)					
15	Telon X(121)					
16	COBOL (CICS) X(122)					
17	Flashpoint X(123)					
18	firstCASE X(124)					
19	HyperAnalyst X(125)					
20	ProjectBridge X(126)					
21	Other X(127)					
22	Do not use CASE X(128)					

If Other indicated, please specify:

	CASE tool	Plan	Anal	Dsgn	Impl	Supp
1						
2						
3						
4						

64. Which of the following prototyping languages and/or tools do you use in the various phases of the systems development life cycle:

	Prototyping tool	Plan	Anal	Dsgn	Impl	Sup
1	ADS/ON-LINE X(129)					
2	dBASE IV X(130)					
3	IDEAL X(131)					
4	INTELLECT X(132)					
5	NOMAD2 X(133)					
6	POWERBUILDER X(134)					
7	SAS X(135)					
8	TELON X(136)					
9	CLARION X(137)					
10	Paradox X(138)					
11	CLIPPER X(139)					
12	FOCUS X(140)					
13	NATURAL X(141)					
14	MANTIS X(142)					
15	ObjectView X(143)					
16	RAMIS X(144)					
17	VISUAL BASIC X(145)					
18	Progress X(146)					
19	R:BASE X(147)					
20	Other X(148)					
21	Do not use prototyping X(149)					

If Other indicated, please specify:

	Prototyping tool	Plan	Anal	Dsgn	Impl	Supp
1						
2						
3						
4						

65. Do you use a microcomputer for prototyping? X(150)
- Always
Almost always
Often
Sometimes
Rarely
Never
66. Do you use an object-oriented approach to systems development? (Y/N) X(151)
67. Have you already developed and implemented any information systems using an object-oriented design approach successfully? (Y/N) X(152)
68. If your answer to question 67 is Y, have you used an object-oriented methodology? (Y/N) X(153)
69. If your answer to question 68 is Y, which methodology/methodologies have you used?
.....
.....
.....

70. Have you employed students from the Technikon before
- 70.1 As part of the co-operative education program (Y/N) X(154)
- 70.2 On a temporary basis (Y/N) X(155)
- 70.3 On a permanent basis (Y/N) X(156)

71. If your answer to anyone of the categories in question 70 is Y, please indicate the post in which the students were employed:

.....

.....

.....

72. In which of the following areas do you have the biggest need for staff? Indicate at most three in order of urgency.

X(157)

X(158)

X(159)

Application programmers	1	Network specialists	2
Software programmers	3	Database Administrators	4
Analyst/programmers	5	Computer operators	6
Systems analysts	7	Project managers	8
Network support assistants	9	Other	10

If other indicated, please specify:

.....

.....

.....

.....

[If respondent has not indicate any need, enter 11]

73. Do you use textbooks in Systems Analysis and Design for reference purposes? (Y/N) X(160)

74. If your answer in question 73 is Y, please list the books you use in order of popularity and relevancy.

Author/s
Title
Publisher

Author/s
Title
Publisher

Author/s
Title
Publisher

Author/s
Title
Publisher

75. In which of the following areas is it possible for you to assist the Technikon in training prospective information technologists:

- Equipment
- Expertise
- Finance
- Real life projects
- Other (please specify)
.....
.....
.....

76. If you can assist the Technikon in one of the above areas, please indicate the contact person and telephone number.

.....
.....

Applications generator (also called fourth generation language)

An applications generator automatically generates one or more programs from analysis and design specifications. Example of applications generators are Focus, Ideal, Mantis, Natural, Clarion, Magic, Objectview, Powerbuilder, RAMIS, SAS, Powerhouse, CSP, etc.

CASE (computer-aided systems engineering)

The application of computer technology to systems development activities, techniques and methodologies. CASE tools are programs (software) that automate or support one or more phases of the systems development life cycle.

Commercial methodology

"Off-the-shelf", purchased methodologies (step-by-step procedures, individual and group roles, tools, deliverables and quality standards for completing the systems development life cycle.

Cost-benefit analysis

A measure of the cost-effectiveness of a project or solution. Also referred to as the economic feasibility.

Data analysis

A procedure that prepares a data model for implementation as a nonredundant, flexible and adaptable database.

Data Flow Diagram

A process modelling tool that depicts the flow of data to, within and from a system, as well as the processing performed on that data by the system.

Data Modelling

A technique for organizing and documenting a system's data.

Decision table

A tabular form of presentation that specifies a set of conditions and their corresponding actions.

Decision tree

A decision tree is a graphic representation of a problem with its associated decisions, events, consequences associated with the problem, various probabilities and the expected values of the related outcomes.

Event Analysis

A technique that studies the entities of a fully normalized data model to identify business events and conditions that cause data to be created, deleted or modified.

Entity Relationship Diagram

A popular data modelling tool that depicts the associations among different categories of data within a business or information system.

Fact-finding techniques

A systematic procedure to collect information about systems, requirements and preferences.

Feasibility analysis

A process by which it is measured how beneficial or practical the development of an information system will be to an organization.

File design

The implementation of a data model defined during systems analysis into computer files, records and fields.

High-level programming language

A set of instructions to the computer in an English-like language (not abbreviations), and where a single instruction can be written to correspond to many operations at machine level. Examples of high-level programming languages are COBOL, FORTRAN, C, C++, BASIC, ADA, Pascal, etc.

Input design

The invention, description and depiction of methods to capture data and get that data into a format suitable for the computer.

Interpersonal communication skills

The ability to communicate effectively, both orally and in writing.

Interpersonal relations

The ability to work with people, to settle conflicting goals and needs from different system users, and to convince people to change procedures and methods to make the organization more effective.

Interviewing

A fact-finding technique whereby systems analysts collect information from individuals face to face.

Joint Application Design (development)

A process whereby highly structured group meetings of all system users, systems owners and analysts occur in a single room for an extended period of time.

Location connectivity diagram

A network modelling tool that depicts the "shape" of a business or information system in terms of its user, data and processing locations. (Refer network modelling).

Methodology

A comprehensive and detailed version of an entire systems development life cycle that incorporates (1) step-by-step tasks for each phase, (2) individual and group roles to be played in each task, (3) deliverables and quality standards for each task, (4) development techniques to be used for each task. Examples include STRADIS, PRIDE, SDM, Method/1, Navigator, SSADM.

Network modelling

A diagrammatic technique used to document the "shape" of a business or information system in terms of its user, data and processing locations. (Refer location connectivity diagram).

Normalization

A data analysis method (refer data analysis).

Observation

A fact-finding technique whereby the systems analyst either participates in or watches a person perform activities to learn about the system.

Output design

The invention, description and depiction of methods to present information produced by a computer system to users.

PERT (Program/Project Evaluation Review Technique) charts

A graphical model that depicts a project as a series of events and milestones that are dependent on one another.

Presentations

Special meetings used to sell new ideas and gain approval for new systems.

Process Analysis and Design

An analytical technique for transforming essential data stores and processes into implementation locations, processes and data stores.

Process Modelling

A software engineering technique for organizing and documenting a system's processes, inputs, outputs and data stores.

Project management

The process of planning, directing and controlling the development of an acceptable system at a minimum cost in a specified time frame.

Prototyping

A technique of building a small-scale, working model of the user's requirements for purposes of discovering or verifying the user's requirements. It is also presented as a technique for systems design and implementation. The inclusion of systems implementation is based on the possibility that a prototype can evolve into a production system.

Questionnaire

Special-purpose documents that allow the analyst to collect information and opinions from respondents.

Report-writing

A technique or skill to communicate information about the systems development project to management, users and technical and/or professional staff in writing.

Structured English

A procedure specification tool adhering to the following rules:

- * Use clear unambiguous English words
- * Use strong imperative verbs
- * Avoid compound sentences
- * Avoid undefined adjectives and adverbs

Systems Analysis

The study of the problem environment and the subsequent definition and prioritization of the requirements for solving the problem.

Systems analyst

A person that studies the problems and needs of the business to determine how people, data, processes, communications (networks) and technology can best accomplish improvements for the business.

Systems Design

The evaluation of alternative problem solutions and the detailed specification of the final solution.

Systems Development Life Cycle

In the first place, it is a systematic and orderly approach to solving business problems, and developing and supporting resulting information systems. Secondly, it is a process by which systems analysts, software engineers, programmers and end-users build information systems and computer applications. Finally, it is a project management tool to plan, execute and control systems development projects.

Systems documentation study

It is a fact-finding technique whereby the systems analyst studies existing documentation, forms and files in order to collect information about a system.

Systems Implementation

The construction or assembly of the problem solution culminating in a new environment based on the solution.

Systems Planning

The ongoing study of the problem environment to identify problem-solving possibilities. Ideally, the projects that are selected provide the greatest long-term benefit to the business.

Systems Support

The ongoing maintenance and enhancement of the solution during its lifetime.

User

Any person, section and/or entity that uses or benefits from a computer system.

User-interface design

The invention, description and depiction of methods by which users of a computer system can communicate directly with a system by means of computer terminals. It is a specification of a conversation between system user and computer.

Walkthroughs

It is a peer group review to verify the validity and correctness of a particular systems development project's documentation.

APPENDIX C: RESULTS OF FREQUENCY ANALYSIS

X(1)	* Sector of the industry to which the institution belongs							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	2.	18	16.1	16.1	7.	38	33.9	69.6
	3.	7	6.2	22.3	9.	12	10.7	80.4
	5.	1	0.9	23.2	10.	22	19.6	100.0
	6.	14	12.5	35.7				
X(2)	* Institution develops own software							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	45	40.2	40.2	3.	59	52.7	98.2
	2.	6	5.4	45.5	4.	2	1.8	100.0
X(3)	* Job title of respondent							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	16	14.3	14.3	6.	5	4.5	75.9
	2.	21	18.8	33.0	7.	4	3.6	79.5
	3.	25	22.3	55.4	8.	1	0.9	80.4
	4.	16	14.3	69.6	9.	14	12.5	92.9
	5.	2	1.8	71.4	10.	8	7.1	100.0
X(4)	* Respondent has systems analysts reporting to him/her							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	39	34.8	34.8	3.	14	12.5	100.0
	2.	59	52.7	87.5				
X(5)	* The number of IT staff in the institution							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	18	16.1	16.1	4.	33	29.5	71.4
	2.	15	13.4	29.5	5.	20	17.9	89.3
	3.	14	12.5	42.0	6.	12	10.7	100.0
X(6)	* The institution train programmers							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	51	45.5	45.5	2.	61	54.5	100.0
X(7)	* The institution train systems analysts							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	32	28.6	28.6	2.	80	71.4	100.0
X(8)	* Lack of user understanding a problem							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	73	65.2	65.2	2.	39	34.8	100.0
X(9)	* Lack of understanding of systems development principles							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	57	50.9	50.9	2.	55	49.1	100.0
X(10)	* Lack of proper communication between user and developer							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	80	71.4	71.4	2.	32	28.6	100.0
X(11)	* Rate of increasing need for systems a problem							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	81	72.3	72.3	2.	31	27.7	100.0

X(12) * Too many different approaches to systems development

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	37	33.0	33.0	2.	75	67.0	100.0

X(13) * Too many skills required to become a systems analyst

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	26	23.2	23.2	2.	86	76.8	100.0

X(14) * Very difficult to train systems analysts

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	45	40.2	40.2	2.	67	59.8	100.0

X(15) * Any other problems in systems development

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	19	17.0	17.0	2.	93	83.0	100.0

X(16) * How important is the SDLC

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	1	0.9	0.9	5.	26	23.2	44.6
2.	2	1.8	2.7	6.	39	34.8	79.5
3.	9	8.0	10.7	7.	23	20.5	100.0
4.	12	10.7	21.4				

* X(17) * How important is Systems Planning

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	1	0.9	0.9	5.	25	22.3	35.7
2.	1	0.9	1.8	6.	33	29.5	65.2
3.	5	4.5	6.2	7.	39	34.8	100.0
4.	8	7.1	13.4				

X(18) * How important is the systems analyst role in Planning

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	1	0.9	0.9	5.	42	37.5	64.3
2.	4	3.6	4.5	6.	28	25.0	89.3
3.	6	5.4	9.8	7.	12	10.7	100.0
4.	19	17.0	26.8				

X(19) * How important is Analysis

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	29	25.9	37.5
2.	1	0.9	2.7	6.	35	31.3	68.7
3.	3	2.7	5.4	7.	35	31.3	100.0
4.	7	6.2	11.6				

X(20) * How important is the systems analyst role in Analysis

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	1	0.9	0.9	5.	24	21.4	41.1
2.	3	2.7	3.6	6.	47	42.0	83.0
3.	6	5.4	8.9	7.	19	17.0	100.0
4.	12	10.7	19.6				

X(21) * How important is Design

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	1	0.9	0.9	5.	21	18.8	30.4
3.	2	1.8	2.7	6.	39	34.8	65.2
4.	10	8.9	11.6	7.	39	34.8	100.0

X(22) * How important is the systems analyst role in Design

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	3	2.7	2.7	5.	35	31.3	59.8
2.	1	0.9	3.6	6.	32	28.6	88.4
3.	9	8.0	11.6	7.	13	11.6	100.0
4.	19	17.0	28.6				

X(23) * How important is Implementation

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
3.	1	0.9	0.9	6.	43	38.4	67.0
4.	11	9.8	10.7	7.	37	33.0	100.0
5.	20	17.9	28.6				

X(24) * How important is the systems analyst role in Implementation

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	24	21.4	67.9
2.	4	3.6	5.4	6.	29	25.9	93.7
3.	14	12.5	17.9	7.	7	6.2	100.0
4.	32	28.6	46.4				

X(25) * How important is Systems Support

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	29	25.9	48.2
2.	4	3.6	5.4	6.	29	25.9	74.1
3.	5	4.5	9.8	7.	29	25.9	100.0
4.	14	12.5	22.3				

X(26) * How important is the systems analyst role in Systems Support

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	8	7.1	7.1	5.	19	17.0	76.8
2.	12	10.7	17.9	6.	17	15.2	92.0
3.	20	17.9	35.7	7.	9	8.0	100.0
4.	27	24.1	59.8				

X(27) * How important is Process modelling in Analysis

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	6	5.4	5.4	5.	18	16.1	38.4
2.	3	2.7	8.0	6.	39	34.8	73.2
3.	7	6.2	14.3	7.	30	26.8	100.0
4.	9	8.0	22.3				

X(28) * How important is Data modelling in Analysis

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	5	4.5	4.5	5.	23	20.5	35.7
2.	2	1.8	6.2	6.	37	33.0	68.7
3.	3	2.7	8.9	7.	35	31.3	100.0
4.	7	6.2	15.2				

X(29) * How important is Process modelling in Design

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	6	5.4	5.4	5.	23	20.5	41.1
2.	2	1.8	7.1	6.	38	33.9	75.0
3.	5	4.5	11.6	7.	28	25.0	100.0
4.	10	8.9	20.5				

X(30) * How important is Data modelling in Design

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	27	24.1	37.5
2.	3	2.7	4.5	6.	38	33.9	71.4
3.	3	2.7	7.1	7.	32	28.6	100.0
4.	7	6.2	13.4				

X(31) * How important is cross referencing between data and process models

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	7	6.2	6.2	5.	26	23.2	59.8
2.	6	5.4	11.6	6.	27	24.1	83.9
3.	6	5.4	17.0	7.	18	16.1	100.0
4.	22	19.6	36.6				

X(32) * How important is interviewing for fact-finding

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	1	0.9	0.9	5.	18	16.1	27.7
2.	1	0.9	1.8	6.	32	28.6	56.2
3.	5	4.5	6.2	7.	49	43.7	100.0
4.	6	5.4	11.6				

X(33) * How important is observation for fact-finding

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	24	21.4	44.6
2.	2	1.8	3.6	6.	35	31.3	75.9
3.	4	3.6	7.1	7.	27	24.1	100.0
4.	18	16.1	23.2				

X(34) * How important is questionnaires for fact-finding

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	7	6.2	6.2	5.	21	18.8	75.0
2.	10	8.9	15.2	6.	16	14.3	89.3
3.	19	17.0	32.1	7.	12	10.7	100.0
4.	27	24.1	56.2				

X(35) * How important is JAD for fact-finding purposes

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	4	3.6	3.6	5.	28	25.0	54.5
2.	3	2.7	6.2	6.	32	28.6	83.0
3.	8	7.1	13.4	7.	19	17.0	100.0
4.	18	16.1	29.5				

X(36) * Any other fact-finding techniques?

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	22	19.6	19.6	2.	90	80.4	100.0

X(37) * How important is network modelling

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	12	10.7	10.7	5.	24	21.4	84.8
2.	6	5.4	16.1	6.	11	9.8	94.6
3.	13	11.6	27.7	7.	6	5.4	100.0
4.	40	35.7	63.4				

X(38) * How important is Location connectivity diagrams

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	11	9.8	9.8	5.	27	24.1	88.4
2.	4	3.6	13.4	6.	8	7.1	95.5
3.	21	18.8	32.1	7.	5	4.5	100.0
4.	36	32.1	64.3				

X(39) * How important is decision trees

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	8	7.1	7.1	5.	30	26.8	83.9
2.	11	9.8	17.0	6.	11	9.8	93.7
3.	14	12.5	29.5	7.	7	6.2	100.0
4.	31	27.7	57.1				

X(40) * How important is decision tables

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	9	8.0	8.0	5.	32	28.6	83.9
2.	7	6.2	14.3	6.	13	11.6	95.5
3.	16	14.3	28.6	7.	5	4.5	100.0
4.	30	26.8	55.4				

X(41) * How important is data analysis

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
2.	3	2.7	2.7	5.	22	19.6	34.8
3.	4	3.6	6.2	6.	42	37.5	72.3
4.	10	8.9	15.2	7.	31	27.7	100.0

X(42) * How important is input design

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	28	25.0	51.8
2.	4	3.6	5.4	6.	36	32.1	83.9
3.	4	3.6	8.9	7.	18	16.1	100.0
4.	20	17.9	26.8				

X(43) * How important is output design

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	26	23.2	47.3
2.	2	1.8	3.6	6.	37	33.0	80.4
3.	4	3.6	7.1	7.	22	19.6	100.0
4.	19	17.0	24.1				

X(44) * How important is entity-event analysis

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	32	28.6	47.3
2.	3	2.7	4.5	6.	30	26.8	74.1
3.	1	0.9	5.4	7.	29	25.9	100.0
4.	15	13.4	18.8				

X(45) * How important is walkthroughs

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	34	30.4	50.9
2.	3	2.7	4.5	6.	33	29.5	80.4
3.	5	4.5	8.9	7.	22	19.6	100.0
4.	13	11.6	20.5				

X(46) * How important is PERT charts

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	13	11.6	11.6	5.	23	20.5	81.2
2.	4	3.6	15.2	6.	11	9.8	91.1
3.	20	17.9	33.0	7.	10	8.9	100.0
4.	31	27.7	60.7				

X(47) * How important is cost-benefit analysis

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	10	8.9	8.9	5.	28	25.0	66.1
2.	5	4.5	13.4	6.	25	22.3	88.4
3.	10	8.9	22.3	7.	13	11.6	100.0
4.	21	18.8	41.1				

X(48) * How important is report-writing

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	37	33.0	62.5
2.	1	0.9	2.7	6.	27	24.1	86.6
3.	6	5.4	8.0	7.	15	13.4	100.0
4.	24	21.4	29.5				

X(49) * How important is presentations

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	31	27.7	51.8
2.	1	0.9	2.7	6.	35	31.3	83.0
3.	7	6.2	8.9	7.	19	17.0	100.0
4.	17	15.2	24.1				

X(50) * How important is project-management

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	22	19.6	48.2
2.	3	2.7	4.5	6.	35	31.3	79.5
3.	5	4.5	8.9	7.	23	20.5	100.0
4.	22	19.6	28.6				

X(51) * Any other techniques

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	7	6.2	6.2	2.	105	93.7	100.0

X(52) * How important is interpersonal communication

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
4.	8	7.1	7.1	6.	42	37.5	62.5
5.	20	17.9	25.0	7.	42	37.5	100.0

X(53) * How important is interpersonal relations

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
3.	1	0.9	0.9	6.	40	35.7	70.5
4.	7	6.2	7.1	7.	33	29.5	100.0
5.	31	27.7	34.8				

X(54) * How important is general business knowledge

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
4.	12	10.7	10.7	6.	40	35.7	72.3
5.	29	25.9	36.6	7.	31	27.7	100.0

X(55) * How important is programming in 3GL

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	3	2.7	2.7	5.	34	30.4	68.7
2.	1	0.9	3.6	6.	21	18.8	87.5
3.	9	8.0	11.6	7.	14	12.5	100.0
4.	30	26.8	38.4				

X(56) * How important is programming in 4GL

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	5	4.5	4.5	5.	32	28.6	75.0
2.	1	0.9	5.4	6.	18	16.1	91.1
3.	11	9.8	15.2	7.	10	8.9	100.0
4.	35	31.3	46.4				

X(57) * How important is a working knowledge of CASE tools

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	9	8.0	8.0	5.	32	28.6	73.2
2.	7	6.2	14.3	6.	19	17.0	90.2
3.	12	10.7	25.0	7.	11	9.8	100.0
4.	22	19.6	44.6				

X(58) * How important is knowledge of the basic principles of CASE

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	9	8.0	8.0	5.	25	22.3	64.3
2.	5	4.5	12.5	6.	25	22.3	86.6
3.	9	8.0	20.5	7.	15	13.4	100.0
4.	24	21.4	42.0				

X(59) * How important is the working knowledge of methodologies

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	32	28.6	58.9
2.	6	5.4	7.1	6.	27	24.1	83.0
3.	4	3.6	10.7	7.	19	17.0	100.0
4.	22	19.6	30.4				

X(60) * How important is understanding methodology principles

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	1	0.9	0.9	5.	35	31.3	52.7
2.	2	1.8	2.7	6.	27	24.1	76.8
3.	2	1.8	4.5	7.	26	23.2	100.0
4.	19	17.0	21.4				

X(61) * How important is working knowledge of prototyping

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	3	2.7	2.7	5.	38	33.9	61.6
3.	4	3.6	6.2	6.	28	25.0	86.6
4.	24	21.4	27.7	7.	15	13.4	100.0

X(62) * How important is understanding prototyping principles

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	5.	40	35.7	61.6
2.	3	2.7	4.5	6.	32	28.6	90.2
3.	5	4.5	8.9	7.	11	9.8	100.0
4.	19	17.0	25.9				

X(63) * Any other skills?

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	4	3.6	3.6	2.	108	96.4	100.0

X(64) * How much do you document requirements in a project repository

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	8	7.1	7.1	5.	25	22.3	68.7
2.	8	7.1	14.3	6.	22	19.6	88.4
3.	20	17.9	32.1	7.	13	11.6	100.0
4.	16	14.3	46.4				

X(65) * How much do you use structured English

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	15	13.4	13.4	5.	18	16.1	83.0
2.	11	9.8	23.2	6.	16	14.3	97.3
3.	22	19.6	42.9	7.	3	2.7	100.0
4.	27	24.1	67.0				

X(66) * How much do you use CASE tools in Analysis

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	41	36.6	36.6	5.	9	8.0	75.0
2.	10	8.9	45.5	6.	18	16.1	91.1
3.	11	9.8	55.4	7.	10	8.9	100.0
4.	13	11.6	67.0				

X(67) * How much do you use CASE tools in Design							
PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	39	34.8	34.8	5.	13	11.6	76.8
2.	8	7.1	42.0	6.	17	15.2	92.0
3.	12	10.7	52.7	7.	9	8.0	100.0
4.	14	12.5	65.2				

X(68) * How much do you use CASE tools in Implementation							
PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	43	38.4	38.4	5.	8	7.1	88.4
2.	11	9.8	48.2	6.	8	7.1	95.5
3.	18	16.1	64.3	7.	5	4.5	100.0
4.	19	17.0	81.2				

X(69) * How much do you use DFD in Analysis							
PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	18	16.1	16.1	5.	21	18.8	60.7
2.	4	3.6	19.6	6.	23	20.5	81.2
3.	13	11.6	31.3	7.	21	18.8	100.0
4.	12	10.7	42.0				

X(70) * How much do you use DFD in Design							
PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	19	17.0	17.0	5.	23	20.5	63.4
2.	4	3.6	20.5	6.	25	22.3	85.7
3.	15	13.4	33.9	7.	16	14.3	100.0
4.	10	8.9	42.9				

X(71) * How much do you use DFD in Implementation							
PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	29	25.9	25.9	5.	14	12.5	82.1
2.	10	8.9	34.8	6.	9	8.0	90.2
3.	19	17.0	51.8	7.	11	9.8	100.0
4.	20	17.9	69.6				

X(72) * How much do you use ERD in Analysis							
PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	10	8.9	8.9	5.	24	21.4	54.5
2.	8	7.1	16.1	6.	26	23.2	77.7
3.	7	6.2	22.3	7.	25	22.3	100.0
4.	12	10.7	33.0				

X(73) * How much do you use ERD in Design							
PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	10	8.9	8.9	5.	24	21.4	55.4
2.	6	5.4	14.3	6.	25	22.3	77.7
3.	7	6.2	20.5	7.	25	22.3	100.0
4.	15	13.4	33.9				

X(74) * How much do you use ERD in Implementation							
PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	26	23.2	23.2	5.	14	12.5	79.5
2.	11	9.8	33.0	6.	11	9.8	89.3
3.	13	11.6	44.6	7.	12	10.7	100.0
4.	25	22.3	67.0				

X(75) * How much do you use prototyping in Analysis

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	15	13.4	13.4	5.	17	15.2	73.2
2.	12	10.7	24.1	6.	18	16.1	89.3
3.	19	17.0	41.1	7.	12	10.7	100.0
4.	19	17.0	58.0				

X(76) * How much do you use prototyping in Design

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	10	8.9	8.9	5.	16	14.3	69.6
2.	10	8.9	17.9	6.	20	17.9	87.5
3.	21	18.8	36.6	7.	14	12.5	100.0
4.	21	18.8	55.4				

X(77) * How much do you use prototyping in Implementation

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	24	21.4	21.4	5.	15	13.4	80.4
2.	10	8.9	30.4	6.	11	9.8	90.2
3.	19	17.0	47.3	7.	11	9.8	100.0
4.	22	19.6	67.0				

X(78) * How much do you use prototyping for verifying requirements

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	10	8.9	8.9	5.	10	8.9	56.2
2.	7	6.2	15.2	6.	30	26.8	83.0
3.	13	11.6	26.8	7.	19	17.0	100.0
4.	23	20.5	47.3				

X(79) * How much do you use Jackson hierarchical diagrams

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	60	53.6	53.6	5.	12	10.7	89.3
2.	5	4.5	58.0	6.	8	7.1	96.4
3.	10	8.9	67.0	7.	4	3.6	100.0
4.	13	11.6	78.6				

X(80) * How much do you use Yourdon

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	83	74.1	74.1	5.	4	3.6	97.3
2.	9	8.0	82.1	6.	2	1.8	99.1
3.	7	6.2	88.4	7.	1	0.9	100.0
4.	6	5.4	93.7				

X(81) * How much do you use Warnier-Orr

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	97	86.6	86.6	4.	4	3.6	98.2
2.	6	5.4	92.0	5.	2	1.8	100.0
3.	3	2.7	94.6				

X(82) * How much do you use Flowcharts

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	33	29.5	29.5	5.	23	20.5	75.9
2.	4	3.6	33.0	6.	16	14.3	90.2
3.	11	9.8	42.9	7.	11	9.8	100.0
4.	14	12.5	55.4				

X(83) * How much do you use Nassi-Schneidermann

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	73	65.2	65.2	5.	6	5.4	93.7
2.	7	6.2	71.4	6.	4	3.6	97.3
3.	11	9.8	81.2	7.	3	2.7	100.0
4.	8	7.1	88.4				

X(84)	* Do you use other program design techniques							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	12	10.7	10.7	2.	100	89.3	100.0
X(85)	* Do you use Cobol							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	64	57.1	57.1	2.	48	42.9	100.0
X(86)	* Do you use RPGII							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	2	1.8	1.8	2.	110	98.2	100.0
X(87)	* Do you use Assembler							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	19	17.0	17.0	2.	93	83.0	100.0
X(88)	* Do you use Fortran							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	6	5.4	5.4	2.	106	94.6	100.0
X(89)	* Do you use ADA							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	1	0.9	0.9	2.	111	99.1	100.0
X(90)	* Do you use C							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	37	33.0	33.0	2.	75	67.0	100.0
X(91)	* Do you use C++							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	38	33.9	33.9	2.	74	66.1	100.0
X(92)	* Do you use Pascal							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	11	9.8	9.8	2.	101	90.2	100.0
X(93)	* Do you use Basic							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	11	9.8	9.8	2.	101	90.2	100.0
X(94)	* Do you use ALGOL							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM				
	2.	112	100.0	100.0				
X(95)	* Do you use MAGIC							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM				
	2.	112	100.0	100.0				
X(96)	* Do you use other languages							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	82	73.2	73.2	2.	30	26.8	100.0
X(97)	* Do you use STRADIS methodology							
			PERCENTS				PERCENTS	
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	2	1.8	1.8	2.	110	98.2	100.0

X(98) * Do you use SSADM methodology		PERCENTS		PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	35	31.3	31.3	2.	77	68.7	100.0

X(99) * Do you use ISAC methodology		PERCENTS		PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	2.	110	98.2	100.0

X(100) * Do you use SSM methodology		PERCENTS		PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	2	1.8	1.8	2.	110	98.2	100.0

X(101) * Do you use Infomet methodology		PERCENTS		PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	12	10.7	10.7	2.	100	89.3	100.0

X(102) * Do you use IE methodology		PERCENTS		PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	5	4.5	4.5	2.	107	95.5	100.0

X(103) * Do you use JSD methodology		PERCENTS		PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	3	2.7	2.7	2.	109	97.3	100.0

X(104) * Do you use ETHICS methodology		PERCENTS		PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	1	0.9	0.9	2.	111	99.1	100.0

X(105) * Do you use Multiview methodology		PERCENTS		PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	1	0.9	0.9	2.	111	99.1	100.0

X(106) * Do you use other methodologies		PERCENTS		PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	28	25.0	25.0	2.	84	75.0	100.0

X(107) * Do you use ADW CASE		PERCENTS		PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	4	3.6	3.6	2.	108	96.4	100.0

X(108) * Do you use Systems Architect CASE		PERCENTS		PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	13	11.6	11.6	2.	99	88.4	100.0

X(109) * Do you use IE CASE		PERCENTS		PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	3	2.7	2.7	2.	109	97.3	100.0

X(110) * Do you use Visible Analyst CASE		PERCENTS	
VALUE	COUNT	CELL	CUM
2.	112	100.0	100.0

X(111) * Do you use DevelopMate CASE		PERCENTS	
VALUE	COUNT	CELL	CUM
2.	112	100.0	100.0

X(112)	* Do you use PC/Prism CASE							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
	2.	112	100.0	100.0				
X(113)	* Do you use APS CASE							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
	2.	112	100.0	100.0				
X(114)	* Do you use CSP CASE							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
	2.	112	100.0	100.0				
X(115)	* Do you use EASEL CASE							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
	2.	112	100.0	100.0				
X(116)	* Do you use ObjectView CASE							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
	2.	112	100.0	100.0				
X(117)	* Do you use PowerBuilder CASE							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	4	3.6	3.6	2.	108	96.4	100.0
X(118)	* Do you use Bachman CASE							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	5	4.5	4.5	2.	107	95.5	100.0
X(119)	* Do you use Excelerator CASE							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
	2.	112	100.0	100.0				
X(120)	* Do you use Synon CASE							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
	2.	112	100.0	100.0				
X(121)	* Do you use Telon CASE							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
	2.	112	100.0	100.0				
X(122)	* Do you use COBOL (CICS) CASE							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	6	5.4	5.4	2.	106	94.6	100.0
X(123)	* Do you use Flashpoint CASE							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
	2.	112	100.0	100.0				
X(124)	* Do you use firstCASE							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
	2.	112	100.0	100.0				
X(125)	* Do you use HyperAnalyst CASE							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
	2.	112	100.0	100.0				

X(126)	* Do you use ProjectBridge CASE							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
		2.	112	100.0	100.0			
X(127)	* Do you use other CASE							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	34	30.4	30.4	2.	78	69.6	100.0
X(128)	* Do you not use CASE at all							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	59	52.7	52.7	2.	53	47.3	100.0
X(129)	*Do you use ADS/ON-LINE prototyping							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	5	4.5	4.5	2.	107	95.5	100.0
X(130)	* Do you use dBASE IV prototyping							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	2	1.8	1.8	2.	110	98.2	100.0
X(131)	* Do you use IDEAL prototyping							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	2	1.8	1.8	2.	110	98.2	100.0
X(132)	* Do you use INTELLECT prototyping							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
		2.	112	100.0	100.0			
X(133)	* Do you use NOMAD2 prototyping							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
		2.	112	100.0	100.0			
X(134)	* Do you use POWERBUILDER prototyping							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	7	6.2	6.2	2.	105	93.7	100.0
X(135)	* Do you use SAS prototyping							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	5	4.5	4.5	2.	107	95.5	100.0
X(136)	* Do you use Telon prototyping							
	PERCENTS							
	VALUE	COUNT	CELL	CUM				
		2.	112	100.0	100.0			
X(137)	* Do you use CLARION prototyping							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	8	7.1	7.1	2.	104	92.9	100.0
X(138)	* Do you use Paradox prototyping							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	2	1.8	1.8	2.	110	98.2	100.0
X(139)	* Do you use CLIPPER prototyping							
	PERCENTS				PERCENTS			
	VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
	1.	15	13.4	13.4	2.	97	85.6	100.0

X(140)	* Do you use FOCUS prototyping								
	PERCENTS								
VALUE	COUNT	CELL	CUM						
2.	112	100.0	100.0						
X(141)	* Do you use NATURAL prototyping								
	PERCENTS					PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM		
1.	11	9.8	9.8	2.	101	90.2	100.0		
X(142)	* Do you use MANTIS prototyping								
	PERCENTS								
VALUE	COUNT	CELL	CUM						
2.	112	100.0	100.0						
X(143)	* Do you use ObjectView prototyping								
	PERCENTS								
VALUE	COUNT	CELL	CUM						
2.	112	100.0	100.0						
X(144)	* Do you use RAMIS prototyping								
	PERCENTS								
VALUE	COUNT	CELL	CUM						
2.	112	100.0	100.0						
X(145)	* Do you use VISUAL BASIC prototyping								
	PERCENTS					PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM		
1.	30	26.8	26.8	2.	82	73.2	100.0		
X(146)	* Do you use Progress prototyping								
	PERCENTS					PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM		
1.	5	4.5	4.5	2.	107	95.5	100.0		
X(147)	* Do you use R:BASE prototyping								
	PERCENTS								
VALUE	COUNT	CELL	CUM						
2.	112	100.0	100.0						
X(148)	* Do you use other prototyping								
	PERCENTS					PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM		
1.	32	28.6	28.6	2.	80	71.4	100.0		
X(149)	* Do you not use prototyping at all								
	PERCENTS					PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM		
1.	32	28.6	28.6	2.	80	71.4	100.0		
X(150)	* Do you use a microcomputer for prototyping								
	PERCENTS					PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM		
1.	34	30.4	30.4	4.	9	8.0	61.6		
2.	18	16.1	46.4	5.	5	4.5	66.1		
3.	8	7.1	53.6	6.	38	33.9	100.0		
X(151)	* Do you use object-oriented approach to systems development								
	PERCENTS					PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM		
1.	52	46.4	46.4	2.	60	53.6	100.0		
X(152)	* Have you developed an object-oriented system yet								
	PERCENTS					PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM		
1.	37	33.0	33.0	2.	75	67.0	100.0		

X(153) * Do you use an Object Oriented methodology

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	19	17.0	17.0	2.	93	83.0	100.0

X(154) * Have you employed Technikon students as part of co-operative education

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	63	56.2	56.2	2.	49	43.7	100.0

X(155) * Have you employed Technikon graduates temporarily

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	61	54.5	54.5	2.	51	45.5	100.0

X(156) * Have you employed Technikon graduates permanently

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	68	60.7	60.7	2.	44	39.3	100.0

X(157) * In which area of IT do you need staff (priority 1)

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	39	34.8	34.8	6.	1	0.9	74.1
2.	8	7.1	42.0	7.	12	10.7	84.8
3.	10	8.9	50.9	8.	6	5.4	90.2
4.	3	2.7	53.6	10.	3	2.7	92.9
5.	22	19.6	73.2	11.	8	7.1	100.0

X(158) * In which area of IT do you need staff (priority 2)

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	4	3.6	3.6	7.	16	14.3	67.9
2.	8	7.1	10.7	8.	1	0.9	68.7
3.	8	7.1	17.9	9.	4	3.6	72.3
4.	10	8.9	26.8	11.	31	27.7	100.0
5.	30	26.8	53.6				

X(159) * In which area of IT do you need staff (priority 3)

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	6	5.4	5.4	7.	10	8.9	35.7
2.	3	2.7	8.0	8.	10	8.9	44.6
3.	5	4.5	12.5	9.	4	3.6	48.2
4.	6	5.4	17.9	10.	1	0.9	49.1
5.	10	8.9	26.8	11.	57	50.9	100.0

X(160) * Do you use textbooks in Systems Development

PERCENTS				PERCENTS			
VALUE	COUNT	CELL	CUM	VALUE	COUNT	CELL	CUM
1.	36	32.1	32.1	2.	76	67.9	100.0

SUMMARY

Systems Analysis and Design (also known as Systems Development) is the systematic process of problem identification, problem definition, analysis of the causes of the problem, the design of alternative solutions to the problem, and the eventual implementation of the selected solution as a computer system (if possible).

During this systems development process there are a quite a large number of principles and approaches that need to be considered. It is also required from the systems analyst to have certain skills and to know how to apply a number of tools and techniques.

The purpose of this research is to determine which approaches, principles, skills, tools and techniques are required by the industry for the development of commercial computer systems so that prospective systems analysts can be properly trained in those aspects. This means that the course content of training institutions should be updated accordingly.

The aspects that form part of the course offered by the training institution where the researcher works, are discussed and identified out of the literature.

The course content is compared with what is required by the industry by means of an empirical research consisting of a questionnaire and frequency and regression analysis.

The results from the research indicate that the following aspects of systems development must be emphasized:

1. Phases of systems development

- * Implementation
- * Analysis
- * Planning
- * Design

2. Important tools and techniques

- * Data modelling
- * Process modelling
- * Presentations
- * Input and output design
- * Observation (for fact-finding)
- * Cross Reference between the data and process model

3. Skills

- * Interpersonal relations
- * Third generation programming languages
- * Basic understanding of CASE
- * Working knowledge of CASE

OPSOMMING

Stelselontleding en-ontwerp (ook stelselontwikkeling genoem) is die sistematiese proses van probleem-identifisering, probleemdefinisie, ontleding van die oorsake van die probleem, die ontwerp van alternatiewe oplossings vir die probleem, en die uiteindelijke implementering van die geselekteerde oplossing as 'n rekenaarstelsel (indien moontlik).

Gedurende hierdie stelselontwikkelingsproses is daar 'n hele aantal beginsels en benaderings wat oorweeg moet word. Dit word ook van die stelselontleder vereis om sekere vaardighede te beskik en te weet hoe om 'n aantal tegnieke en hulpmiddele te gebruik.

Die doel van hierdie navorsing is om vas te stel watter benaderings, beginsels, vaardighede, tegnieke en hulpmiddels deur die industrie benodig word vir die ontwikkeling van kommersiële rekenaarstelsels sodat voornemende stelselontleders in daardie aspekte behoorlik opgelei kan word. Dit beteken dus dat die kursusinhoud van opleidingsinstansies dienooreenkomstig aangepas behoort te word.

Die aspekte wat tans deur die opleidingsinstansie waar die navorser werk, as deel van die kursus Inligtingstelsels aangebied word, word bespreek en uit die uit die literatuur geïdentifiseer.

Die kursusinhoud word vergelyk met dit wat deur die industrie benodig word

deur middel van 'n empiriese navorsing bestaande uit 'n vraelys en frekwensie- en regressieontledings.

Die resultaat van die navorsing toon dat klem gelê moet word op die volgende aspekte van stelselontwikkeling:

1. Fases van stelselontwikkeling

- * Implementering
- * Ontleding
- * Beplanning
- * Ontwerp

2. Belangrike tegnieke en hulpmiddele

- * Data modellering
- * Prosesmodellering
- * Aanbiedings
- * Toevoer- en afvoerontwerp
- * Waarneming (vir feite-versameling)
- * Kruisverwysing tussen die data en prosesmodel

3. Vaardighede

- * Interpersoonlike verhoudinge
- * Derde-generasie programmeringstale
- * Basiese begrip van CASE
- * Werkskennis van CASE

BIBLIOGRAPHY

ASHWORTH, CAROLINE. 1990. SSADM: a practical approach. London: MacGraw-Hill.

AVISON, DE, FITZGERALD, G. 1990. Information Systems Development Methodologies, Techniques and Tools. London: Blackwell Scientific Publications

BRUWER, PJS, 1991. Toepassing van 'n beperkte liniêre regressiemodel op verskillende Inligtingstelselasette, *SA Tydskrif vir Bedryfsleiding*, 22(3), 41-45

CONGER, SUE. 1994. The New Software Engineering. Belmont, California: Wadsworth Publishing Company

CUTTS, G. 1991. Structured systems analysis and design methodology. 2nd ed. Oxford: Blackwell.

DAVIS, WILLIAM S. 1983. Systems analysis and design - a structured approach. Reading, Massachusetts: Addison-Wesley.

DICKSON, WJ, BROWN, MB. 1981. Biomedical computer programs, P-series. Berkeley: University of California Press.

EDWARDS, PERRY. 1993. Systems analysis and design. Watsonville: Mitchell McGraw-Hill

EVA, MALCOLM. 1992. SSADM Version 4: A user's guide. Berkshire: McGraw-Hill

FERTUCK, LEN. 1995. System analysis and design with modern methods. Dubuque: Business and Educational Technologies

GOLDBERG, R. 1986. Software engineering: an emerging discipline. *IBM Systems Journal*, Vol 25, Nos 3/4, pp.334 to 353

GOLDSMITH, SYLVIA. 1993. A practical guide to real-time systems development. Hertfordshire: Prentice Hall International (UK) Ltd

GORDON, CARL L, NECCO, CHARLES R, TSAI, NANCY W. 1987. Toward a standard systems development life cycle. *Journal of Systems Management*. August 1987, pp. 24 to 27

HAWRYSZKIEWYCZ, I T. 1991. Introduction to systems analysis and design. 2nd ed. Sydney: Prentice Hall

KENDALL, KENNETH E, KENDALL, JULIE E. 1992. Systems analysis and design. 2nd ed. Englewood Cliffs: Prentice-Hall

KENDALL, PENNY A. 1992. Introduction to Systems Analysis and Design: a structured approach, 2nd ed. Dubuque, USA: W C Brown Publishers

MARTIN, JAMES. 1982. Application development without programmers. Englewood Cliffs, New Jersey: Prentice-Hall.

MARTIN, JAMES, MACLURE, CARMA. 1985. Diagramming techniques for analysts and programmers. Englewood Cliffs, New Jersey: Prentice-Hall

MODELL, MARTIN E. 1988. A professional's guide to systems analysis. New-York: MacGraw-Hill.

SCHACH, STEPHEN R. 1990. Software engineering. Homewood, Boston: Richard D. Irwin

SCHACH, STEPHEN R. 1992. Practical software engineering. Homewood, Boston: Richard D. Irwin

SOMMERVILLE, I. 1982. Software engineering. 2nd ed. Wokingham, England: Addison-Wesley Publishing Company

VONK, ROLAND. 1990. Prototyping: the effective use of CASE technology. Hempstead, Hertfordshire: Prentice Hall

WHITTEN, JEFFREY L, BENTLEY, LONNIE D, BARLOW, VICTOR M. 1989.

Systems analysis and design methods. 2nd ed. Homewood, Boston: Richard D. Irwin.

WHITTEN, JEFFREY L, BENTLEY, LONNIE D, BARLOW, VICTOR M. 1994.

Systems analysis and design methods. 3rd ed. Homewood, Boston: Richard D. Irwin.