A CONTINGENCY FRAMEWORK FOR INFORMATION SYSTEMS DEVELOPMENT

DAVID ERNEST AVISON

Doctor of Philosophy

THE UNIVERSITY OF ASTON IN BIRMINGHAM

October 1990

Summary

A CONTINGENCY FRAMEWORK FOR INFORMATION SYSTEMS DEVELOPMENT

DAVID ERNEST AVISON

Doctor of Philosophy

THE UNIVERSITY OF ASTON IN BIRMINGHAM

October 1990

This research concerns information systems and information systems development. The thesis describes an approach to information systems development called Multiview. This is a methodology which seeks to combine the strengths of a number of different, existing approaches in a coherent manner. Many of these approaches are radically different in terms of concepts, philosophy, assumptions, methods, techniques and tools. The research is classified as action research and three cases are described presenting Multiview 'in action'. The first is used mainly to expose the strengths and weaknesses of an early version of the approach discussed in the thesis. Tools and techniques are described in the thesis which aim to strengthen the approach. Two further cases are presented to illustrate the use of this second version of Multiview. This is not put forward as an 'ideal methodology' and the action research exposes some of the difficulties and practical problems of information systems work and the use of this methodology. A more contingency based approach to information systems development is advocated using Multiview as a framework rather than as a prescriptive tool. Each information systems project and the use of the framework is unique, contingent on the particular problem situation. The skills of different analysts, the backgrounds of users and the situations in which they are constrained to work are three crucial elements of any information systems development project. The realities of the situation will cause departure from the 'ideal methodology' in order to allow for the exigencies of the real world. Multiview is proposed as a means by which an application area can be 'explored' in order to develop an information system.

KEY WORDS:
Multiview
Information systems
Information systems development methodologies
Contingency approach
Action research.

Acknowledgements

During the last seven years, the period in which this research took place, I have written several research papers and texts with Guy Fitzgerald, now of Templeton College, Oxford, and Trevor Wood-Harper, now at the University of Salford. They have contributed to many of the ideas expressed in this thesis. Multiview itself was originally designed in raw form by Trevor Wood-Harper in 1982. I became involved in 1984, when asked to contribute to the text Information Systems Definition: the Multiview Approach (Wood-Harper, A.T., Antill, L & Avison, D.E) which was published in 1985. The first case discussed in the thesis (Chapter 4) was carried out by Lyn Antill, Trevor Wood-Harper and others at South Bank Polytechnic. It is discussed in this thesis in order to reveal the strengths and weaknesses of the early version of Multiview described in that text. The second case (Chapter 6) was carried out in the main by Paul Catchpole as part of his PhD research under my supervision. The third case (Chapter 7) is a distillation of work carried out by a number of students either as a practical project but mainly taking the third year options in the BSc Computer Science degree (also latterly BSc Management and Computer Science) called Practical Data Modelling and Practical Databases over the last five years under my leadership. Ken Bowcock, Paul Gardner and Julie Horton have also recently become involved in the teaching of these courses. I am grateful to all the above named and also the many students taking the courses mentioned. I would also like to acknowledge the support of my Head of Department, Brian Gay, which has enabled me to carry out this research. I also thank Brian, Guy Fitzgerald, Paul Golder and Hanifa Shah for commenting on an earlier draft of this thesis.

Contents

List of Figures

Chapter 1
Research Theme and Method

In this chapter we will look at the theme of the research, that is the research area, and then the research method. Finally an outline of the thesis is presented.

1.1 RESEARCH THEME

The theme of the research is in the area of information systems and methodologies to develop them. In particular the research concerns a contingency approach (called Multiview) to information systems development. By a contingency approach is meant an approach which enables the systems developer to call on the tools and techniques of the methodology as required for the particular situation.

1.1.1 Information Systems

An information system provides information about the organisation and its environment. This information, which is useful to members and clients of that organisation, could concern its customers, suppliers, products, equipment, and so on. The organisation could be, for example, a business, church, hospital, university, bank or library.

In this research, the concern is with formalised information systems. Here formalised does not mean 'mathematical', which is one interpretation of the term 'formal', in particular as used in 'formal methods' (for example, Bjorner & Jones, 1982). The term is used here to distinguish information systems discussed in this thesis from less formalised information systems such as the 'grapevine', consisting of rumour, gossip, ideas and preferences, which is also a valid information system. These informal information systems tend to be intuitive or qualitative. Organisations also need to develop formalised systems which will provide information on a regular basis and in a pre-defined manner, and these are the concern in this research.

The research is also concerned with computer-based information systems, for the computer can process data (the basic facts) speedily and accurately and provide information when and where required and at the correct level of detail. This does not mean that the information system is 'purely' a computer system - there will be many manual (or clerical) aspects - it means that part of the system is likely to use a computer.

This might be used to store data, produce reports or handle management enquiries.

Buckingham et al (1987) broaden the area and highlights the fact that although information technology is normally used, it is not necessary for information systems:

> 'An Information System is any system which assembles, stores, processes and delivers information relevant to an organisation (or to society), in such a way that information is accessible and useful to those who wish to use it, including managers, staff, clients and citizens. An Information System is a human activity (social) system which may or may not involve the use of computer systems.'

As stated previously, this is a broader definition than that used in this research. Nevertheless, the author is aware that there are societal implications of the information systems issues that are discussed and that the term 'information systems' does not necessarily imply 'computer information systems'.

Some examples of information systems might be helpful:

- A payroll system is an information system. All organisations have employees and they will normally be paid. The raw data of a payroll system includes the number of hours worked by the employees, their rates of pay, and deductions, such as tax and national insurance. The system might produce payslips, and reports for management about the payroll.
- A sales ledger system is an information system. It is a system relating to the accounts of customers. The raw data of a sales ledger system relates to sales to customers and remittances from them. The system will provide statements of any balances owing, and could produce analyses of debtors' balances according to area, sales representative and customer group.
- A project planning and control system is an information system. The raw data will include the various activities that make up the project and the range of resources that might be used to develop the project. The system schedules projects so that completion is at the earliest possible date, with the least drain on resources, and provides reports on progress during the life of the project. These reports enable management to act on projects that are behind schedule or where costs are above predictions.
- A decision-support system is an information system. The raw data includes the whole range of facts about the organisation, or part of the organisation, or sometimes relates to aspects external to the organisation, (that is, its environment). The system is designed to enable managers to retrieve information which will help them make decisions about, for example, where to build a factory, whether to

merge with competitors, which products to sell, the prices of products, and the salaries of employees.

- An airline ticket reservation system is an information system. It processes customers' requests for seats on aircraft. It may also be used to provide information regarding the take up of seats.

There have always been information systems, indeed their history extends at least as far back as 600 BC (Avison & Fitzgerald, 1991) although it is only in the recent past that they have used computers. If firms have employees, there needs to be some sort of system to pay them. If firms manufacture products, then there will be a system to order the raw materials from the suppliers and another to plan the production of the goods from the raw materials. Companies need to have a system to deal with orders from customers, another to ensure that products are transported, and yet another to send invoices to the customers and to process payments.

In the time before computers, these systems were largely manual. The word 'largely' is appropriate, because the manual workers would use adding machines, typewriters, and other mechanical or electrical aids to help the system run as efficiently as possible. The use of computers represents only an extension (though a significant extension) of this process. A solution to develop an information system which involves the use of computers may well be contemplated where the manual system has proved inadequate in some way, for example:

- Increasing workloads have overloaded the system.
- Suitable staff are expensive and difficult to recruit.
- There is a change in the type of work.
- There are frequent errors.

1.1.2 Information Systems Development Methodologies

The early applications of computers - say, until the 1960s - were implemented without the aid of an explicit information systems methodology. In these early days, the emphasis of computer applications was towards programming, and the skills of programmers were particularly appreciated. The systems developers were therefore technically trained but were not necessarily good communicators. This often meant that the needs of the users in the application area were not well established, with the consequence that the information system design was frequently inappropriate for the application.

Few programmers would follow any formal methodology. Frequently they would use rule-of-thumb and rely on experience. Estimating the date on which the system would be operational was difficult, and applications were frequently behind schedule. Programmers were usually overworked, and frequently spent a very large proportion of

their time on correcting and enhancing the applications which were operational.

Typically, a user would come to the programmers asking for a new report or a modification of one that was already supplied. Often these changes had undesirable effects on other parts of the system, which also had to be corrected. This vicious circle would continue, causing frustration to both programmers and users. This was not a methodology, it was only an attempt to survive the day.

As computers were used increasingly and management was demanding more appropriate systems for its expensive outlay, this could not go on. There were three main changes that occurred around the 1960s:

- The first was a growing appreciation of that part of the development of the system that concerns analysis and design and therefore of the role of the systems analyst as well as that of the programmer.
- The second was a realisation that as organisations were growing in size and complexity, it was desirable to move away from one-off solutions to a particular problem and towards a more integrated information system.
- The third was an appreciation of the desirability of an accepted methodology for the development of information systems.

As Utterback & Abernathy (1975) argue, innovation has three elements: improved process, improved product, and improved organisation. In the context of this research, the improved process, that is, better information systems development methodologies, should lead to improved product, that is, better information systems, and thereby improved organisation, through better decision support, for example.

Information systems development methodologies were therefore adopted by many computer data processing installations from the 1960s. An information systems development methodology is a collection of procedures, techniques, tools, and documentation aids which will help the systems developers in their efforts to implement a new information system. A methodology will consist of phases, themselves consisting of sub-phases, which will guide the systems developers in their choice of the techniques that might be appropriate at each stage of the project and also help them plan, manage, control and evaluate information systems projects.

But a methodology is more than merely a collection of these things. It is usually based on some philosophical view, otherwise it is merely a method, like a recipe. Methodologies may differ in the techniques recommended or the contents of each phase, but sometimes their differences are more fundamental. They could differ according to the philosophy on which they are based. Some methodologies emphasise the human-oriented aspects of developing an information system, others aim to be scientific in approach, others pragmatic, and others attempt to automate as much of the work of developing a project as possible. These differences may be best illustrated by

their different assumptions, stemming from their philosophy which, when greatly simplified, might be that, for example:

- A system which makes the best use of computers is a good solution.
- A system which produces the best documentation is a good solution.
- A system which is the cheapest to run is a good solution.
- A system which is implemented earliest is a good solution.
- A system which is the most adaptable is a good solution.
- A system which makes the best use of the techniques and tools available is a good solution.
- A system which is liked by the people who are going to use it is a good solution.

In the case of a contingency approach, it might be:

- A system designed for the particular circumstances using appropriate tools and techniques for that particular situation is likely to be a good solution.

This research is about the design and use of information systems development methodologies. They differ greatly, often addressing different objectives. These objectives could be:

i) To record accurately the requirements for an information system.
ii) To provide a systematic method of development in such a way that progress can be effectively monitored.
iii) To provide an information system within an appropriate time limit and at an acceptable cost.
iv) To produce a system which is well documented and easy to maintain.
v) To provide an indication of any changes which need to be made as early as possible in the development process.
vi) To provide a system which is liked by those people affected by that system.

Avison & Fitzgerald (1988a) provide a list of twenty-four criteria for judging the end product following the use of a systems development methodology. The weighting given to each factor will depend on the particular problem situation (for example, the types of user, the likelihood of frequent change, the other systems already computerised, the need for fast development, the need for high reliability, and so on). This suggests that a contingency approach to information systems development might produce gains not apparent in many of the methodologies which are mentioned in Chapter 3, such as the NCC approach (Daniels & Yeates, 1971), SSADM (Downs, Clare & Coe, 1988) or Merise (Quang & Chartier-Kastler, 1991). These approaches are far less flexible, as their basic structure and content are expected to be followed fairly rigorously in all problem situations by all analysts.

In the next section we look further at contingency approaches to information systems development, in particular the Multiview approach.

1.1.3 Multiview: A Contingency Approach

Davis (1982) advocates the contingency approach to information systems development where the methodology chosen will depend on the particular circumstances where it is to be applied. In other words, different methodologies will be used for different situations, a kind of 'horses-for-courses' approach. He suggests that the level of uncertainty in a situation is critical in this choice, and argues that four components of the overall system affect the total level of uncertainty:

- The complexity and ill-structuredness of the system.
- Its current state of flux.
- The number of users affected, skills needed and skills possessed.
- The level of experience and skill of the analysts.

The methodology adopted therefore will be contingent on the particular situation, according to the complexity of the problem and the levels of user and analyst competence. For example, in situations of high uncertainty, a prototyping approach might be adopted.

Other writers such as Burns & Dennis (1985), Capper (1985), Episkopou & Wood-Harper (1986), Land & Somogyi (1986) and Markus (1984) have either extended Davis's work on uncertainty or offered other factors which will affect methodology choice.

The implication here is that organisations will avoid standardising on one chosen methodology because there will be circumstances where it is not particularly suitable. For each particular situation they will have available a number of different information systems development methodologies from which they choose one. As Longworth (1985) and Bubenko (1986) show, there are a large number of information systems development methodologies from which to choose.

Iivari (1987) presents a different contingency framework which emphasises contingent approaches within the methodology rather than between methodologies. This is the type of contingency approach described in this research. Iivari suggests that the choice of tools and techniques used in an application using a contingency framework will depend on:

- The comprehensiveness and depth of the information systems design process required.
- Whether the designers choose a 'goal-oriented' strategy or an 'alternative-oriented' strategy. The goal-oriented strategy negotiates on what is to be achieved, and then proceeds to find ways to accomplish the tasks. The alternative-oriented strategy does not assume that consensus can be reached on the goals, but rather that negotiation must occur on how we are doing things.
- The choice of an appropriate adaptation strategy, reflecting on the perception about

future events. One choice is to ignore future requirements, the second is to assume they are predictable, and the third is to assume that they are unpredictable, but can be dealt with.

- The choice of an appropriate implementation strategy.

Multiview is a contingency approach providing a flexible framework (following Iivari) as an alternative to choosing between different methodologies or standardising on one particular methodology. The techniques and tools available within the Multiview framework are chosen and adjusted according to the particular problem situation.

Multiview is a blended methodology drawing from a number of major methodologies already in use or proposed. The analysis of human activity systems (Checkland, 1984 and Wilson, 1984) and socio-technical systems (Mumford, 1981 and Land & Hirschheim, 1983) has been wedded to the more conventional work on data analysis (Rock-Evans, 1981 and Shave, 1981) and structured analysis (Gane & Sarson, 1979 and DeMarco, 1979) so as to create a theoretical framework for tackling computer systems design which attempts to take account of the different points of view of all the people involved in using a computer system.

However, these earlier works have been interpreted in a particular way (indeed, the original authors may not agree with the interpretation) and since the publication of the original text on Multiview (Wood-Harper, Antill & Avison, 1985) other work has been assimilated into the framework.

We have seen that there are theoretical and academic antecedents, as well as those based on commercial experience. However, the methods described in the thesis are, above all, of practical value. The three cases (Chapters 4, 6 and 7) describe some of that experience. They are there to serve as commentary on the theory of analysis expounded in the thesis. In other words, they are there to address the questions: 'does it work in practice?' and 'what problems accrue from using such an approach?' and therefore 'how can it be improved?'. The cases are an important and integral part of the thesis and are described in the main body of the thesis rather than in appendices. Although they are all overviews of work done, they are substantial in terms of the thesis because there needs to be a fuller description of Multiview in practice. One indication of this comes from an analysis of comments from academics receiving an inspection copy of Wood-Harper, Antill & Avison (1985). The major criticism was that only one case study was presented. Indeed, before this research, this one case was the only substantial example of Multiview in action. This research attempts to redress the balance between theory and practice in the context of Multiview.

As Checkland (1981) has argued, it is not possible to design a series of 'scientific' experiments to test out whether our theories are right. To evaluate a methodology, and in particular one based on contingency, it is necessary to see it working in different

situations and being used by different analysts. It is also important that the process of analysis and its results should be looked at from different viewpoints. What looks good to the analysts may not be so impressive to the user.

The cases are not 'classic examples of how systems analysis should be done'. In a practical discipline one must always distinguish between the ideal methodology as taught in text books, and the realities of any situation which cause departure from the ideal in order to allow for the exigencies of the real world. The cases expose the difference between what one would like to find in an ideal world and what is in the real world. Many design methodologies are prescriptive not only of what must be done but of the order in which it has to be done. This is not always feasible in the real world. For example, decisions are often made before all the facts have been gathered. Frequently this is due to time or cost constraints. In the case described in Chapter 4, it occurs because of political constraints. No one could pretend that this is the ideal way of decision making, but Multiview is based on a recognition of such facts of life and it can help users to make decisions which are supportable even in these situations.

The first case, described in Chapter 4, takes place at a polytechnic distance learning unit. The information system concerns a computerised system related to learning materials for students taking a vocational course at a distance from the polytechnic. The case exposes some strengths and weaknesses of the original Multiview approach. The second case (Chapter 6) involves the development of a prototype computer system for a district health authority. It provides an example where Multiview was adapted significantly for the particular problem situation. User reactions to the prototype information system for community health workers are discussed. The final case (Chapter 7) concerns an integrated information system for an academic department. It provides a full example of the application of a second version of Multiview (detailed in Avison & Wood-Harper, 1990) which is in part a result of the research described in this thesis.

Multiview in operation differs widely according to the particular situation. Projects follow various courses depending on such issues as to whether the objectives are clear and stable, the degree of user involvement that is practicable, working practices, the complexity of the situation, and the type and views of management (Andersen et al, 1990). For this reason, the use of Multiview can be said to be an 'exploration' in information systems development and the three cases described provide an opportunity to see Multiview in operation in different situations. Further cases are discussed in Avison & Wood-Harper (1990). In this thesis there is a discussion in Section 5.1 of the experiences provided by the first case and in Chapter 8 there is a discussion of the lessons learnt and conclusions made from the research based on experiences from all three cases.

These cases were all real examples of Multiview in use. Multiview has been developed in the tradition of action research (discussed more fully in Section 1.2.9), where there is a close interaction between theory and practice, the methodology changing frequently according to the experience gained from using it.

1.2 RESEARCH METHOD

In this section the concern is with research methods relevant to information systems. Action research is stressed, as it is the main research method used in the research described in this thesis. An overview of research methods is provided because action research is not a traditional research method, and there needs to be some discussion and justification of its use in information systems research in general and in this research in particular. Much of the following analysis of research methods is taken from Wood-Harper (1989) and discussed further in Avison & Wood-Harper (1991).

1.2.1 Alternative Research Methods

Information systems can be said to be a 'broad church' discipline with a plurality of research methods (Mumford et al, 1985). Douglas (1976) presents a broad continuum, shown as Figure 1.1, which can be used as a basis for framing the discussion on research methods in information systems.

Figure 1.1 shows that proceeding from left to right on the continuum produces more preconceived categories, with less induction and holism. As we move from right to left, the continuum shows a greater degree of flexibility and participation, but a less controlled environment for research.

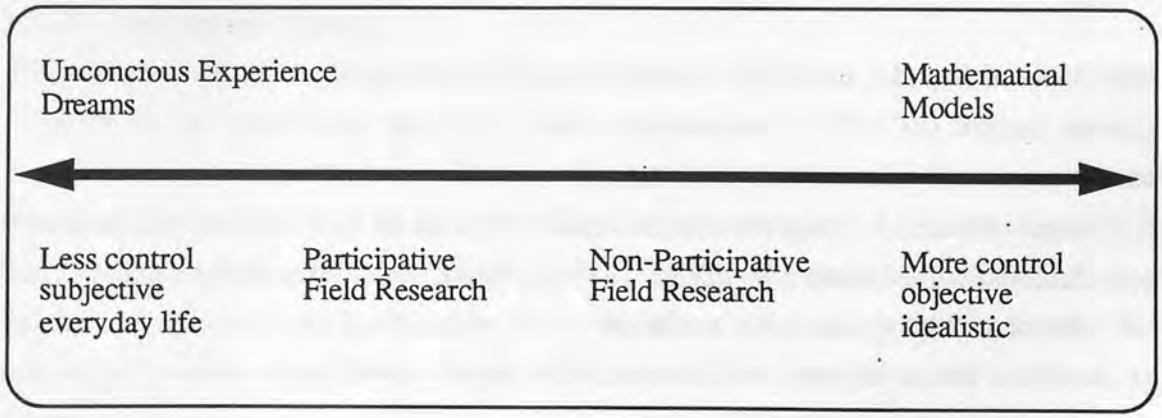


Fig. 1.1. A continuum for framing research methods (adapted from Douglas, 1976)

A more detailed typology produced by Van Horn (1973) forms the foundation for most classification schemes. He identified four methods for empirical research: case studies, field studies, field tests and laboratory studies. Ein-Dor & Segev (1981) produced a similar scheme: field experiments, surveys, case studies and laboratory experiments, whilst Hamilton & Chervany (1981) added conceptual study as a research method.

Galliers (1985) suggests other categories of research method. Amongst these are phenomenological studies which attempt to produce truly 'objective knowledge of what things are' and action research, which differs from a case study or field experiment in that its underlying philosophy includes involvement on the part of the researcher. Although action research was the main approach used in the research described in this thesis, 'many flowers should bloom' (Klein & Lyytinen, 1985), each research method having its place. Galliers (1985) highlights the strengths and weaknesses of each approach and Jenkins (1985) also discusses criteria that might be used when assessing which one to use for a particular situation. Jenkins uses the term participative research rather than action research.

The following research methods are considered:

1 Conceptual study.
2 Mathematical modelling.
3 Laboratory experiment.
4 Field experiment.
5 Surveys.
6 Case studies.
7 Phenomenological research/hermeneutics.
8 Action research.

1.2.2 Conceptual Study

This method is to the extreme right of the continuum in Figure 1.1, and is sometimes referred to as 'armchair research'. The implication is that no actual on-site experimentation is carried out. The development of frameworks for research and representative models may be seen as falling into this category. A related approach is that labelled as philosophical research (Jenkins, 1985). The basis for the research may be that of opinion and speculation. It is therefore a mental pursuit whereby the researcher reasons using flows of logic which are explicit, replicable and verifiable by others.

A conceptual study often involves the logical reasoning of causal relations using a standard set of rules. This ensures that anyone may repeat the process and obtain the same results. These methods require only the thinking process and no active

experimentation. There is therefore no measurement or observation. The testing of a hypothesis is therefore also irrelevant. The major drawback with this approach is the necessity, sometimes difficult, to link the conceptual environment with a 'real environment'. That type of mapping requires the application of one of the other research methodologies used in information systems work.

1.2.3 Mathematical Modelling

Mathematical modelling is to the right on the Douglas continuum of all the empirical research methods. Here the degree of control is nearly absolute. All independent and dependent variables are known, no human subjects are required and no context exists to affect the results. The entire area of interest is condensed into a set of mathematical equations. This reflects a Leibnizian orientation, in which truth is derived based on a well specified set of derivation rules (Churchman, 1971). The major drawback is the inability to condense (or model) many social, organisational and context issues into a set of equations. Objectivity is high, but reflection of the often unpredictable social world is low.

1.2.4 Laboratory Experiments

Jenkins (1985) describes a laboratory experiment as one in which independent variables are manipulated by the researcher, 'noise' variables are controlled or accounted for, and the resulting impact on the dependent variable is observed and measured. On the Douglas continuum, laboratory experiments can be considered to fall on the control and objective end of the spectrum. The choice of type of laboratory experiment depends on the area of concern, and they differ according to, for example, the amount of control exercised by the researched. Nevertheless, all these methods require tight control on the part of the researcher.

The category of laboratory experiments includes experimental simulation, free simulation, experimental gaming and small group experiments. Simulation is presented by Dickson et al (1977) as the development of models of the organisation used to study the impact of variables of interest. It therefore attempts to mirror parts of the real world. Experimental simulation is a closed model. The researcher has control of the presentation of the experimental events and human subjects can be exposed to the model and their responses recorded (Jenkins, 1985). Free simulation differs in that the responses of the subjects may affect the 'how' and 'when' of the events presented to them. This subject determination is in addition to the control exerted by the researcher. Experimental gaming uses a simulator to provide subjects with an artificial decision-making environment with which they interact. The researcher therefore exerts a sizeable measure of control, much like experimental simulation, enabling the controlled

conditions necessary for statistical analysis. Small group experiments (Dickson et al, 1977) are presented as a related research method to experimental gaming. In addition they allow the study of human behaviour in 'real' decision-making environments.

1.2.5 Field Experiments

Field experiments include field studies, field tests, adaptive experiment and group feedback analysis (Dickson et al, 1977 and Jenkins, 1985). These fall more towards the middle of the Douglas continuum, demonstrating a mix of subjective and objective elements, along with a moderate degree of control.

A field study is one conducted in an organisation's natural setting with human subjects. No independent variables are manipulated, but large amounts of data are collected in order to determine its effect. The dependent variables are systematically measured so that the effects of the independent variables on the dependent ones are recorded.

The term field test is used to indicate an experiment which takes place in the organisational environment, where the researcher attempts to manipulate the independent variable to ascertain its effect on the dependent variable. Some control is exerted over the 'noise' variables. The subjects, however, are not randomly assigned to the tests, which may introduce bias into any statistical inferences that one might draw.

An adaptive experiment falls under the category of 'quasi-experimental' research (Jenkins, 1985). Measurements are taken before and after the introduction of independent variables. Data about prior conditions is collected and a control group is maintained, but, as in the field test, the participants are not randomly assigned to the treatments. Group feedback analysis allows an interaction of objective and subjective components. The group is asked to complete an 'objective' instrument or form. The results are analysed. The group is brought back together to get their subjective reactions to the instrument and the data analysis.

1.2.6 Surveys

Surveys, sometimes referred to as opinion research, concerns the gathering of data from human subjects on attitudes, opinions and beliefs. They are usually carried out through one of several methods: mail surveys, telephone surveys or personal interviews. Mail surveys have the advantage of possible large sample sizes. This advantage is realised providing that the respondents have the ability and inclination to complete the questionnaire at their leisure. Another advantage is the relatively low cost. However, two major disadvantages can be noted. Firstly, there is no guarantee that the people who actually respond are those from whom a response is required. Further, people other than the named respondent, such as secretaries, office co-workers, or

family members, may affect the response. Secondly, the problem of non-response may seriously bias the result. The possibility exists that the portion of the sample which did not respond have significantly different opinions from those who responded.

The personal interview suffers mainly from cost, in terms of time to both researcher and respondent, and cost in financial terms to the researcher. Whilst a more in-depth study can be carried out, this is at the expense of a reduced sample size. Again, there may be bias introduced by the nature of people who are willing to invest the necessary time in a 'one on one' interview.

To have any validity for statistical conclusions, the samples must be randomly chosen. Some techniques such as judgement sampling, where the researcher chooses the subjects to select, or convenience sampling, where the researcher interviews those easy to select, or snowball sampling, where a person is interviewed due to a tip from a previous interviewee, and panel groups, where a group may participate over a range of time, are all techniques which may bias the results.

Although, hypothesis testing is feasible using the survey technique, the researcher would be mainly interested in verifying facts either as the population thinks they exist or will exist. There is however little possibility of showing causality, since no treatment is being administered. Jenkins (1985) argues that researchers may determine 'attitudes, opinions, impressions and beliefs of human subjects'.

Douglas (1976) places questionnaires on the control end of the spectrum, whereas interviews fall much closer to the middle of the continuum. Nevertheless, both these methods are used in information systems research, sometimes for cross-checking purposes.

1.2.7 Case Studies

As Benbasat et al (1987) point out, there is no universally accepted definition of a case study. They define a case study as one which:

> '...examines a phenomenon in its natural setting, employing multiple methods of data collection to gather information from one or a few entities (people, groups, organisations). The boundaries of the phenomenon are not clearly evident at the outset of the research and no experimental control or manipulations are used.'

Jenkins (1985) points out that the researcher is not involved, as no intervention takes place and no control is exercised, and that the research captures the nature of a certain environment at some point in time.

Benbasat et al (1987) identify four criteria to assess the appropriateness of the case

study method. The first is the question of whether the area of interest can be divorced from its natural setting: if it cannot, then the case study approach may be appropriate. The second is whether or not the interest lies in current events: if it does, then the case study approach may be appropriate. Longitudinal studies, where a view is taken of 'the relationship between organisational structure and process in the context of historical change' (Galliers, 1985) may be appropriate for long-term studies. Galliers also describes futures research, where different scenarios are postulated and different inputs identified. Longitudinal studies and futures research are sometimes differentiated from case studies and sometimes included in this category. The third is the question of the need for control or manipulation of variables: if there is such a need, then the case study approach will not be appropriate. The fourth is the existence of an established theoretical base: if there is no established theoretical base, then the case study approach may be appropriate.

Case studies might utilise questionnaires, interviews, archival records, documentation, physical artifacts or direct observation as their source of evidence. However, some authors would not include archival research, where historical documents are examined and/or recorded data studied, in the case study category.

Benbasat et al, (1987) identified three reasons why the case study is appropriate for information systems research:

1. The study can be done in a natural setting, learning about the state of the art, and generating theories for practice.
2. The nature and complexity of the process can be studied.
3. Situations which are new and rapidly changing can be studied using the approach.

The case study lies on the subjective end of the Douglas continuum, a situation which has often constituted its main source of criticism in the United States. It has been argued that the case study is relatively high in cost, and control is minimal or even non-existent. Offsetting this disadvantage, the strength of the case study is in its use for study of the natural situation, including contextual influences, and for the provision of a deep and comprehensive analysis which is unavailable through the application of other methods discussed previously.

1.2.8 Phenomenology/Hermeneutics

Boland (1985) defines phenomenology as:

> 'the methodological study of consciousness in order to understand the essence of experience . . . phenomenology is concerned with the structure of meaning that gives sense and significance to our immediate experience.'

In other words, the concern is not in finding out how things work but rather in what things are (Galliers, 1985). Boland points out that a hermeneutic problem is one involving the interpretation of situations. The impetus of this mode of research is that we cannot really understand communication itself unless we are aware of the context in which that communication was generated.

Hermeneutics is an ancient science, its main origins being the interpretation of religious texts. Two schools of thought existed in this science. One was the objectivist school, which wished to 'decontextualise' the text, thereby producing an objective analysis of what is really there. This is described in Winograd & Flores (1986). The other school views interpretation as an essential key, allowing us to map from the prejudices of the communicator to our current set of prejudices.

Heidegger (1962) refutes the idea of the objective world, which needs no interpretation, and the subjective world comprising of our thoughts and feelings only. He argues that the two are not independent, but rather that one does not exist without the other. No objective view of reality exists, it is flavoured by our subjective perception.

The method may be seen as close to the subjective end of the spectrum. Its application may not be so much as an independent method but an approach to use in conjunction with other methods, for example action research or case study. Phenomenology/hermeneutics would not be appropriate where the researcher wishes to use statistical inference as the 'guarantor of truth'.

1.2.9 Action Research

In action research, described in Checkland (1981), researchers test and refine principles, tools, techniques and methodologies to address real-world problems. It is characteristic of action research that the practitioners as well as the researchers participate in the analysis, design and implementation processes and contribute as much as the researchers in any decision-making. Thus there is a synergy between the researchers and practitioners, the researchers building up theories and modifying them on the basis of practical experience and the practitioners using and modifying research ideas for solving real-world problems.

Action research is often confused with case study research, but, as Benbasat et al (1987) have shown, whereas case study research examines phenomena in its natural setting with the researcher an independent outsider, in action research the researcher might be a participant in the implementation of a system and simultaneously evaluate a particular approach. In action research, the researcher ought to be 'useful' as well as an observer. The justification for action research is based on the acceptance of the

subjective processes and the possibility that knowledge is socially constructed.

The history of the development of action research has been traced by Warmington (1980), who attributes the first explicit use of the method as that applied to social problems by the social psychologist Kurt Lewin in the 1940s. In Warmington's account, the implicit use of the method by researchers such as F.W.Taylor and Elton Mayo in solving practical organisational problems predates Lewin's work.

Other versions of the method were used in the 1960s by the Tavistock Institute and Lancaster University in the United Kingdom. Some of the work at the Tavistock Institute in developing a socio-technical view and that at Lancaster in soft systems have been very influential to the formation of the Multiview approach.

Hult & Lennung (1980) argue that:

> 'Action research simultaneously assists in practical problem-solving and expands scientific knowledge, as well as enhances the competencies of the respective actors, being performed collaboratively in an immediate situation, using data feedback in a cyclical process aiming at an increased understanding of a given social situation, primarily applicable for the understanding of change processes in social systems and undertaken within a mutually acceptable ethical framework.'

Action research therefore attempts to link theory and practice, thinking and doing, achieving both practical and research objectives. Gaining this knowledge is seen as an active process, so that beliefs may be redefined in light of the outcomes. A representation of reality is not so much desired, but rather is a means for dealing with reality. Action research is a pragmatic approach which desires to 'come to terms' with the world. This emphasis on the pragmatic is further reflected by Susman (1983), who states that action research builds on a learning cycle. His learning cycle is diagnosis, action planning, action taking, evaluation and the specification of learning. The researchers/actors/participants/subjects begin in a real concrete situation and return to it at the end of the learning cycle.

This type of learning represents the enhanced understanding of a complex problem. Information about a particular situation and a particular environment have been obtained, which gives a contingent value to the truth learned. The researcher expects, however, to generate knowledge which will further enhance the development of models and theories. The aim is the understanding of the complex human process rather than a universal prescriptive truth.

The mutually accepted ethical framework mentioned in the quote from Hult &

Lennung (1980) may cause some concerns. If the goals of the researcher and client differ drastically, there is tension. Where these tensions do occur, some method for satisfying each of their goals must be found (Warmington, 1980 and Jackson, 1987). As Willcocks & Mason (1987) show, coercive situations are not uncommon in information systems development, the context of the action research work described in this thesis. Unresolved conflicts did not occur in this research, but such a situation did occur in that described in Mansell (1990).

Finally, in the process of learning, an explicit clear conceptual framework must exist in which the researcher imposes on the situation, which is acceptable to that researcher as well as the organisational actors in the action research study. This is needed so that the explicit lessons will emerge from the research cycle.

In comparing action research to the other methods, three important distinctions can be brought out:

1. The researcher is actively involved, with expected benefit for both researcher and organisation.
2. Knowledge obtained can be immediately applied. There is not the sense of the detached observer, but that of an active participant wishing to utilise any lessons learned based on a explicit clear conceptual framework.
3. The research is a cyclical process linking theory and practice.

The above discussion can be seen within Checkland's intellectual context: the 'organised use of rational thought' (Checkland, 1985). This context is based on a simple model of a cycle of continuous inquiry where theory interacts with practice.

Depicted in Figure 1.2 is the use of a theory: an intellectual framework of linked ideas (**F**), through a methodology or an intervention process (**M**), to an application area (**A**). In going through a cycle, learning can be generated about **F, M** and **A.**

Fig. 1.2. Organised use of rational thought (source: Checkland, 1985)

In the particular research described in this thesis, the area of application is the development of information systems. The researcher has had many years' experience teaching information systems and as a practicing systems analyst. There is confusion over choice of approaches that might be appropriate for information systems development. In an attempt to remedy this, after a theoretical exploration (discussed in Chapter 2 of the thesis), an intellectual framework, called Multiview (discussed in Chapter 3), was constructed. The question now is which research method is appropriate to learn about the theory, the methodology and the development of information systems?

The need for this learning implies that the practical implications of the three aspects should be explored. To do this, the methodology ought to be used as an exploratory tool to search out patterns in the practical world (an example of one such effort is given in Chapter 4). There are iterations in the research - it is a cyclical process - and as the results from the work discussed in Chapter 4 are analysed, Multiview was adapted (Chapter 5) to take account of the practical experience of using it. The modified methodology is itself used as an exploratory tool (two further examples are given in Chapters 6 and 7). Chapter 8 looks at lessons learnt and suggests further adaptations to Multiview which will be explored further in practical situations. The research is continuous and this thesis represents some parts of the process.

Why was action research chosen as the main method in this research? In helping us in the choice of a research method, Galliers & Land (1987) have suggested that out of the research methods discussed in Section 1.1.2, mathematical modelling and laboratory experiments are inappropriate for research into methodologies. By reviewing the remaining methods in order to choose an appropriate one, it is clear that the conceptual study does not fulfil the requirements for learning in practice mentioned above. Further, out of the rest of the research methods, only action research fulfils the research requirements to understand fully the Multiview methodology in a practical situation. It gives the opportunity for the researcher to be involved in the practical work and get feedback from the practitioners to adapt Multiview and apply it again.

Land (1983) captures these research requirements, the choice in this study and the implications in using action research:

> 'If you're going to do research in this area, which method of research is valid? We're concerned with a multi-dimensional world in which it is very difficult to analyse cause and effect...some people call it action research...develop techniques, methods and methodologies and thus we go into the world and try them out...find a host who is willing to use these and we try to see as best as

possible how things will work out...It is difficult to say what effect is due to the methodology or to some personal or environmental factors... '

The implications can also be seen in comparison to the other methods. As described above, the methods should be viewed as a continuum and the comparison of the extremes, laboratory experiments (positivist science) to action research, is shown in Figure 1.3.

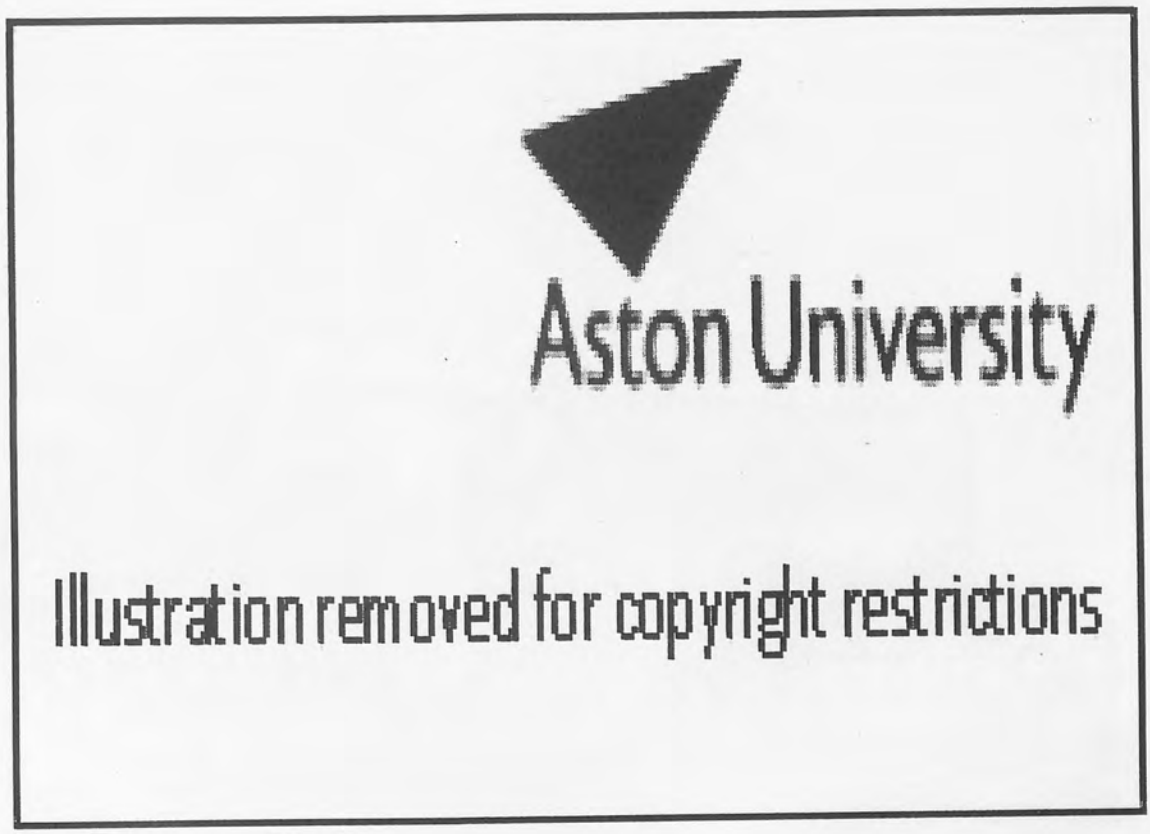


Fig. 1.3. A comparison of positivist science and action research (from Susman & Evered, 1978)

Much information systems research, particularly in the United States, is towards the right of the Douglas continuum. This is evidenced in McFarlan (1984). Weick (1984) argues that information systems research should be more directed towards the middle. Action research is towards the left of the continuum, but it can be appropriate to information systems research. It is hoped that the above short discussion has demonstrated its possible appropriateness to the research area described in this thesis.

1.3 PLAN OF THESIS

This chapter has looked at the area of research, that of information systems, information systems development methodologies, and one approach in particular, Multiview, which is a contingency approach to information systems development. It has also looked at information systems research in general, and placed research presented in this thesis in the action research category. Criticisms of the action research method has been discussed along with its strengths.

In Chapter 2 themes in information systems development methodologies are described. Each of these themes addresses one or more of the weaknesses of the conventional approach to information systems development, which is discussed first, but no single approach is appropriate to all problem situations. However, each has an important contribution to information systems development, and Multiview has been greatly influenced by each theme discussed in the chapter.

The original version of Multiview is described in Chapter 3. As the discussion in Chapter 2 shows, no single approach to information systems development is appropriate for all problem situations. Nevertheless each has strengths which shall be harnessed in information systems development where appropriate. Multiview is a blended methodology, having five stages: the analysis of human activity systems, information analysis, the analysis and design of socio-technical systems, the design of the human-computer interface and the design of the technical solution. The tools and techniques described in its exposition are used contingent on the particular problem situation.

In Chapter 4 a case is examined. It was the first real use of Multiview in action. This took place at South Bank Polytechnic and the information system designed supports distance learning courses at the Polytechnic.

In Chapter 5 there is a re-evaluation of Multiview based on a consideration of the approach in isolation, as a result of the experiences gained from the particular piece of action research described in Chapter 4, and based on improvements to information systems knowledge and practice gained since the early exposition of Multiview in 1985. Suggestions are made in terms of improving the techniques already used and in

adding new techniques and tools to the "tool box'.

Chapters 6 and 7 look at two other cases using the updated version of Multiview. The first takes place at a district health authority and concerns an information system supporting community health care and the second concerns the development of an information system at Aston University's Department of Computer Science and Applied Mathematics. The final chapter discusses lessons learnt from this field work and suggests further research work.

Chapter 2
Themes in Information Systems Development Methodologies

2.1 INTRODUCTION

An information systems development methodology is usually adopted to provide a better end product, a better development process and a standardised process (Avison & Fitzgerald, 1988a). There appears to exist many hundreds of system development methodologies. Longworth (1985) identified over 300 and Bubenko (1986) argues that:

> 'It is a reasonable estimate that hundreds of more or less similar methodologies have been published. In practice, probably tens of thousands of more or less different approaches are being used. Most organizations have developed their own methodology and prescribed it in the organization's (data processing) handbook.'

There have been many attempts either to compare methodologies or to provide frameworks for their comparison (Avison & Fitzgerald, 1988a, Bjørn-Andersen, 1984, Maddison, 1983, and Olle et al, 1982, 1983 and 1986).

This chapter provides an overview of the various themes in information systems development methodologies that have emerged which have been influential in the design of Multiview. The chapter therefore provides a background to Multiview, which is outlined in Chapter 3. Some themes, for example, formal methods, are not discussed because they are not fundamental to the design of Multiview. The first theme is called conventional systems analysis and design, and describes a generalised methodology based on the approach used in the 1970s for developing information systems. Weaknesses are exposed, as it is these weaknesses which other approaches have attempted to address. These approaches have been categorised according to the following themes: systems, planning, participation, prototyping and automation, structured analysis and data oriented. Examples of actual methodologies which fall under each of these themes are mentioned. This is not straightforward, because most methodologies are blended to some extent and they have been influenced by a number of these themes.

2.2 CONVENTIONAL SYSTEMS ANALYSIS

Before looking at the conventional approach, we consider computer applications of the 1950s and 1960s when there was no formalised methodology to develop data processing systems. Early computing was associated with scientific applications, and though some computers were installed in business environments, there were few documentation standards, training schemes or practical guidelines which gave help on their use for commercial applications. The analysis work was rarely done well. The people who implemented these systems were usually programmers, who were not necessarily good communicators, thus making it difficult for users to communicate their needs to the technologists. Further, the development of applications frequently are more costly and arrived later than expected. Projects were seen more as short term exercises or one-off solutions to sort out problems, than as long-term, well-planned implementation strategies for new applications. Technologists were learning their skills at the expense of the user. As a consequence, users were frequently dissatisfied with the operational systems. Their needs had not been clearly identified in any analysis phase.

As a reaction to this, there was a growing appreciation of the importance of that part of the development of the system that concerns analysis and design. There was a change of emphasis in many data processing departments. Some job titles changed to programmer-analysts, analyst-programmers, and systems analysts. Another reaction to these problems was the appearance of methodologies to develop computer applications. That of the National Computing Centre (NCC) in the United Kingdom typified the methodologies of this time. It is fully described in Daniels & Yeates (1971).

The methodology has the following steps:

- Feasibility study.
- System Investigation.
- Systems Analysis.
- Systems Design.
- Implementation.
- Review and Maintenance.

These stages together form a life-cycle for systems development, from birth at the feasibility stage to the time in the reviewing process when the system is found to be inadequate and the process starts again with a new feasibility study.

This approach had all the attributes that we expect of a systems development methodology:

- A series of phases from the feasibility study to review and maintenance, and each of these has sub-phases. The implementation phase, for example, had a number of sub-phases, such as program development, systems testing, training and education,

writing user guide and operations guide, and changeover, and each of these had sub-phases themselves, all described in detail in the manual supporting the methodology, and on the NCC training courses.

- A series of techniques, such as ways to evaluate the costs and benefits of different solutions in the feasibility study, and the techniques of observation, interviewing, designing and writing questionnaires, and sampling, at the systems investigation phase.
- A series of tools, which could be used, in particular documentation aids such as the various flowcharts, and file and record specifications forms. Some tools were automated, particularly in later years, such as those supporting project management.
- A training scheme, so that all analysts could adopt the standards suggested. It was taught in a number of courses given by the NCC at a number of educational institutions.
- A philosophy, perhaps implied rather than stated, which might be that 'computer systems are usually good solutions to clerical problems'.

This conventional systems analysis methodology has a number of features to commend it. It has been well tried and tested. The use of documentation standards helps to ensure that the specifications are complete, and that they are communicated to systems development staff, the users in the department, and the computer operations staff. It also ensures that these people are trained to use the system. The education of users on subjects such as the general usage of computers is also recommended, and helps to dispel fears about the effects of computers. Following this methodology also prevents - to some extent at least - missed cut-over dates (the date when the system is due to become operational) and unexpectedly high costs and lower than expected benefits. At the end of each phase the technologists and the users have an opportunity to review progress. By dividing the development of a system into phases, each sub-divided into more manageable tasks, along with the improved training and the techniques of communication offered, the conventional approach gave much greater control over the development of computer applications than before.

Although the methodologies typified by the NCC approach of the 'seventies were adequate, indeed enlightened perhaps, for their day, there are also serious limitations to this approach. These are described in more detail in Avison (1985) and include:

- *Failure to meet the needs of management*: Although systems developed by this approach often successfully deal with such operational processing as payroll and the various accounting routines, the needs of middle management and top management have been largely ignored by computer data processing.
- *Unambitious systems design*: Computer systems usually replaced manual systems, which had proved inadequate in changing circumstances. Apart from using a new

technology, the computer system designs were often similar to those of the existing systems. More radical, and potentially more beneficial, computer systems were not being implemented.

- *Models of processes are unstable*: The conventional methodology attempts to improve the way that the processes in businesses are carried out. However, businesses do change, and processes need to change frequently to adapt to new circumstances in the business environment. Because computer systems model processes, they had to be modified or rewritten frequently.
- *Output driven design leads to inflexibility*: The outputs that the system is meant to produce are usually decided very early in the systems development process. Design is 'output driven'. However, changes to required outputs can be frequent and because the system has been designed from the outputs backwards, changes in requirements to outputs usually necessitated a very large change to the system design.
- *User dissatisfaction*: Many companies expected users to 'sign off' their requirements at an early stage when they do not have the information to agree the exact requirements of the system. Sometimes systems are rejected as soon as they are implemented - it may well be only then that the users see the repercussions of their 'decisions'. Further, as we have seen, it was normally very difficult to incorporate changes in requirements once the systems development is under way.
- *Problems with documentation*: The NCC documentation is oriented towards the computer person and not the user, and therefore did not sufficiently help communication between the two groups. A further problem is that forms are often completed reluctantly and therefore rarely done well and frequently not kept up to date. The documentation therefore cannot be relied upon as an accurate reflection of the system.
- *Maintenance workload*: Because of user dissatisfaction with the operational systems, maintenance of them can be a major user of resources. This leads to a long queue of applications in the pipeline.
- *Application backlog*: There may well be a number of systems waiting to be developed, a consequence largely of the maintenance workload. Users may also postpone requests because the systems will not be developed in time, leading to an invisible backlog.

The approaches that follow attempt to address one or more of these limitations in the conventional approach.

2.3 SYSTEMS APPROACHES

General systems theory attempts to understand the nature of systems which are large and complex. It stems in modern times from the work of Bertalanffy (see, for example, Bertalanffy, 1968). Although often considered to be too impractical and wide ranging (Wood-Harper & Fitzgerald, 1982), many of its principles have been taken up by the information systems community, and are an integral feature of a number of information systems methodologies, in particular soft systems methodology (Checkland, 1981) and viable systems diagnosis (Beer, 1985 and Espejo & Harden, 1989).

Ackoff (1971) defines a system as a set of inter-related elements. A system will have a set of inputs going into it, a set of outputs going out of it, and a set of processes which convert the inputs to the outputs. Systems also have a purpose, and the inter-related elements interact in the attempt to achieve this purpose. The boundary of a system is defined when we describe it. This may not correspond to any physical or cultural division. A payroll system might include all the activities involved in the payment of staff in a business. These activities fall within the boundary of that system and those systems outside it, with which it relates, are referred to as the environment. The systems approach concerns itself with inter-actions between the system and its environment.

One of the bases of systems theory concerns Aristotle's dictum that the whole is greater than the sum of the parts. This would suggest that we must try to develop information systems for the widest possible context, at least the organisation as a whole rather than for functions in isolation. If we fail to follow this principle then a small part of the organisation may be operating to the detriment of the organisation as a whole. (Unambitious and limited systems design was one of the criticisms of the conventional approach discussed in Section 2.2.) Further, if we do break up a complex problem into smaller manageable units, a step in many of the methodologies, we are assuming that this division will not distort the overall system being studied. This is referred to as being reductionist. This may be reasonable in a scientific discipline, but information systems concern people as well as technology, and such human activity systems are more complex. The human components in particular may react differently when examined singly as when they play a role in the whole system.

Another aspect of systems theory is that organisations are open systems. They are not closed and self-contained, and therefore the relationship between the organisation and its environment is important. They will exchange information with the environment, both influencing the environment and being influenced by it. The system which we call the organisation will be effected by, for example, policies of the government, competitors, suppliers and customers, and unless these are taken into account, predictions regarding the organisation will be incorrect. As organisations are

complex systems this would suggest that we require a wide-range of expertise and experience to understand their nature and how they react with the outside world. Multi-disciplinary teams might be needed to attempt to understand organisations and analyse and develop information systems.

Systems theory has had widespread influence in information systems work. It would suggest, for example, that whatever methodology is adopted, the systems analyst ought to look at the organisation as a whole and also be aware of externalities beyond the obvious boundaries of the system - its customers, competitors, governments, and so on (Churchman, 1979). Systems theory would also suggest that a multi-disciplinary team of analysts, not all computer-orientated, is much more likely to understand the organisation and suggest better solutions to problems. After all, specialisms are artificial and arbitrary divisions. Such an approach should prevent the automatic assumption that computer solutions are always appropriate (Section 2.2) as well as preventing a study of problem situations from only one narrow point of view - another criticism of the conventional approach. With this approach comes the acknowledgement that there may be a variety of possible solutions, none of them obviously 'best' perhaps, but each having some advantages. These solutions may involve structural, procedural, attitudinal or environmental change.

Checkland (1981) has attempted to turn the tenets of systems theory into a practical methodology which is called *Soft Systems Methodology* (SSM). It is further developed, in the context of information systems, in Wilson (1984). Checkland argues that systems analysts apply their craft to problems which are not well defined. These fuzzy problems are common in organisations. His description of human activity systems also acknowledges the importance of people in organisations. It is relatively easy to model data and processes, but to understand organisations, it is essential to include people in the model. This is difficult because of the unpredictable nature of human activity systems. The claims of the proponents of this approach are that a better understanding of these complex problem situations is more likely to result using this approach than with the more simplistic conventional approach already discussed or the structured or data orientated approaches (Sections 2.7 and 2.8).

The methodology of Checkland has been developed at Lancaster University. It was developed through action research (Section 1.2.1) whereby the systems ideas are tested out in client organisations by ISCOL (a consultancy company owned by the University). The analysts neither dictate the way the action develops nor step outside as observers: they are participants in the action and results are unpredictable. The practical work provides experience which can be used to draw conclusions and to modify these ideas. This proves to be a successful approach in this context because it is not possible to develop a good 'laboratory model' of human activity systems and set up repeatable

experiments.

Figure 2.1 (adapted from Checkland, 1975) summarises the stages of Checkland's methodology. Stages 1 and 2 are about finding out about the problem situation. The unstructured view gives some basic information from the views of the individuals involved which are then expressed in the form of a rich picture. The 'picture' represents both subjective and objective perceptions of the problem situation in diagrammatic and pictorial form, showing the structures of the processes and their relation to each other. From the rich picture the problem solver extracts problem themes - things noticed in the picture that are, or may be, causing problems. Taking these problem themes, the next stage of the methodology involves the problem solver imagining and naming relevant systems that may help to relieve the problem theme. By relevant is meant a way of looking at the problem which provides a useful insight. Several different relevant systems should be explored to see which is the most useful. After constructing a rich picture, a root definition is developed for the relevant system. A root definition is a kind of hypothesis about the relevant system, and improvements to it, that might help the problem situation:

> 'The root definition is a concise, tightly constructed description of a human activity system which states what the system is' (Checkland 1981).

Using the checklist technique called the CATWOE criteria (Section 3.2), root definitions are created. In stage 3 the analyst selects from these those views which which he or she considers gives insight to the problem situation. Stage 4 is to do with model building, that is, what the systems analyst might do (as against what the system is - the root definition). Stage 5 compares the conceptual models from stage 4 and the rich pictures formed at stage 2. This comparison process leads to a set of recommendations regarding change, and Stage 6 analyses these recommendations in terms of what is feasible and desirable. Stage 7, the final phase, suggests actions to improve the problem situation. Note that the methodology does not describe methods for implementing solutions. The methodology helps in understanding problem situations rather than provides a scheme for solving a particular problem.

Multiview uses rich pictures, root definitions and conceptual models as techniques for investigation, and these will be discussed in more detail in Section 3.2. Unlike most other methodologies, particularly 'hard' methodologies such as structured analysis (Section 2.7) and data analysis (Section 2.8), SSM relies much less on techniques and tools. SSM provides all actors, including the analysts, opportunities to understand and

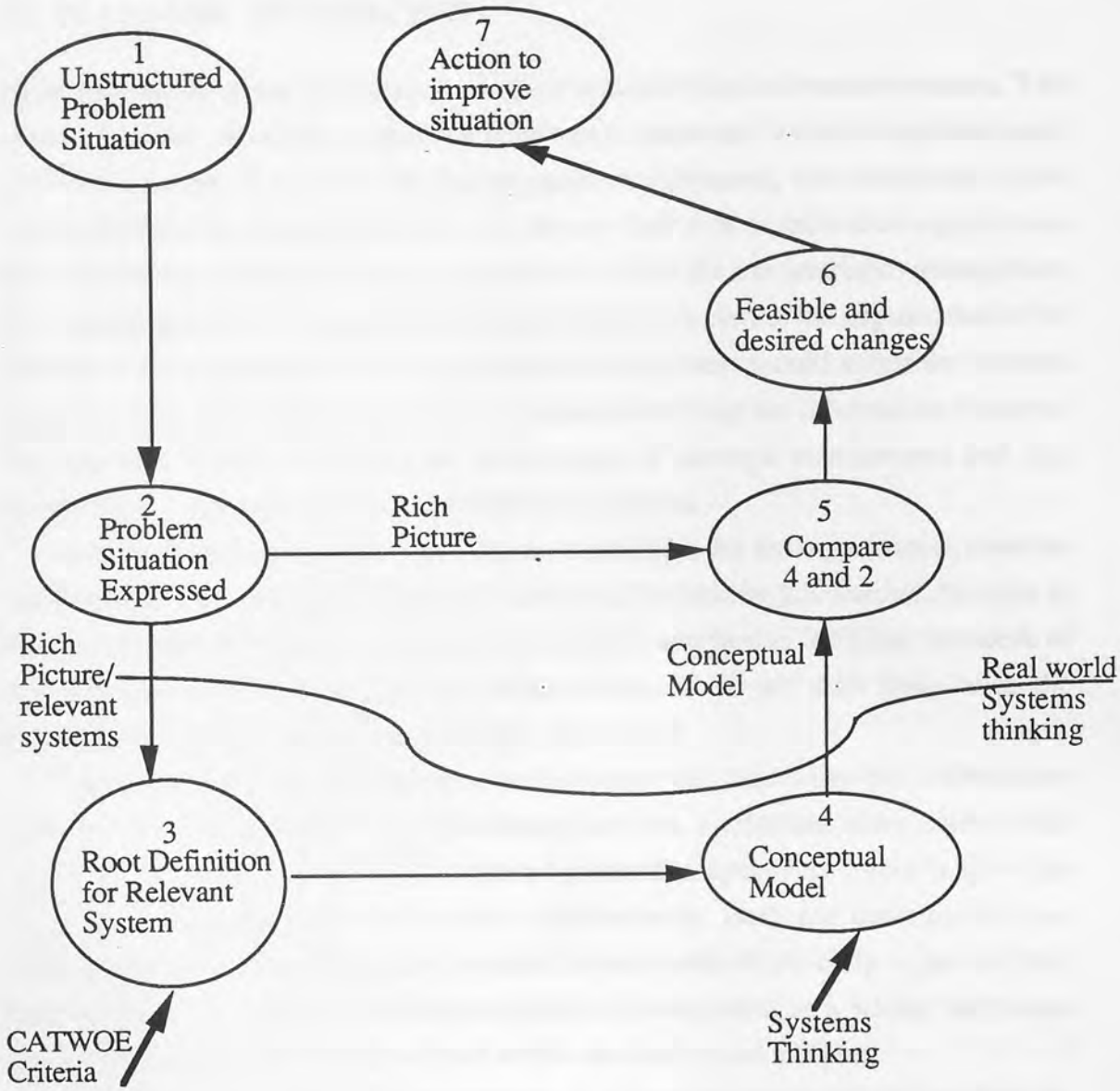


Fig. 2.1. Analysis of human activity systems to depict processes and products

to deal with the problem situation, that is, the human activity system. In the tradition of action research, the analysts are perceived as actors involved in the problem situation, as much as those of the client and problem owner - they are not perceived as outside onlookers providing objectivity (as in scientific research).

The process is iterative and the analysts learn about the system and are not expected to follow a laid down set of procedures. This does present problems: it is difficult to teach and to train others, and it is difficult to know when a stage in the project has been satisfactorily completed. However, these features are also its strengths, because it does not have any pre-conceived notions of a 'solution' and use of the methodology gives a better understanding of the problem situation.

2.4 PLANNING APPROACHES

These approaches stress the planning involved in developing information systems. Two criticisms of the conventional approach were that the approach looked at applications in isolation and that it ignored the management requirement, concentrating on the computerisation of operational functions. Rather than look at individual applications and sub-systems in detail, planning approaches involve the top (strategic) management (the managing director, financial services manager, and so on) of the organisation in the analysis of the objectives of that organisation. Management should assess the possible ways in which these objectives might be achieved utilising the information resource. The approach therefore requires the involvement of strategic management and data processing management in planning information systems.

Because of the requirement to develop an overall plan for the organisation, there are obvious links with the systems approach, described in Section 2.3. Further, because of the top management involvement, and the emphasis attached to fulfilling the needs of top management in controlling the organisation, there are also links with the participative approach which is described in Section 2.5.

Planning approaches are designed to counteract the possibility that information systems will be implemented in a piecemeal fashion, a criticism often made of the traditional approach. A narrow, function-by-function approach, could lead to the various applications failing to integrate satisfactorily. Both top management and technologists therefore need to look at organisational needs in the early stages and they need to develop a plan for information systems development as a whole. Individual information systems are then developed within the confines of this plan.

There is a need for information systems to address corporate objectives directly. Planning approaches aim to ensure that management needs are met by information systems. More radically, some information systems are designed around management needs, a sort of 'top-down' approach (Blumenthal, 1969, Zani, 1970, and Crowe & Avison, 1980), perhaps ignoring operational needs.

Managers may also set standards for information systems and one of these requirements will be choosing the methodology for developing information systems (Maddison, 1983 and Avison & Fitzgerald, 1988a).

Many general information systems development methodologies discuss strategic planning. It may be considered as a separate stage called business analysis which is carried out early in the development of information systems. In Avison (1985), business analysis is described as involving an assessment of the strategic goals of the organisation which could be long term survival, increasing market share, increasing profits, or improving public image. The business analysis stage looks at the environment of the organisation - customers, suppliers, government, trade unions and

financial institutions - whose actions will effect business performance. It also looks at the key personnel in the organisation. With this knowledge, it is possible to assess the type of information that an information system might provide. The first stage of Information Engineering (Martin & Finkelstein, 1981) also concerns itself with planning aspects.

Other approaches look at critical success factors (see Rockart, 1979). These are the factors in the business environment which are considered critical to the continued success of the business. In this way it is possible to establish business goals. A methodology firmly based on critical success factors is described in Bullen & Rockart (1984).

Methodologies that emphasise these planning and top management aspects fully are classified as planning approaches. There is a need for this emphasis. One survey (Galliers, 1986) has revealed that although most respondents agreed that systems planning and business objectives should be aligned, only 10% of 130 organisations surveyed claimed that they had successfully married planning with business objectives and only 43% expressed the view that there were even tenuous links in these organisations. The survey also showed that top management was unambitious about systems planning, for example, giving a low priority to developing systems which attempted to seek out those opportunities that could give the business competitive advantage.

Three approaches which address these issues more fully are BIAIT, BSP and ends/means analysis (Wetherbe & Davis, 1983 and Davis & Olsen, 1985). In Business Information Analysis and Integration Technique (BIAIT) (Carlson, 1979 and Burnstine, 1986), fundamental questions are asked which relate to the objectives of the business. In the approach these are reduced to seven basic questions. Each question is constructed in such a way that either there can only be two possible answers, 'yes' or 'no', or a choice is made from options provided. A grid or matrix of possible responses is formed in the model generated by BIAIT. This profile is used to suggest reports which analyse whether the set of objectives have been met and the information handling necessary. Only at this point does the method begin to look at possible information systems, computer-based or otherwise, which might support these requirements.

IBM's Business Systems Planning (BSP) (IBM, 1975 and Martin, 1980), also addresses the requirements of top management and the importance of ensuring that information systems development coincides with and supports the business plan. BSP follows three principles. The first stresses the need for an organisation-wide perspective. This factor alone separates the BSP approach from the conventional approach discussed in Section 2.2. Rather than address the information needs of any single area of the organisation, it takes the perspective of strategic management. The

second principle suggests analysis from top management downwards (but implementation from the detailed level upwards). It is strategic management that define organisational needs and priorities to help ensure that their perspective predominates in the initial system definition. The BSP approach takes on a bottom-up orientation during the design and implementation phases. This is when the database is created and the processing requirements defined which are necessary to fulfil the organisational objectives. The final principle establishes the need for independence of the business plan from the computer application systems, including data storage aspects. This means that desirable organisation changes are not prevented from taking place because of restrictions of the computer systems. Similarly, the computer application systems can be modified, perhaps by using newer technology, without effecting the organisation.

The phases of BSP are:

- Identification of requirements.
- Definition of requirements.
- General design.
- Detailed design.
- Development and test.
- Installation.
- Operation.

This is not too dissimilar to the conventional approach, except for its emphasis on strategic planning in the early stages and its iterative design process from the general to the specific.

The organisation-wide perspective is obtained by interviewing key executives and is structured to provide information relating to the organisational environment, organisational plans, planning processes, organisational structure, products, markets, geographical distribution, financial statistics, industry position, industry trends, and major problem areas. Information is also gathered about the current information systems and how these support the organisation. Following this analysis it is possible to show how information systems can better assist in improving management decision-making. These information systems should be part of an integrated plan for using the information systems resource most effectively to help top management both in the short and long terms.

2.5 PARTICIPATION AND SOCIO-TECHNICAL APPROACHES

In conventional systems analysis, the computer professional was the person who was making the real decisions and driving the development process. Systems analysts were trained in, and knowledgeable of, the technological and economic aspects of computer

applications but far more rarely on the human (or behavioural) aspects which are at least as important. As we saw in Section 2.2, users were frequently dissatisfied with operational systems, and top management did little more than pay lip-service to computing. The systems analyst may be happy with the system when it is implemented. However, this is of little significance if the users, who are the customers, are not satisfied with it.

The planning approaches discussed in the previous section highlighted the necessity for top management to play a role in information systems development. The approach discussed in this section highlights the role of all users who may control and take the lead in the development process. If the users are involved in the analysis, design and implementation of information systems relevant to their own work, particularly if this involvement has meant user decision-making, these users are more likely to be fully committed to the information system when operational. This will increase the likelihood of its success (Mumford, Land & Hawgood, 1978).

Some information systems may 'work' in that they are technically viable, but fail because of 'people problems'. For example, users may feel that the new system will make their job more demanding, less secure, will change their relationship with others, or will lead to a loss of the independence that they previously enjoyed. As a result of these feelings, users may do their best to ensure that the computer system does not succeed. This may show itself in attempts to 'beat the system', for example, by 'losing' documents. Frequently it manifests itself in people blaming the system for causing difficulties that may well be caused by other factors. Some people may just want to have 'nothing to do with the computer system'. In this kind of situation, information systems are unlikely to be successful, or at the very least, fail to achieve their potential.

One way to help achieve more successful information systems is to involve all those affected by computer systems in the process of developing systems. As we saw in Section 2.4, this includes the top management of the organisation as well as operational level staff. The advocates of the participative approach (see, for example, Land & Hirschheim, 1983) recommend a working environment where the users and analysts work as a team rather than as expert and non-expert. Although the technologist might be more expert in computing matters, the user has the expertise in the application area. It can be argued that the latter is the more important. An information system can make do with poor equipment, but not poor knowledge of the application. Where the users and technologist work hand in hand, there is less likely to be a misunderstanding by the analyst which might result in an inappropriate system. The user will also know how the new system operates by the time it is implemented with the result that there are likely to be fewer 'teething troubles' with the new system.

The role of computer analysts in this scenario may be that of 'facilitators', advising

on the possibilities from which the user chooses and helping these to fruition. This movement can be aided by the use of application packages which the users can try out and therefore choose what is best for them. Another possibility is the development of a 'prototype'. There are packages available which will set up screen layouts and bring in blocks of code for validating or presenting data. The users can compare all these possibilities before making up their minds on a final design. (Prototyping is discussed in Section 2.6.)

Mumford (1983b) distinguishes between three levels of participation: consultative, representative and consensus. *Consultative participation* is the lowest level of participation and leaves the main design tasks to the systems analysts, but tries to ensure that all staff in the user department are consulted about the change. The systems analysts are encouraged to provide opportunity for increasing job satisfaction when redesigning the system. It may be possible to organise the users into groups to discuss aspects of the new system and make suggestions to the analysts. Most advocates of the conventional approach to system development discussed in Section 2.2 would probably accept that there is a need for this level of participation in the design process.

Representative participation requires a higher level of involvement of user department staff. Here, the 'design group' consists of user representatives and systems analysts. No longer is it expected that the technologist dictates to the users the design of their work system. Users have equal say in any decision. It is to be hoped that the representatives represent the interests of all the users affected by the design decisions.

Finally, *consensus participation* attempts to involve all user department staff throughout the design process, indeed this process is user-driven. It may be more difficult to make quick decisions, but it has the merit of making the design decisions those of the staff as a whole. Sometimes the sets of tasks in a system can be distinguished and those people involved in each task set make their own design decisions.

Effective Technical and Human Implementation of Computer-based Systems (ETHICS) is a participative systems design methodology and, as well as being an acronym, the name is meant to imply that it is a methodology that embodies an ethical position (Mumford & Weir 1979). As well as being a methodology based on the philosophy of the participative approach, it encompasses the socio-technical view, that for a system to be effective the technology must fit closely with the social and organisational factors. In particular, this means that an improved quality of working life and enhanced job satisfaction of the users must be a major objective of the systems design. This is not simply to guard the interests of the users in the introduction of computing and technology, although this is obviously of major importance, but it is an essential prerequisite to achieve effective systems as far as the organisation and its

management is concerned. There needs to be a good 'fit' between what the employee is seeking from his work - his job needs, expectations and aspirations - and what he is required to do in his job - the organisational job requirements which mould his experience'.

Parsons & Shils (1951) identify five areas of measurement, identified as follows:

- *The knowledge fit* - a good fit exists when the employee believes that their skills are being adequately used and that their knowledge is being developed to make them increasingly competent.
- *The psychological fit* - a job must fit the employee's status, advancement and work interest (some of the Herzberg (1966) motivators). These needs are recognised to vary according to age, background, education, and class.
- *The efficiency fit* - this comprises three areas. Firstly, the effort-reward bargain, which is the amount an employer is prepared to pay (as against the view of the employee about how much he is worth). Secondly, work controls, which may be tight or loose but need to fit the employee's expectations. Thirdly, supervisory controls, such as the necessary back-up facilities, including information, materials, specialist knowledge, and supervisory help.
- *The task-structure fit* - this measures the degree to which the employee's tasks are regarded as being demanding and fulfilling. Particularly important are the number of skills required, the number and nature of targets, plus the feedback mechanism, the identity, distinctiveness, and importance of tasks, and the degree of autonomy and control over the tasks that the employee has.
- *The ethical fit* - this is also described as the social value fit and measures whether the values of the employee match those of the employer organisation.

The smaller the difference of the system from these measurements (sometimes referred to as the variance), then the more the users are likely to like the system and the more successful the information system is likely to be.

Participation is seen as vital in ETHICS (Hirschheim, 1985). Whereas in other methodologies participation is sometimes simply regarded as 'allowing the users to choose the colour of the terminals that they use', in ETHICS users are involved in the decisions concerning the work process and how the use of technology might improve their job satisfaction.

Mumford (1983a) recognises that change causes difficulty:

> 'all change involves some conflicts of interest. To be resolved, these conflicts need to be recognized, brought out into the open, negotiated and a solution arrived at which largely meets the interests of all the parties in the situation....successful change strategies

require institutional mechanisms which enable all these interests to be represented, and participation provides these'.

Depending on which sources are referred to, the actual steps of ETHICS differ somewhat in their names and numbers, however the content remains much the same. The fifteen step version outlined here corresponds to that described in Mumford (1983a and 1986). A six stage, twenty five step version of ETHICS is described in Hirschheim (1985). The major difference is that there is a greater separation of the technical and social issues. This might mean that the technical issues of design could be addressed by a separate, more technically experienced, design group. ETHICS has also been amended slightly, although the principles are the same, for use by small businesses thinking of purchasing a computer system for the first time. The work in the steps that follow is performed by the design group or groups themselves.

1. Ask, why change.
2. Establish system boundaries.
3. Describe the existing system.
4, 5 and 6. Define key objectives and tasks.
7. Diagnose efficiency needs.
8. Diagnose job satisfaction needs.
9. Analyse future needs.
10. Specify and weigh efficiency and job satisfaction needs and objectives.
11. Design organisational aspects of the new system.
12. Specify technical options.
13. Prepare detailed work design.
14. Implementation.
15. Evaluation.

Mumford has described a number of experiences using ETHICS (Mumford, 1985a, 1985b and 1985c). One example concerns a group of secretaries at Imperial Chemical Industries (ICI) in the UK who designed new work systems for themselves in the wake of the introduction of word processing equipment and another concerns a group of purchase invoice clerks who helped design a major on-line computer system.

Other methodologies, though not based on participation in the same way as ETHICS, could be termed people-oriented (rather than process or data oriented). For example, ISAC (Lundeberg et al, 1982 and Lundeberg, 1982) seeks to identify the fundamental causes of users' problems. The approach is designed to analyse users' problems and to solve aspects of them where appropriate.

2.6 PROTOTYPING AND AUTOMATION

The theme of automating some aspects of the information systems development process is an abiding one, and has intrigued developers for many years. It has often been recognised that certain aspects of the process are repetitive or rule based and therefore susceptible to automation. Early examples include decision table software which could generate accurate code directly from a decision table, project control packages to help organise and control the development process, and report generators to help speed up some programming tasks. On the analysis side, software has existed for many years to help in the construction of traditional programming flowcharts.

A variety of more ambitious projects are well documented (see, for example, Yadav, 1983) and these include ADS (Accurately Defined System), which was a system definition and specification technique developed by NCR in the 1960s. Five forms were provided which were used to specify outputs, inputs, computations, files and process logic in the form of decision tables. Efforts were made to automate some aspects of ADS by using programs which interpreted the data on the forms to produce information and for cross-referencing.

Systematics, see Grindley (1966 and 1968) was a more ambitious project which attempted to incorporate, amongst other tools, automated decision table interpretation into an automated methodology for information systems design

TAG (Time Automated Grid), see IBM (1971), was another attempt to automate some aspects of the systems analysis process. This was produced by IBM in 1966. Details of the required outputs of a system were fed into TAG and the package would work backwards to determine the inputs necessary to produce those outputs and in what sequence. Reports were produced which helped the analyst in the design process.

Perhaps the most important attempts in the 1960s and early 1970s were carried out at the University of Michigan which led to the ISDOS (Information System Design and Optimization System) project. The basis of ISDOS was PSL and PSA. Problem Statement Language (PSL), which is similar to structured English (Section 5.3.4), enables the analyst to state the requirements in formal terms (indeed in machine-readable form) which are then entered into a kind of database. The PSL statements are then analysed by the Problem Statement Analyzer (PSA) which has similarities with a data dictionary system (Section 5.4.2) and which interprets these statements, validates the data, stores it and analyses it, and then produces the output required. The PSL and PSA together produced a set of documents for the project including the systems analysts' formal specification and the users' requirements. These languages are described in Teichroew & Hershey (1977), Teichroew et al (1979) and Welke (1987). The emphasis of the approach is on the documentation aspects, for they help the manual tasks of systems analysis rather than genuinely automating the processes of analysis

and design.

ISDOS attempts to link all aspects of systems development, from a computerised problem statement into programming language statements. System requirements are input using PSL and PSA but ISDOS also incorporates an Optimizer, so that the code generated is as efficient as possible, and a file generation sub-system. ISDOS had as an aim the incorporation of all aspects of systems work in a complete automated package, but this proved over-ambitious at the time. Indeed, the ideal of converting a user specification in natural language automatically into verified software is still a long way off.

Whilst many of these attempts at automating various aspects of systems development can be seen as far sighted and valiant, they were not overly successful in terms of numbers of users and applications. It has often been noted that data processing professionals have been very keen to automate everybody else but have shown a certain reticence to 'take their own medicine'. Apart from innate conservatism, there appear to have been a variety of reasons for this: some of the software was not very good or easy to use; it was often found that the benefits were outweighed by the effort and costs involved; and the technology was also a limiting factor.

Recently, some of these factors have changed. The technology is clearly more powerful, cheaper and more widely available. Improved graphics facilities have also had an impact. The quality of the software has improved, and in general there is a growing climate of opinion that believes that automated tools are feasible and beneficial.

However, none of the above systems is in widespread use. An approach which is in more common use is that of prototyping. A prototype is an approximation of a type that exhibits the essential features of the final version of that type. Prototyping is common in areas such as engineering where details of a design are worked out and tested thoroughly first. The reference to engineering is appropriate, as prototyping belongs to an experimental engineering paradigm rather than the conventional deterministic methods. Prototyping was not easily carried out satisfactorily until the availability of software tools, in particular fourth generation systems, which greatly reduce the costs associated with prototyping. Boar (1984), Budde et al (1984), Lantz (1985), Law (1985), Naumann & Jenkins (1982), amongst many others, evidence the interest in prototyping during the middle 1980s. Rock-Evans (1989) provides a recent evaluation of some of the various products available that support the approach.

As well as approaches designed around prototyping, some methodologies provide tools which can be used for prototyping. Macdonald (1987) has a number of examples where the use of the Information Engineering Facility (IEF), a software tool supporting Information Engineering (IE), has in places changed the Information Engineering methodology itself. IE provides one example of a methodology incorporating

prototyping as one of the many building blocks.

By implementing a prototype first, the analyst can show the users inputs, intermediary stages, and outputs from the system. These are not diagrammatic approximations, which tend to be looked at as abstract things, or technically-oriented documentation, which may not be understood by the user, but the actual figures on computer paper or on terminal or workstation screens. The formats can be changed quickly, as the users suggest changes, until the users are given a reasonable approximation of their requirements. It may only be by using this technique that the users discover exactly what they want from the system, as well as what is feasible. It is also possible to try out a run using real data. It is therefore a way of attacking three of the criticisms of the conventional approach (Section 2.2): user dissatisfaction, inflexibility and inappropriate documentation.

A prototype is frequently built using special tools such as screen painters and report generators which facilitate the quick design of workstation screens and reports. The user may be able to see what the outputs will look like in a day. Whereas a hand drawing of the screen layout will need to be drawn again for each iteration which leads to a satisfactory solution, the prototype is quickly drawn (or painted) again using the tools available. The ease and speed with which prototypes can be modified are as important as the advantages gained from building the prototype in the first place, iteration becomes a practical possibility. A fourth generation system (Section 5.4.3) is likely to have a whole set of facilities including a database management system, data dictionary, query language, screen generator, report generator, and program generator.

Frequently a prototype system can be developed in a few days, and it rarely takes more than 10% of the time and other resources necessary to develop the full operational system. This can be a good investment of time. Some systems teams use the prototype as the user sign-off. This is likely to be a much better basis for obtaining a user decision than the documentation of conventional systems analysis mentioned in Section 2.2.

One possible drawback of prototyping is that the users may question the time taken to develop an operational system when that taken to develop a prototype was so quick. There is also the risk that the system requirements may change in the meantime. Some users may also argue that the time, effort and money used to develop a prototype is 'wasted'. It is sometimes difficult to persuade busy people that this effort does lead to improved information systems.

Another question regarding prototyping in use, which can also condition user responses, concerns the position of the user in relation to the learning curve when developing a prototype. Frequently the user is at the bottom of the learning curve. In other words user reaction to a prototype is established too early. If the user were able to

climb to a higher position on the learning curve the response to the systems design might well be different. Conversely, sometimes the user is too high on the learning curve, and is asked for reactions when the real analysis has been done and the prototype is in a fairly fixed form and change would be costly and lead to considerable delays. In this circumstance there is sometimes a 'conspiracy' between analyst and user to change only very superficial elements of the prototype, such as the placement of headings or the format of date fields on a report. These considerations relating to the learning curve might imply that an incremental development model may be the most useful way of prototyping, because the users are learning at the same time as the analysts. The approach, like any other, then, is not a panacea, it has to be used carefully and appropriately. A discussion of ways to control prototyping so that it might be used more effectively can be found in Avison & Wilson (1991).

Prototyping may be more than just another tool available to the analyst. It could be used as a basis for a methodology of systems development in the organisation. This may have:

- An analysis phase, designed to understand the present system and suggest the functional requirements of an alternative system.
- A prototyping phase to construct a prototype for evaluation by users.
- A set of evaluation and prototype modification stages.
- A phase to design and develop the target system using the prototype as part of the specification.

Prototyping as part of the systems development cycle is discussed in Dearnley & Mayhew (1983).

Many prototypes are intended to be discarded. They have not been designed to be used as operational systems being used only as a development tool, as a learning vehicle. Prototypes lack features which are essential in an operational system but inappropriate in a prototype, and this needs to be stressed to users who may expect the target system to be developed in the same time as the prototype. Reviews of prototyping can be found in Hekmatpour & Ince (1986) and Mayhew & Dearnley (1987).

An alternative approach is to use the final prototype as the basis for the operational system. In this scenario, the system has 'evolved' by an iterative process. Once the users are satisfied with the prototype, it can be converted to become an operational system. It is now no longer a prototype. Evolutionary information systems development is similar except that there is never a 'final version', there is always the possibility of iterative revision even when the system is live. In this approach the prototype may always develop in an evolutionary systems development process and may never 'die'. The prototype is changed to reflect new conditions. Here, maintenance

is regarded as positive rather than negative (see McCracken & Jackson, 1982), the likelihood of change is catered for, rather than being seen as a problem. This evolutionary approach therefore addresses the problem of dealing with change, an aspect which many other approaches do not address, although a four stage future analysis is discussed in Land (1982) - work which is often incorporated into participative approaches.

2.7 STRUCTURED APPROACHES

Structured methodologies are based on functional decomposition, that is the breaking down of a complex problem into manageable units in a disciplined way. The development of structured methodologies in systems analysis and design stemmed from the perceived benefits of software engineering. Structured programming has been successful in increasing programmer productivity and ensuring that the programming code produced is easier to maintain. But programs, however well produced, may be useless if the analysis and design that preceded it was poorly carried out. Further, even good analysis and design, if poorly documented, may lead to incorrect interpretation by programmers and even the best programming techniques cannot overcome such problems.

Structured systems analysis and design has been associated with a number of authors including DeMarco (1979), Gane & Sarson (1979), Myers (1975 and 1978), Weinberg (1978), Yourdon & Constantine (1978) and Yourdon (1989). These tend to stress techniques such as decision trees, decision tables, data flow diagrams, data structure diagrams, and structured English, and tools such as data dictionaries. These prove to be a considerable improvement on the conventional techniques such as flowcharting, both from the point of view of understanding the real-world processes and in communicating the knowledge acquired. Most of the documentation aids are graphic representations of the subject matter. This is much easier to follow than text or computer-oriented documentation.

Many of these techniques have a strong characteristic of structure. The data flow diagram, for example, enables a system to be partitioned into independent units. The technique of functional decomposition enables a series of data flow diagrams to be drawn, at the top level, a context diagram, and at lower levels more and more detail. For example, the next level may show only the basic activities and flows, and a further level show the detail of validation or the processing of exceptions. This structure enables an easier understanding of both the existing system and the planned system that they may model. This leads to better communications and should also lead to more reliable systems, thereby reducing the maintenance burden.

Structured analysis documentation includes documents describing the logical (real-world) analysis of the processes and not just their physical (implementation) level designs. In other words, there is usually a clear distinction between any application logic and the computer representation of that logic. There will also be a separation of any data or information that the system is likely to input, output, process or store and the physical record - a computer file or part of a database. This separation gives a level of 'data independence', in other words the processes can change without necessarily changing the computer files. Similarly, the files can change without necessarily altering the user views of the data.

As well as improved tools, structured methodologies usually incorporate 'structured walkthroughs'. These are team meetings where the analysis and design specifications and other documentation (which will have been previously circulated) are exposed to review by the members of the team. Usually the group explores problems rather than looks for solutions (although some work this way). A peer review is likely to reveal errors, omissions, ambiguities, inconsistencies and weaknesses in style of documentation, and also ensure a process of continuous training of the staff involved. By having walkthrough sessions at each stage, requirements definition before analysis, analysis before design, and design before implementation, this should avoid the late detection of errors and flaws in logic, and hence greatly reduce the risk of failure when the system is implemented.

Gane & Sarson (1979) suggest four major steps in their methodology (STRADIS):

- Initial study.
- Detailed study.
- Defining and designing alternative solutions.
- Physical design.

The initial study is an attempt to ensure that the systems chosen to be developed are those that most warrant development in a competing environment. The most important criterion in this selection process in this approach is the monetary costs and benefits of each proposal. A decision is made on whether to proceed to a more detailed study. This takes the work of the initial study further. In particular the existing system is examined in detail. Boundaries are drawn to define the scope of the project. Gane & Sarson describe in detail the drafting of data flow diagrams at various levels, showing how each level is exploded into lower levels through to the level where the logic of each process box in the lowest level can be specified using the appropriate process logic representation, for example, decision trees, decision tables, or structured English (see Chapter 5). The detailed study contains:

- A definition of the user community for a new system.
- A logical model of the current system.

- A statement of increased revenue/avoidable cost/improved service that could be provided by an improved system.
- An account of competitive/statutory pressures (if any).

The third stage defines alternative solutions to the problems of the existing system, taking into account the organisational objectives which are converted into a set of detailed system objectives. Analysts and designers then work together to produce various alternative implementation designs which meet a variable selection of the identified system objectives. The report of this phase of the project should be presented to the relevant decision makers and a commitment made to one of the alternatives.

At the final stage, the design team refines the chosen alternative into a specific physical design which involves a number of parallel activities, for example, specifying detailed processing requirements, designing the physical files or database, and defining the clerical tasks. Subsequent phases of the methodology are not clearly defined by Gane & Sarson as the methodology is effectively concerned mainly with analysis, to a lesser extent design and hardly at all with implementation.

The approach of DeMarco (1979) has the following steps:

- Building a logical model of the current system.
- Building a logical model of a future system.
- Creating a number of physical models.
- Choosing one model.
- Packaging the structured specification.

Structured systems analysis and design can be considered as an alternative to the conventional approach, having the same philosophy but more up to date tools and techniques. In fact this is a simplification. The authorities on structured analysis do not view it in the same way. The techniques can be regarded as a useful alternative to many techniques used in the conventional approach, and therefore could be incorporated in that approach rather than replace it. A structured methodology may also replace the conventional approach. Some writers emphasise the analysis aspect, such as DeMarco, whilst others also stress design aspects. Writers also emphasise different techniques, although some techniques are common to all structured methodologies.

2.8 DATA ANALYSIS

Whereas structured analysis and design emphasises processes, data analysis concentrates on understanding and documenting data which, it is argued, represents the 'fundamental building blocks of systems'. It involves the collection, validation and classification of the entities, attributes and relationships that exist in the area investigated. Even if applications change, the data already collected may still be relevant

to the new or revised systems and therefore need not be collected and validated again. This is a response to the criticism of the conventional approach in that, being process oriented, it will need to be changed frequently, causing a heavy maintenance burden. The data model, the result of data analysis, is oriented towards that part of the the real world it represents (organisation, department, or whatever) and should be implementation independent. This means that data analysis is suitable whether the physical model is a database, file or card index, though it is frequently associated with databases, as in, for example, Avison (1985) and Robinson (1989).

The interest in data analysis stems partly from the problem of 'graduating' from computer files to databases. What may be called 'informal data analysis', an ad hoc approach to understanding the data structures in an organisation, adequate for file processing systems, is not adequate for database applications where the data is shared between applications and therefore the data and its structure must be well defined. Early database experience did not always bring about the expected flexibility of computer applications, usually because the database was not a good reflection of the organisation it was supposed to represent. Modelling the organisation on a computer database is not easy. It is argued that data analysis has proved successful in creating a model which is independent of any database system, accurate, unambiguous and complete enough for most applications and users. Its success comes in the systematic way by which it identifies the data in organisations and, more particularly, the relationships between these data elements, the 'data structure'.

Data analysis techniques (Howe, 1983) attempt to identify the data elements and analyse the structure and meaning of data in the organisation. This is achieved by interviewing people in the organisation, studying documents, observation, and so on, and then formalising the results. A number of documentation aids, many of which are pictorial, also help in the process of data analysis. This becomes the first step in extrapolating the complexities of the real world into a model that can be held on computer. The documentation produced in data analysis is more easily understood by users than the documentation for data representation used in the conventional approach, such as the file and record specification forms.

Data analysis does not necessarily precede the implementation of a database or computer system of any sort. It can be an end in itself, to help in understanding a complex organisation. Good models will be a fair representation of the 'real world' and can be used as a discussion document for improving the effectiveness of the role of data in an organisation.

Avison (1981) describes a number of alternative approaches to data analysis. In the data collection approach, frequently referred to as document-driven analysis, the documents used in a department are analysed. Such documents include reports, forms

and enquiry formats. The analysis of each document in turn will lead to the formation, and then improvement, of a data model showing the data and the relationships between these data. The particular structures arrived at are relations, and rules are applied to the set of relations to ensure that the model is flexible for future use, such as its mapping on to a database. These are known as the rules of normalisation (Section 5.3.2). The result of this type of data analysis is therefore a data model represented as a set of normalised relations.

The document-driven approach is a useful contribution, as documents in organisations are usually easy to analyse. But organisations are not always fully represented by the documents used in that organisation. Frequently the number of documents is such that they would take too long to analyse. An alternative approach to data analysis, entity modelling, gains its information by interviewing people in the organisation, such as department managers and manual staff. Entities such as customers, suppliers, parts and finished goods, are identified and the relationships between these entities are also ascertained. The entities and their relationships can be expressed as a graphical model. This is a more common approach to data analysis.

The fourth phase of SSADM (NCC, 1986 and Downs et al, 1988), detailed data design, employs both approaches to data analysis to produce a well-checked 'composite data model'. Although SSADM is a data driven methodology, it has elements of a number of themes discussed in this chapter. The six phases to SSADM are:

- Analysis of the current system.
- Specification of the required system.
- User selection of service levels.
- Detailed data design
- Detailed procedure design.
- Physical design control.

Data analysis is also a main feature in the French methodology Merise (Quang & Chartier-Kastler, 1991 and Collongues et al, 1989).

Although Information Engineering (Martin & Finkelstein, 1981) and its precursor D2S2 (Macdonald & Palmer, 1982) are not entirely data analysis oriented, their advocates stress the stability of data, and data analysis plays a key role in these methodologies. Additional arguments for this approach are:

- The data model is not computer orientated.
- It is a model which is readily understandable by technologists, and managers and users can far more readily appreciate the meaning of the data relationships illustrated in a graphical form than they could in the file and record specification forms of conventional systems analysis.
- It does not show bias towards particular users or departmental view: The data

model can reflect a variety of different views of the data.

- Although the model can represent the organisation as a whole, it can be adapted so that a particular part of the model can appear to different users.
- The data model is readily transformable into other models, such as relations, hierarchies or networks, which are often but not exclusively the ways in which database management systems require the data structures to be presented.

The approach does have critics. Some argue that although the documents and the techniques used are easier to understand by users when compared to some traditional methods, data analysis does not lend itself fully to user participation. Others have argued that the emphasis on data may be misplaced. Further, the assumption in data analysis is that it is possible to model reality, is questionable, see Kent (1978). The data model can only be 'A' model and not 'THE' model of that part of the real-world being modelled. It cannot reflect reality completely and accurately for all purposes. Even if data analysis has 'gone according to plan', the resultant data model cannot objectively represent the organisation. It is a subjective view distorted by the perceptive process. Nevertheless, the data model derived from data analysis usually proves in practice to be suitable for the purpose.

For many, data analysis is one-sided and although it is beneficial to develop a data model which can be mapped on to a database, there is still a parallel need to understand the functions that will be applied to the database when it is implemented. Indeed most data analysis-based methodologies now address process aspects as well, and incorporate many of the techniques discussed in Section 2.7.

2.9 MULTIVIEW: A BLENDED CONTINGENCY APPROACH

At the beginning of this chapter there was a short discussion on methodologies and methodology comparisons. There are a large number of methodologies, and this reflects different viewpoints, cultures and experiences, though many methodologies are similar. This chapter has attempted to provide a schema for understanding and classifying methodologies by identifying six broad themes.

The systems approach would suggest that attempting to develop information systems for the widest possible context would be advantageous, as developing an information system for a small part of the organisation might be to the detriment of the organisation as a whole. It highlights the importance of the relationship between an organisation and its environment and in multi-disciplinary teams to understand organisations. Soft systems methodology offers help in understanding these larger and complex problem situations. Planning approaches can involve strategic management in information systems work so that their needs are analysed and that information systems

are implemented which do more than computerise the operations level applications. Like the systems approach, planning approaches attempt to meet the needs of management. They help to ensure that systems are not implemented in a piecemeal fashion and that there is overall planning for information systems development. Participation goes further in that all users are expected to contribute to and gain from any information system, and thereby increase the likelihood of its success. This should help to prevent technically viable information systems failing because of 'people problems'. Prototyping enables users to comment on the proposed information system, its inputs, processing and outputs, before the system has been designed in its final form and therefore, like participation, reduce the likelihood of user dissatisfaction. It avoids the inflexibility of the output driven design of the conventional approach. Further, through the use of prototyping tools, the application backlog can be reduced by the greater speed of information systems development. Structured approaches aid the understanding of a complex problem through functional decomposition and the associated documentation techniques. This should reduce the maintenance burden. Data analysis is a useful modelling technique and the data model produced is likely to be relevant for a longer period than models of processes which can be unstable. It is also an important stage in the creation of a computer database and this can increase the viability of data sharing and thereby reduce the likelihood of inconsistency and inflexibility of information systems.

In general, each of these themes is a response to some of the weaknesses of the conventional approach. But no methodology has proved to be a panacea. Indeed it could be argued that some are inferior to the conventional approach in some ways, in that they do not all stress some of the important aspects of that approach, for example, thorough and detailed documentation standards for communication, training and so on.

Further, each of these six general approaches has been attacked on a number of fronts (Avison & Fitzgerald, 1988b). For instance, systems theory has been attacked on the grounds that it represents an ideal academics' position and is not relevant to the practitioner. It expands the problem boundaries too far, leading to revolutionary change which is not appropriate for most organisations. Further, systems approaches do not describe methods for implementing solutions. They might help to understand problem situations, but they do not provide a scheme for solving problems. Planning approaches do not in themselves address the need for participation of all users. Further, they tend to be mechanical and inflexible following the identification of strategic needs. Participative approaches might lead to inefficient systems designed by those who are good managers, clerks, or salesmen, but poor, and unwilling, systems analysts. It has been the author's experience that managers have sometimes refused to participate fully, arguing that the consultant was the systems expert and should be designing good

information systems for them. Many analysts using prototyping methods concern themselves with the user interface and do not address the fundamental problems of systems analysis. They are simply making poor systems palatable to users. For example, rearranging headings and columns of a report may give a more aesthetically pleasing report layout containing data which still does not help the decision maker. Further possible disadvantages of prototyping include the relative times necessary to construct a prototype and that for the operational system. In breaking down a system into manageable units and then even more manageable units, structured analysis may offer a simplistic view of a complex system and fail to identify fully the importance of the links between subsystems. Finally, data analysis may not solve the underlying problems that the organisation might have. Indeed it may have captured in the data model the existing problems of the firm, and make them even more difficult to solve in the future.

It is therefore unreasonable perhaps to rely on any methodology falling into any one of these themes. As well as the criticisms mentioned above, there are other arguments which would suggest that one approach can never be the full answer (Avison & Wood-Harper, 1986):

- The tools and techniques appropriate for one set of circumstances may not be appropriate for others.
- The 'fuzziness' of some applications require an attack on a number of fronts. This exploration may lead to an understanding of the problem area and hence lead to a reasonable solution.
- As an information systems project develops, it takes on very different perspectives or 'views' and any methodology adopted should incorporate these views, which may be organisational, technical, economic, and so on.

Multiview is an explorative structure, a loose framework with which to define and develop information systems. Multiview incorporates many of the principles, techniques and tools of a number of the themes discussed in this chapter into a blended approach or meta-methodology. It is also a contingent approach, in that alternative techniques and tools are available which may be chosen according to the dictates and requirements of each situation. The user of Multiview in effect produces from this structure a unique method for every project. It therefore follows Iivari's view of a contingency approach (within the methodology) rather than that of Davis, who argues for a range of methodologies to be available, one chosen according to the particular situation (Section 1.1.3).

Some authors have argued for a 'tool kit approach' to systems development without a framework, such as that which Multiview provides. The flexibility provided by a 'cut and paste' approach to developing information systems has many advantages, in

particular the avoidance of a 'one best way' approach which leads to 'elaborate and bureaucratic methodologies' (Benyon & Skidmore, 1987). But there are many problems with the tool kit approach, some of which are readily identified by management services people in the real world. Avison, Fitzgerald & Wood-Harper (1988) suggest that the 'DIY analysts' who followed this approach would be akin to the 'tradition and common sense analysts' that operated before information systems development methodologies were adopted, producing idiosyncratic and unmaintainable systems of variable value. The tools themselves are useful, and they might be appropriate in different situations, but choosing which tools are appropriate (and when they are appropriate) is a very skilled job, and those with these skills are few and far between.

A reflective practitioner, who may not want to follow a rigid methodology, still needs to use a coherent approach. The Multiview approach is a contingency approach offering alternatives, but it has been designed so that the techniques and tools hang together within a framework. This is described in Chapter 3.

Chapter 3
Early Multiview

3.1 AN OVERVIEW OF MULTIVIEW

As we saw in Chapter 2, the chase for the perfect methodology is somewhat illusory, because different methodologies represent different views of the world. Information systems design could be seen as a logical, technical or people problem. Different analysts have adopted different methods because they have taken a different view of the situation. There are also differences in the systems analysis and design approaches used which are caused by differences in the situation in which the analyst is working. Approaches that may be successful in a large bureaucratic organisation may well be different to those which work in a small fast moving company.

There exists in actual systems analysis and design practice a three-way relationship between the analyst, the methodology and the situation, shown in Figure 3.1, but parts of the relationship are missing in many expositions of information systems development methodologies. For example, many methodologies assume - though the assumption is not stated - that each situation is essentially the same and that analysts are similar in background and experience.

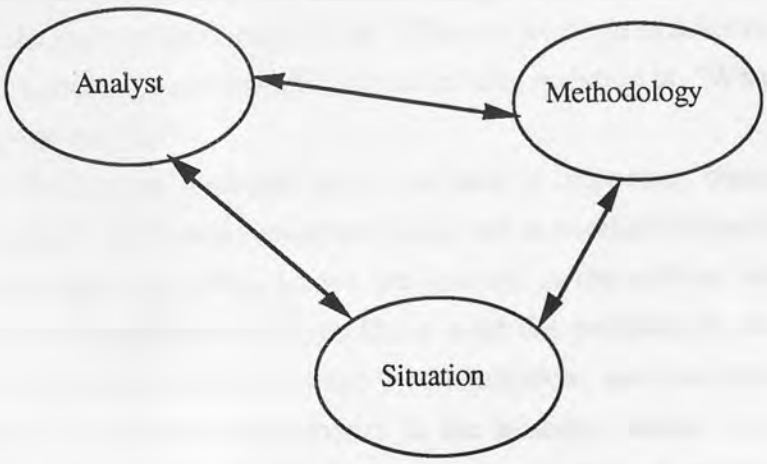


Fig. 3.1. The interaction between the analyst, the problem situation and the methodology

This chapter describes an approach to information systems development which combines important aspects of some of the major methodologies and themes discussed

in Chapter 2 into a coherent and yet flexible approach. The version of Multiview presented here is adapted from that found in Wood-Harper, Antill & Avison (1985). The methodology covers five different stages of systems analysis and design, each with its own appropriate view of the problem, and each with methods for tackling that aspect of the problem.

The stages of the Multiview methodology and the inter-relationships between them are shown in Figure 3.2. The boxes refer to the analysis stages and the circles to the design stages. The arrows between them describe the inter-relationships. Some of the outputs of one stage will be used in a following stage. The dotted arrows show other major outputs. The five stages are:

1. Analysis of human activity.
2. Analysis of information (sometimes called information modelling).
3. Analysis and design of socio-technical aspects.
4. Design of the human-computer interface.
5. Design of technical aspects.

They incorporate five different views which are appropriate to the progressive development of an analysis and design project, covering all aspects required to answer the vital questions of users. These five views are necessary to form a system which is complete in both technical and human terms. The outputs of the methodology, shown as dotted arrows in Figure 3.2, are listed in Figure 3.3, together with the information that they provide and the questions that they answer.

Because it is a multi-view approach, it covers computer-related questions and also matters relating to people and business functions. It is part issue-related and part task-related. An issue-related question is: "What do we hope to achieve for the company as a result of installing a computer?". A task-related question is: "What jobs is the computer going to have to do?".

The distinction between issue and task is important because it is too easy to concentrate on tasks when computerising, and to overlook important issues which need to be resolved. Too often, issues are ignored in the rush to 'computerise'. But, you cannot solve a problem until you know what the problem is! Issue-related aspects, in particular those occurring at stage 1 of Multiview, are concerned with debate on the definition of system requirements in the broadest sense, that is 'what real world problems is the system to solve?'. On the other hand, task-related aspects, in particular stages 2-5, work towards forming the system that has been defined with appropriate emphasis on complete technical and human views. The system, once created, is not just a computer system, it is also composed of people performing jobs.

Another representation of the methodology, rather more simplistic, but useful in providing an overview for discussion, is shown in Figure 3.4. Working from the

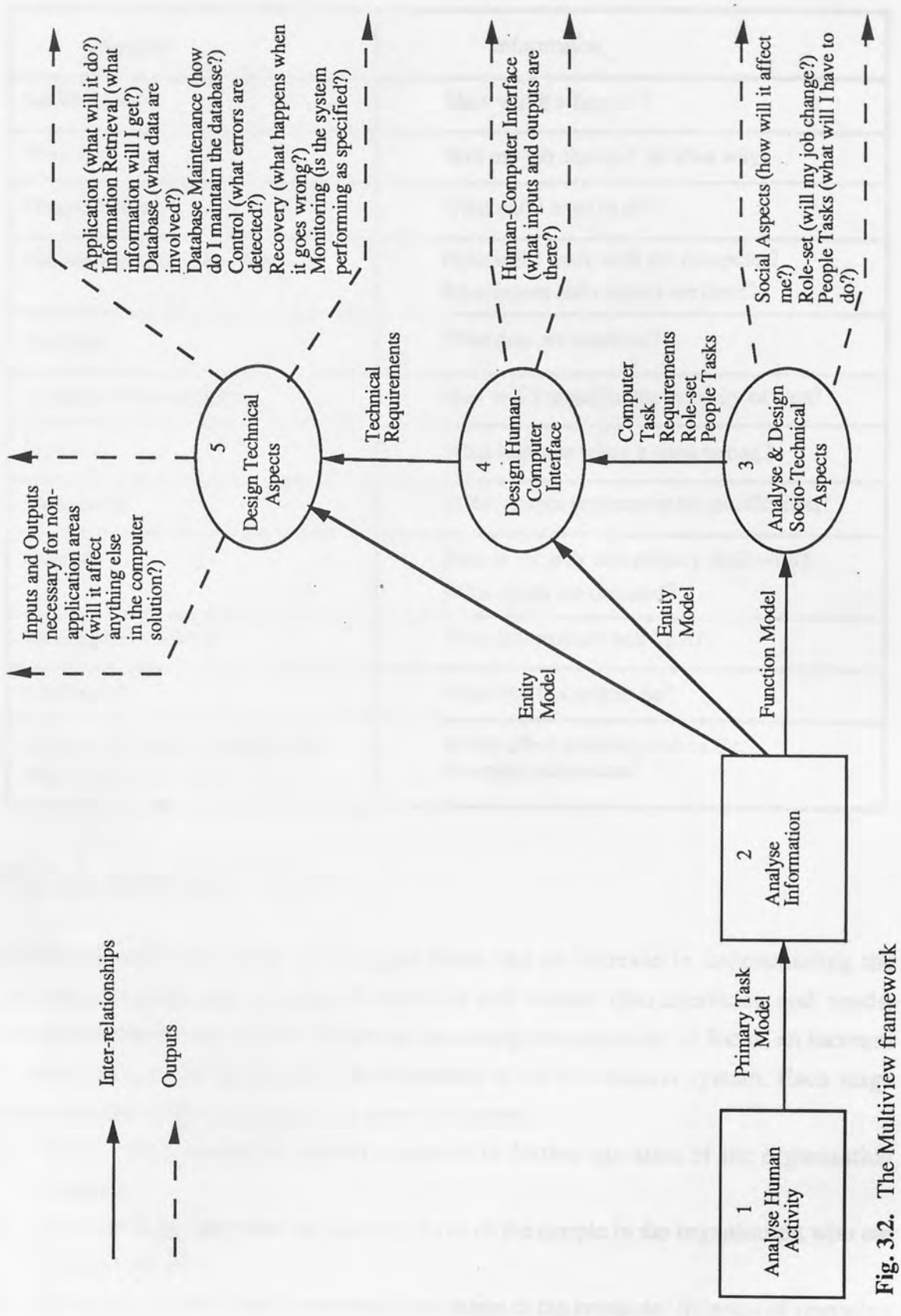


Fig. 3.2. The Multiview framework

Outputs	Information
Social Aspects	How will it affect me?
Role-set	Will my job change? In what way?
People Tasks	What will I have to do?
Human-Computer Interface	How will I work with the computer? What inputs and outputs are there?
Database	What data are involved?
Database Maintenance	How will I maintain the integrity of data?
Recovery	What happens when it goes wrong?
Monitoring	Is the system performing to specification?
Control	How is security and privacy dealt with? What errors are detected?
Information Retrieval	What information will I get?
Application	What will the system do?
Inputs and Outputs necessary for non-Application Areas	Will it affect anything else on the computer subsystem?

Fig. 3.3. Methodology outputs

middle outwards we see a widening of focus and an increase in understanding the problem situation and its related technical and human characteristics and needs. Working from the outside in, we see an increasing concentration of focus, an increase in structure and the progressive development of an information system. Each stage addresses one of the following five questions posed:

1. How is the information system supposed to further the aims of the organisation using it?
2. How can it be fitted into the working lives of the people in the organisation who are going to use it?
3. How can the individuals concerned best relate to the computer in terms of operating it and using the output from it?
4. What information processing function is the system to perform?
5. What is the technical specification of a system that will come close enough to doing the things that you have written down in the answers to the other four questions?

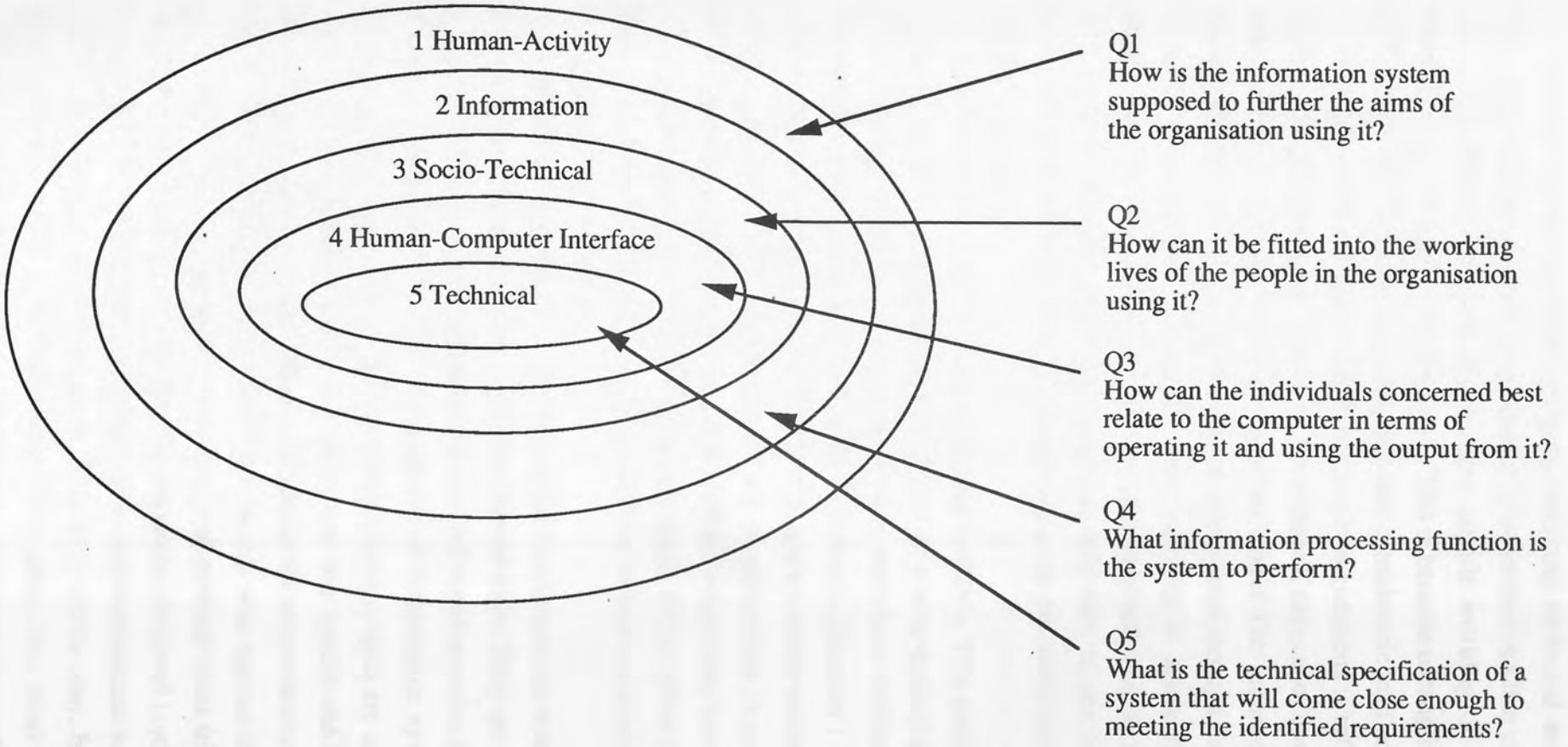


Fig. 3.4. The Multiview methodology

Most computer scientists, programmers and technical analysts are primarily interested in *question 5*. It is their interest in the technology that probably brought them into computing in the first place. Most people working on large systems have developed a great interest in *question 4*. This is because untangling the logic of all the different records that need to be kept provides considerable intellectual satisfaction.

The interest of computer people in these two questions is reflected in the fact that there are large numbers of books on the technical aspects of computer hardware and programming, and an increasing number about the techniques of information modelling, that is determining the flow of information through the system and the way it has to be processed and stored. Most teaching in academic computer science departments has centred around these two aspects of information systems design, and it may be entirely appropriate for computer scientists to specialise in these areas. However, good answers to these two questions are not sufficient to ensure successful information systems.

Question 3 relates to the human-computer interface. This methodology looks at this aspect of information systems at length. Trade unions have taken an interest in *question 2*, even though British and American unions have been behind their Scandinavian counterparts (who have had legislation to back them). *Question 1* is what the managing director has been asking all along. Unfortunately too many senior managers have had trouble with the answers. Either they did not understand the jargon in which proposals from would-be suppliers were couched or bitter experience convinced them that the answers contained more sales talk than substance. All too often managers themselves have not had the right training or advice on how to harness information technology to the needs of their operations.

Information systems are often seen as technical systems which have behavioural and social consequences. This is a rather narrow view. They are better seen as social systems which rely to an increasing extent on information technology. The technology is only a component. This wider perspective of information systems requires that a number of stakeholders in an information systems project are identified. These will include the individuals or group who request the new system and authorise the work to go ahead; the managers, who are responsible for the organisational functions in which the system is embedded; clerical staff and others, who operate the system; computer staff, who create and maintain the system; and external users of the system, such as customers or claimants, who receive the computer-produced invoices and forms. One major objective of Multiview is to help them to communicate with each other about what is needed and what is proposed. This may not be easy, because they may be looking at the system in different ways. One group may think in economic terms, another in terms of status and responsibilities, another in terms of job satisfaction, and

yet another in terms of the nature of contact with the organisation.

The rest of this chapter outlines the five related views of the Multiview methodology.

3.2 STAGE 1 - ANALYSIS OF HUMAN ACTIVITY

This stage is based on the work of Professor Peter Checkland at Lancaster University. It mainly represents an adaptation of his work on soft systems described in Section 2.3.

Checkland has carried out extensive studies, both theoretical and practical, on systems analysis using the concept human activity system (Checkland & Griffin 1970 and Checkland 1981 and 1984). The very general term human activity is used to cover any sort of organisation. This could be, for example, an individual, a company, a department within a larger organisation, a club, or a voluntary body. They may all consider using a computer for some of their information systems.

The central focus of this stage of the analysis is to search for a particular view (or views). This Weltanschauung (sometimes also called 'Assumptions' or 'World View') will form the basis for describing the systems requirements and will be carried forward to further stages in the methodology. This world view is extracted from the problem situation through debate on the main purpose of the organisation concerned, sometimes described using the terms 'raison d'etre', 'attitudes', 'personality' and so on. Examples of world view might be: "This is a business aimed at producing maximum long term profits" or "This is a hospital dedicated to maintaining the highest standards of patient care".

The phases within the methodology of this stage are shown in Figure 3.5. This is an adaptation of Figure 2.1, bringing Checkland's approach within the context of Multiview. The substages can be grouped into four main ones:

1. Perceiving the problem situation (substages 1-3).
2. Constructing systems models (substages 4-7).
3. Comparing the systems models to perceived reality (substage 8).
4. Deciding on the comparison and then implementing the consequences of these decisions (substages 9 and 10).

This is a simplification because there are several iterations not appearing in the diagram. Firstly, the problem solver, perhaps with extensive help from the problem owner, forms a rich picture of the problem situation. The problem solver is normally the analyst or the project team. The problem owner is the person or group on whose behalf the analysis has been commissioned. The 'picture' represents a subjective and objective perception of the problem situation in diagrammatic and pictorial form,

showing the structures of the processes and their relation to each other. Elements of the rich picture will include the clients of the system, the people taking part in it, the task being performed, the environment, and the owner of the system. This picture can be used to help the problem solver better understand the problem situation. It is also a very useful tool from which to stimulate debate, and it can be used as an aid to discussion between the problem solver and the problem owner. There are usually a number of iterations made during this process until the 'final' form of the rich picture is decided. The process here consists of gathering, sifting and interpreting data which is sometimes called 'appreciating the situation'. Drawing the rich picture is a subjective process. There is no such thing as a 'correct' rich picture. The main purpose of the diagram is to capture a holistic summary of the situation. An example rich picture is shown as Figure 4.1, drawn in the context of the case described in Chapter 4. (In general, examples of the techniques used in Multiview are given in case descriptions so as to avoid repetition. It is also easier to illustrate the use of techniques in the context of an actual application.)

One may start to construct a rich picture by looking for elements of structure in the problem area. This includes things like departmental boundaries, activity types, physical or geographical layout and product types.

Having looked for elements of structure, the next stage is to look for elements of process, that is, 'what is going on'. These include the fast-changing aspects of the situation: the information flow, flow of goods, and so on.

The relationship between structure and process represents the climate of the situation. Very often an organisational problem can be tracked down to a mismatch between an established structure and new processes formed in response to new events and pressures.

The rich picture should include all the important hard 'facts' of the organisational situation, and the examples given have been of this nature. However, these are not the only important facts. There are many soft or subjective 'facts' which should also be represented, and the process of creating the rich picture serves to tease out the concerns of the people in the situation. These soft facts include the sorts of things that the people in the problem area are worried about, the social roles which the people within the situation think are important, and the sort of behaviour which is expected of people in these roles.

From the rich picture the problem solver extracts problem themes, that is, things noticed in the picture that are, or may be, causing problems and/or it is felt worth looking at in more detail. The picture may show conflicts between two departments, absences of communication lines, shortages of supply, and so on.

Taking these problem themes, the problem solver imagines and names relevant

systems that may help to relieve the problem theme. By relevant, we mean a way of looking at the problem which provides a useful insight, for example:

Problem theme	=	conflicts between two departments
Relevant systems	=	conflict resolution system or system for redefining departmental boundaries.

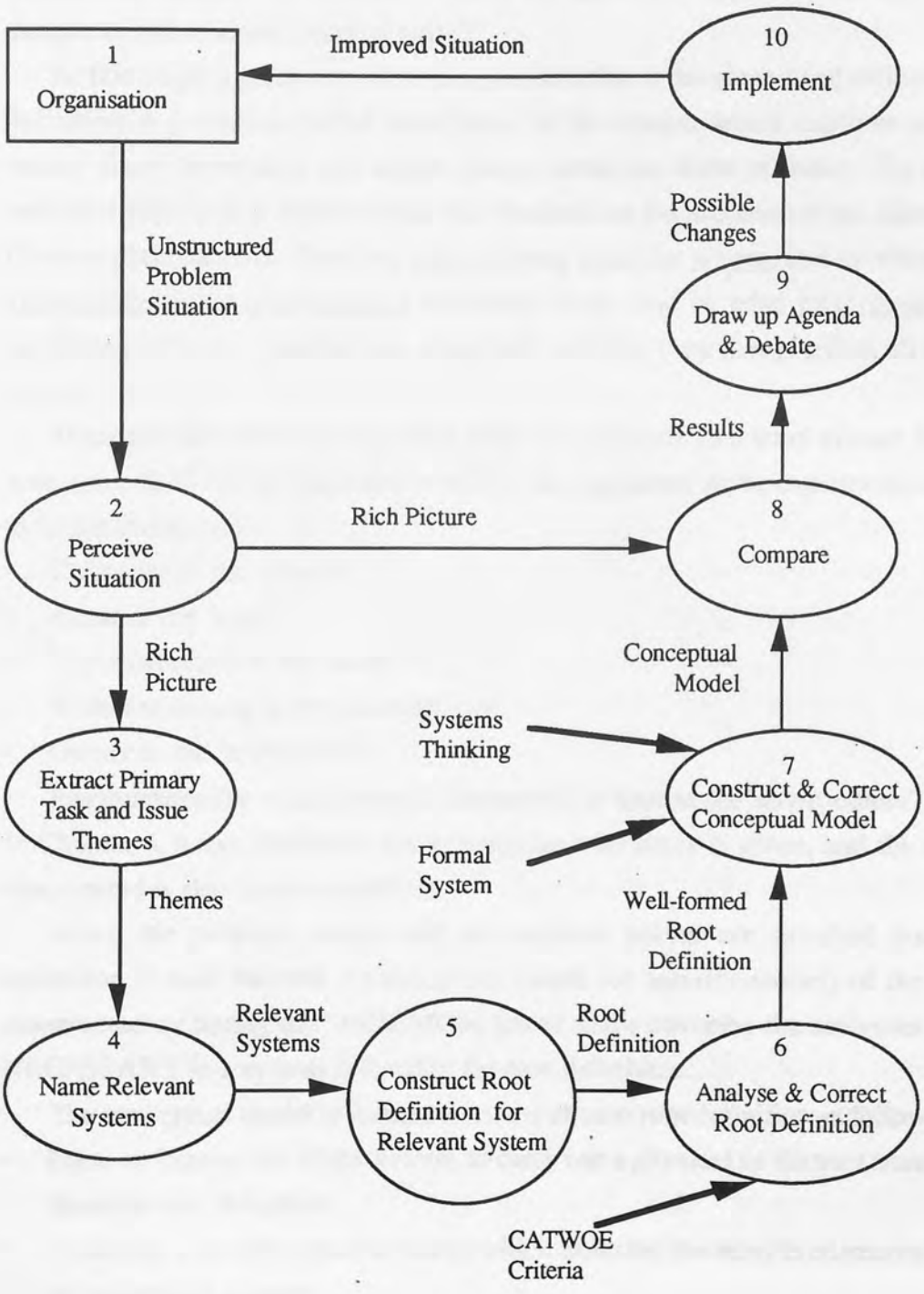


Fig. 3.5. Analysis of human activity systems to depict processes and products

Several different relevant systems should be explored to see which is the most useful. It is at this stage that debate is most important. The problem solver and the problem owner must decide on which view to focus, that is how to describe their relevant system. For example, will the conflict resolution system be 'a system to impose rigid rules of behaviour and decision-making in order to integrate decisions and minimise conflict' or will it be 'a system to integrate decisions of actors through increased communication and understanding between departments' or even 'an arbitration system to minimise conflict between departments by focussing disagreements towards a central body'?

At this stage, a particular view or root definition is developed and refined. The root definition is a concise verbal description of the system which captures its essential nature. Each description will derive from a particular view of reality. To ensure that each root definition is well-formed, it is checked for the presence of six characteristics. Put into plain English, these are who is doing what for whom, and to whom are they answerable, what assumptions are being made, and in what environment is this happening? If these questions are answered carefully, they should tell us all we need to know.

There are technical terms for each of the six parts, the first letter of each forming the mnemonic CATWOE. The order in which they appeared in the explanation is changed to fit this mnemonic:

- Customer is the 'whom'.
- Actor is the 'who'.
- Transformation is the 'what'.
- Weltanschauung is the 'assumptions'.
- Owner is the 'answerable'.
- Environment (or environmental constraints) is kept as the 'environment'.

In Chapter 4, a root definition for a particular case study is given, and the way that it was created is also demonstrated.

When the problem owner and the problem solver are satisfied that the root definition is well formed, a conceptual model (or activity model) of the system is constructed by listing the "MINIMUM list of verbs covering the activities which are NECESSARY in a system defined in the root definition....".

The conceptual model is formed from the chosen root definition as follows:

- Form an impression of the system to carry out a physical or abstract transformation from the root definition.
- Assemble a small number of verbs which describe the most fundamental activities in the defined system.
- Decide what the system has to do, how it would accomplish the requirement and

how it would be monitored and controlled.

- Structure similar activities in groups together.
- Use arrows to join the activities which are logically connected to each other by information, energy, material or other dependency.
- Verify the model.

Aids to forming the conceptual model include the checklist technique of comparing the system described against a 'formal systems model'. This is a compilation of features which have to be present if a set of activities is to comprise a system capable of purposeful activity. This checking process ensures that no vital elements have been excluded. (Note that provided that the root definition has been well formed according to the CATWOE criteria, it is unlikely that discrepancies will occur at this stage.) Other aspects of systems thinking can also be used to ensure that the conceptual model is adequate (Checkland 1984). Illustrations of conceptual models are also given in Chapter 4 (Figures 4.2 to 4.7).

The completed conceptual model is then compared to the representation of the 'real world' in the rich picture. Differences between the actual world and the model are noted and discussed with the problem owner. Possible changes are debated, agendas are drawn up and changes are implemented to improve the problem situation.

In some cases the output of this stage is an improved human activity system and the problem owner and the problem solver may feel that the further stages in the Multiview methodology are unnecessary. In many cases, however, this is not enough. In order to go on to a more formal systems design exercise, the output of this stage should be a well formulated and refined root definition to map out the universe of discourse, that is the area of interest or concern. It could be a conceptual model which can be carried on to stage 2 - the analysis of entities, functions and events.

3.3 STAGE 2 - ANALYSIS OF INFORMATION (ENTITIES, FUNCTIONS AND EVENTS)

The purpose of this stage, which is also known as information modelling, is to analyse the entities and functions of the system described, independent of any consideration of how the system will eventually develop. Its input will be the root definition/conceptual model of the proposed system which was established in stage 1 of the process.

This stage combines the concepts of structured analysis (discussed in Section 2.7) as outlined in DeMarco (1979), Gane & Sarson (1979), Martin & McClure (1984) and Yourdon (1989), for example, and data modelling (discussed in Section 2.8) and outlined in Shave (1981), Howe (1983) and Robinson (1989), for example, with the

practical orientation of Rock-Evans (1981), Macdonald & Palmer (1982), Veryard (1984), and Avison (1985). The concepts utilised are the entity-function models as outlined for a database environment in Davenport (1978).

Two phases are involved: (a) the development of the functional model, and (b) the development of an entity model.

(a) *Development of a functional model*

The first step in developing the functional model is to identify the main function. This is always clear in a well formed root definition. This main function is then broken down progressively into subfunctions, until a sufficiently comprehensive level is achieved. This occurs when the analyst feels that the functions cannot be usefully broken down further. This is normally achieved after about four or five subfunction levels, depending on the complexity of the situation.

As an example, if the main function of an accounts department is to 'provide financial information', it can be broken down into the subfunctions 'keeping books' and 'providing reports'. These are further broken down. A series of functional model diagrams are presented in Chapter 4, in the context of the first case study (Figures 4.8 to 4.16).

The next stage is to think about what events trigger each of these actions (sometimes referred to as operations), and what information is involved. An event (also called a transaction) is a stimulus that initiates a function. The event 'request by tutor' is arrowed in Figure 4.10. There are two types of events: internal and external. An internal event occurs inside the model and is a function of completing or starting. An external event occurs outside the domain of the model. These events can be depicted on the function hierarchies (through the use of arrows) and on data flow diagrams (see Figures 4.17, 4.18, and 4.19).

Data flow diagrams highlight processes or functions and the use of the data. The analyst constructs these data flow diagrams by examining how information flows in and around the function hierarchies. These diagrams create a good basis for the questions that the analyst must ask the user who is doing the job, and who is the source of the analyst's information. The data flow diagram acts as a check on the previous diagrams. By using this diagram, entries missing from the functional chart can be checked. Formally, this checking can also be done by a Function/Event matrix (Figure 4.20). This chart can be used to check the accuracy and completeness of the models. Each function should be triggered by at least one event and every event should trigger at least one function.

A series of data flow diagrams, each showing the sequence of events, are developed from this hierarchical model. The hierarchical model and data flow diagrams

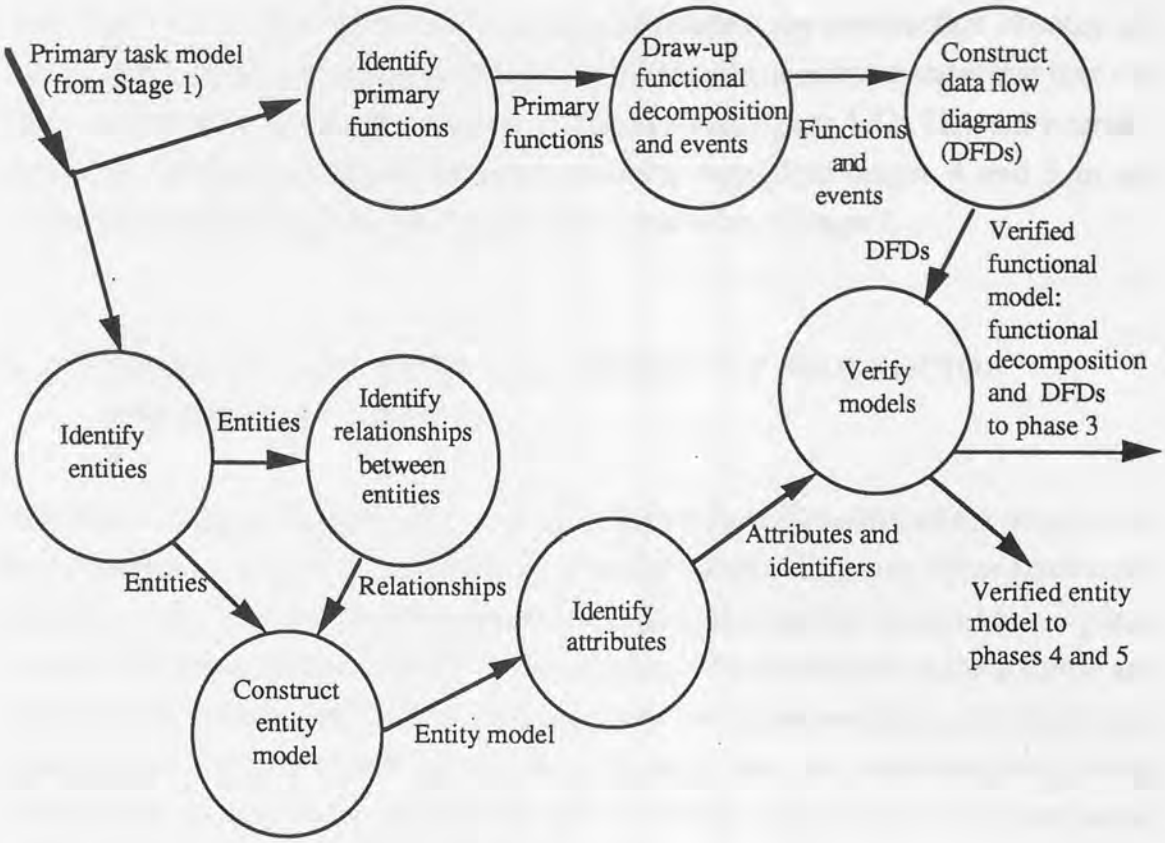


Fig. 3.6. Summary of Multiview stage 2

are the major inputs into stage 3 of the methodology, the next stage, which is the analysis and design of the socio-technical system.

(b) *Development of an entity model*

In developing an entity model, the problem solver extracts and names entities from the area of concern. An entity is anything that you want to keep records about. For example, depending on the problem situation, entities could include customers, sales, patients, hospital beds, and hotel reservations. Relationships between entities are also established, such as 'patients occupy beds' and 'doctors treat patients'. The degree of a relationship is established by finding out the number of occurrences of each of the associated entity types. The attributes of the entity are then established. These are their properties. The processes of forming entities and relationships can be very subjective. A good model will require a good understanding of the problem situation in order to decide which entities and relationships are important. The preceding stage in the methodology - analysis of the human activity systems - should have already given this necessary understanding and have laid a good foundation for this second stage.

An entity model can then be constructed. An entity model is a chart with entities shown in boxes and relationships indicated by lines and arrows between the entities

(see Figure 4.21). The final technique used at this stage is the construction of entity life cycles which show the progress of entities, identifying the legitimate states that they can be in over time. An example is shown in Chapter 4 as Figure 4.22. The entity model, following further refinement, becomes a useful input into stages 4 and 5 of the Multiview methodology. Figure 3.6 provides a summary of stage 2.

3.4 STAGE 3 - ANALYSIS AND DESIGN OF THE SOCIO-TECHNICAL ASPECTS

This stage is largely based on the work of Professor Enid Mumford of the Manchester Business School and Professor Frank Land of the London Business School discussed in Section 2.5. The philosophy behind this stage is that people have a basic right to control their own destinies and that if they are allowed to participate in the analysis and design of the systems that they will be using, then the implementation, acceptance and operation of the system will be enhanced. It takes the view therefore that human considerations, such as job satisfaction, task definition, morale and so on, are just as important as technical considerations. The task for the problem solver is to produce a 'good fit' design (using the framework described in section 2.5), taking into account people and their needs and the working environment on the one hand, and the organisational structure, computer systems and the necessary work tasks on the other.

What sort of options are we discussing in this stage of Multiview? To help us answer this, here is a further list of questions that ought to be asked:

- How much of the work is to be done by the computer and how much by the staff?
- Is the system to be used by trained experts, users with some training, or anybody who needs access to the information?
- How urgently will different pieces of information be needed from the system?
- What, if any, alterations will be necessary to the physical workplace? For example, will there be special computer workstations, acoustic panelling around the printers or improved lighting?
- What changes will there be in the number of jobs, staff gradings, or the working day of individual staff?
- Who will have what responsibilities for which aspects of the computer operation?

The whole business of fitting machines into people's working lives has been given the title 'socio-technical design'. In any large technologically dependent organisation it is obviously a matter of great importance. It can also be a minefield of industrial relations' problems and there is great scope for getting it wrong and setting up mechanisms which are very inefficient to operate. The problems for users in small

organisations may be easier to solve, but they are still there.

Within the socio-technical stage of the Multiview approach, participation is used for these reasons:

- Ethical, because people have the moral right to a major input into the design process of their working situation. This allows users to protect their interests.
- Pragmatic, because detailed knowledge of the working system is possessed by the people who work within the system. Participation is a good way of acquiring this knowledge.
- Psychological, because people do not mind change if they know the reason.

Participation will occur when the users are concerned with decisions about fitting the information system into their working lives. Further, there is a recognition of the many different interest groups or stakeholders and the approach seeks to 'discover their social, technical and organisational objectives which are perceived by the interest groups' (Land, 1982).

An outline of this stage is shown in Figure 3.7. The central concern at this stage is the identification of alternatives: alternative social arrangements to meet social objectives and alternative technical arrangements to meet technical objectives. All the social and technical alternatives are brought together to produce socio-technical alternatives. These are ranked, firstly in terms of their fulfilment of the above objectives, and secondly in terms of costs, resources and constraints - again both social and technical - associated with each objective. In this way, the 'best' socio-technical solution can be selected and the corresponding computer tasks, role-sets and people tasks can be defined.

The emphasis of this stage is therefore not on development, but on a statement of alternatives and choice between the alternatives, according to important social and technical considerations.

It is also clear that, in order to be successful in defining alternatives, the groundwork in the earlier stages of the methodology is necessary and, in order to develop and implement the chosen system, we must continue to the subsequent stages.

In order to be successful in defining and ranking alternatives, the analysts and the users must predict the future environment. A tool applicable to this situation is future analysis (Land, 1982). The tool is based on the idea of forecasting the expected life-span of the information system. The analyst and the users together predict the planning horizon as the period of time during which the designed system will meet the organisation's needs and expectations. The idea here is a trade-off between flexibility and design costs. This technique can be described as consisting of four steps:

1. Predict the kinds of changes which are possible. Are they technological, legal, or economic changes for the organisation?
2. Predict the likely outcome of the information system in the future.

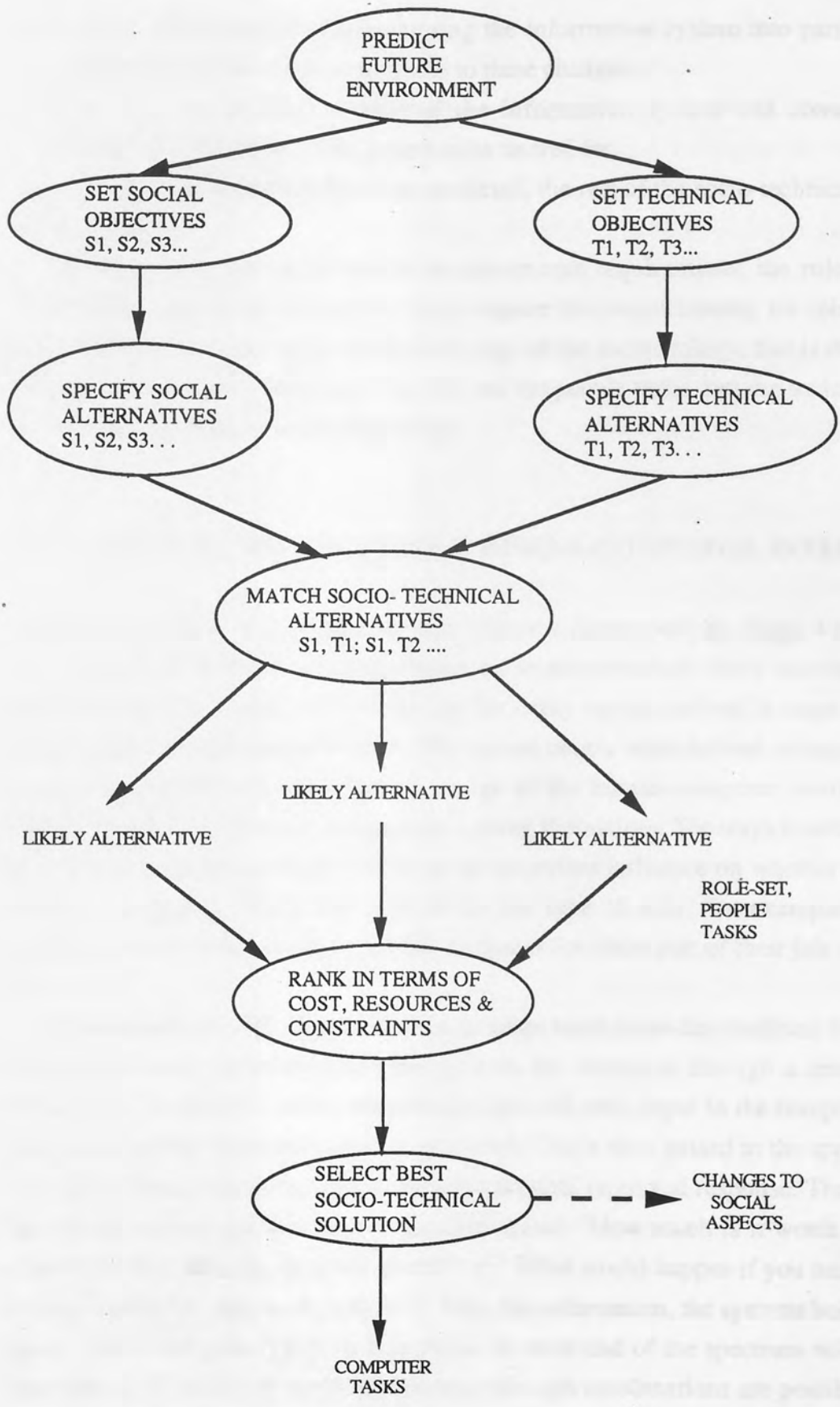


Fig. 3.7. Outline of socio-technical analysis and design (adapted from Mumford 1983a, Wood-Harper et al, 1985 and Wood-Harper, 1989)

3. Evaluate which features (decomposing the information system into parts) of the proposed system are more susceptible to these changes.
4. Determine the planned horizon of the information system and consequently ascertain the flexibility of the system to be catered for.

Once the future environment has been predicted, the rest of the socio-technical design can be carried out.

The outputs of this stage are the computer task requirements, the role-set, the people tasks, and the social aspects. The computer task requirements, the role-set and the people tasks become inputs to the next stage of the methodology, that is the design of the human-computer interface. The role-set, the people tasks, and the social aspects are also major outputs of the methodology.

3.5 STAGE 4 - DESIGN OF THE HUMAN-COMPUTER INTERFACE

Up to now, we have been concerned with what the system will do. Stage 4 relates to how, in general terms, we might achieve an implementation which matches these requirements. The inputs to this stage are the entity model derived in stage 2 of the methodology, and the computer tasks, role-set and people tasks derived in stage 3. This stage is concerned with the technical design of the human-computer interface and making specific decisions on the technical system alternatives. The ways in which users will interact with the computer will have an important influence on whether the user accepts the system. Much will depend on the type of user, for example trained operators, people who routinely use the computer for some part of their job or casual users.

A broad decision will relate to whether to adopt batch or on-line facilities. In on-line systems, the user communicates directly with the computer through a terminal or workstation. In a batch system, transactions are collected, input to the computer, and processed together when the output is produced. This is then passed to the appropriate user. Considerable time may elapse between original input and response. The analyst can ask the user to put a price on increased speed: "How much is it worth to get a response within seconds, minutes, hours?" or " What would happen if you had to wait until the end of the day, week, month?". With this information, the systems builder can lay out the alternatives and their feasibility. At each end of the spectrum will be the basic choice of on-line or batch processing, although combinations are possible, such as on-line validation of data but regular processing of validated transactions in batch.

Another broad decision concerns the type of interface, for example, an interface based on the desk-top metaphor of windows, icons, menus and pointer (WIMP) or a

more traditional command driven system. Then there are decisions relating to specific designs for the menus, commands, soft copy forms, and hard copy forms (with or without character recognition, mark sensing or magnetic stripe reading), that particular types of user will have with the computer system. Some examples of menus are presented in the first case study (Figures 4.23 to 4.26).

There are also decisions relating to the necessary inputs and outputs and related issues, such as error prevention and minimising the number of key strokes. Information may be presented in narrative, as a series of statistics or using graphs of various kinds. There are different ways to display the information and to generate user responses. The analyst must also consider the concepts and vocabulary that the user will be comfortable with. These decisions are taken according to the information gained during stages 1 and 2 of Multiview.

One of the major concerns, after the dialogue has been specified, is that every possible precaution should be taken to prevent errors occurring. Error trapping has long been thought part of program design, and many of the techniques for preventing errors and detecting them relate to program design. Nevertheless, error prevention should be regarded as the responsibility of the systems analyst and systems designer. It is important to make sure that dialogues are clear and unambiguous. If the user knows exactly what is happening and what input is required at each point in the dialogue, then mistakes are less likely to happen. Using meaningful prompts is one way of achieving this. A prompt of '?' only, which is the standard on many systems, does not give any help to the user at all, except that the system is waiting for something.

Another way of preventing confusion, and therefore error, is to ensure that the design follows the natural logic of the situation. An obvious example of this would be to use soft copy forms for data entry where the data was originally on a paper form. Unless there are good reasons for an alternative, the VDU form and the paper form should have the same design. The operator is simply copying the data from the paper to the screen and is much less likely to put data in the wrong place. The user will also feel more 'at home' with the new system.

The system must also allow for all possible error correction. There are many well-known techniques for ensuring the accuracy of data input, such as range checks, hash totals, check digits, cross checks and checks with information held in files. None of these systems is likely to be fool-proof, though they may reduce the amount of human checking necessary. Once errors have been detected, they have to be corrected. The sooner the error has been detected, the easier it is to correct.

There are various things that humans can be 'saying' to computers, for instance:

- They may be issuing an instruction, for example, to run a program, print a report or compute a value.

- They may be putting data into the computer.
- They may be requesting some information from the computer.

Even though the computer will only react in the way that it was programmed, it is important for users that it react in a way that they can relate to. If humans have certain expectations about the way that statements should be interpreted, they will have to spend emotional and intellectual effort getting used to the computer if it reacts differently.

Psychological and linguistic factors must be taken into account when designing the computer part of human-computer dialogues (that is the computer message to the user). For example, if the user makes an error when storing data, the response by the system should not normally be an error number without further explanation. It is not the response that people would expect when dealing with other people, yet it is still a common form of error message.

Once human-computer interfaces have been defined, the technical requirements to fulfil these can also be designed. These technical requirements become the output of this stage and the input to stage 5, the design of technical subsystems. The human-computer interface definition becomes a major output of the methodology.

3.6 STAGE 5 - DESIGN OF THE TECHNICAL ASPECTS

This phase of Multiview owes much to Waters (1979a and 1979b). The inputs to this stage are the entity model from stage 2 and the technical requirements from stage 4. The former describes the entities and relationships for the whole area of concern, whereas the latter describes the specific technical requirements of the system to be designed.

After working through the first stages of Multiview, the technical requirements have been formulated with both social and technical objectives in mind and also after consideration of an appropriate human-computer interface. Therefore, necessary human considerations are already both integrated and interfaced with the forthcoming technical subsystems.

Up to now we have been concerned with WHAT the system is to do. This includes activities, functions, tasks and dialogues. We are now in a position to consider HOW we might achieve an implementation that matches these requirements. Having completed the logical specification, that is, the statement of the user's requirements, we now proceed to the technical specification.

The HOW questions cover things like computers, programs, databases and procedures. The designer is not being asked to create any of these things, as that is the job of the hardware and software engineer. The designer has to specify the particular

hardware and software facilities required to meet the users' needs. In some instances however, particularly in a microcomputer environment or where special facilities known as fourth generation languages are used, the designer may also be the system builder.

A computer specification has to be supplied. The designer must be rigourous so that, for example, the computer expert does not make decisions which affect the way in which information is presented to the users. The design can be created in many ways, for example, using an application package on a microcomputer, developing a prototype with the use of one of the tools now available to support prototyping, or a tailor-made solution using a conventional programming language and a database system. The difference is largely one of emphasis, as the package solution must be looked at as a possible way forward for at least part of the design, and prototypes are very helpful in creating a detailed design. The advantages of prototypes are that they permit users to visualise and act out those parts of the system that concern them and therefore to make better decisions about what they want. Prototyping is particularly useful in situations where the requirements are not clear-cut. There are many options open to the analyst.

Having made some broad-based decisions relating to hardware and software, Multiview takes a more detailed technical view so that the analyst can concentrate on efficient design and the production of a full systems specification. Many technical criteria are analysed and technical decisions made which will take into account all the previous analysis and design stages. The interaction of design activities are shown as Figure 3.8.

The final major outputs of the methodology are shown in Figure 3.9. These are the application, and the information retrieval, database maintenance, control, recovery, and monitoring aspects, and the database. The inputs and outputs necessary to support these are also defined.

The application is concerned with the technical solution formed from the user requirements. The output from this phase will be the various messages to the users (such as soft copy and hard copy reports) and the updated database with the new information gleaned from the application processing. The processes are the routines on which the programs are based. The applications would traditionally be looked at as the 'programs to be written', but today, as we have seen, many applications are handled by software packages. As far as the user is concerned, the application subsystem contains all the things that the system was supposed to do (except for information retrieval, which is treated separately). All the functions from the function hierarchy chart that were to be computerised would be part of the application subsystem.

Information retrieval is different from the application in that the database is unchanged after the enquiry has taken place. Information retrieval is the primary

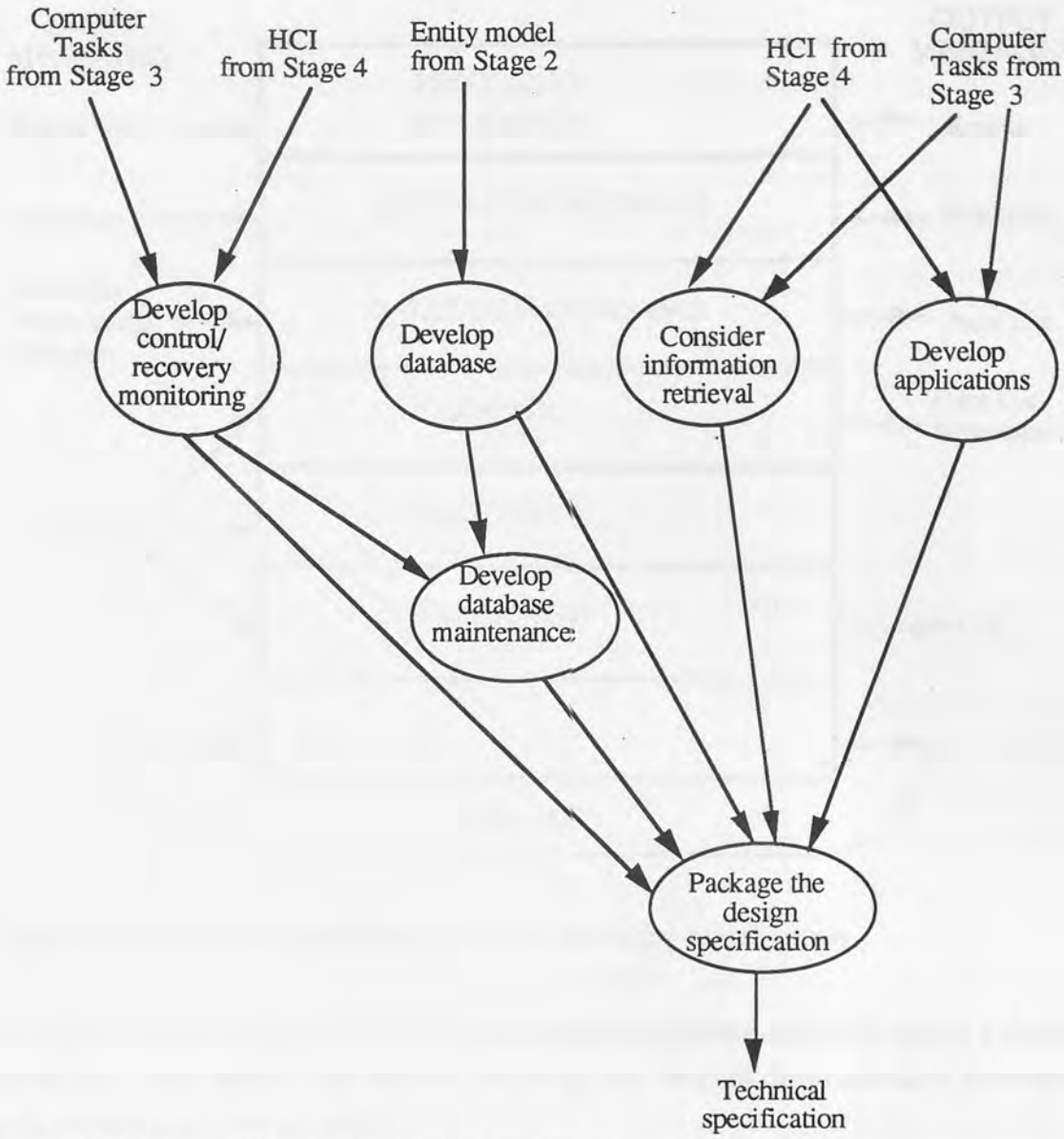


Fig. 3.8. The overall process of technical design

purpose of many computer systems, for example, those available in libraries for borrower enquiries and on-line databases, such as Prestel in the UK, The Source in the US and Minitel in France, which are available for public enquiries. There are, in general, two ways of performing information retrieval. The first of these are the predefinable reports which are generated as a matter of routine within the information system. Many systems have a report generator which permits users to specify the reports that they require themselves without too much technical coding. Most systems now also give the user opportunities to make enquiries directly using a conveniently located computer terminal or a microcomputer with compact disk or hard disk containing the information. These enquiries may be standard ones within the organisation for which a special routine has already been written. Alternatively, they could be one-off enquiries set up using the query language or natural language interface

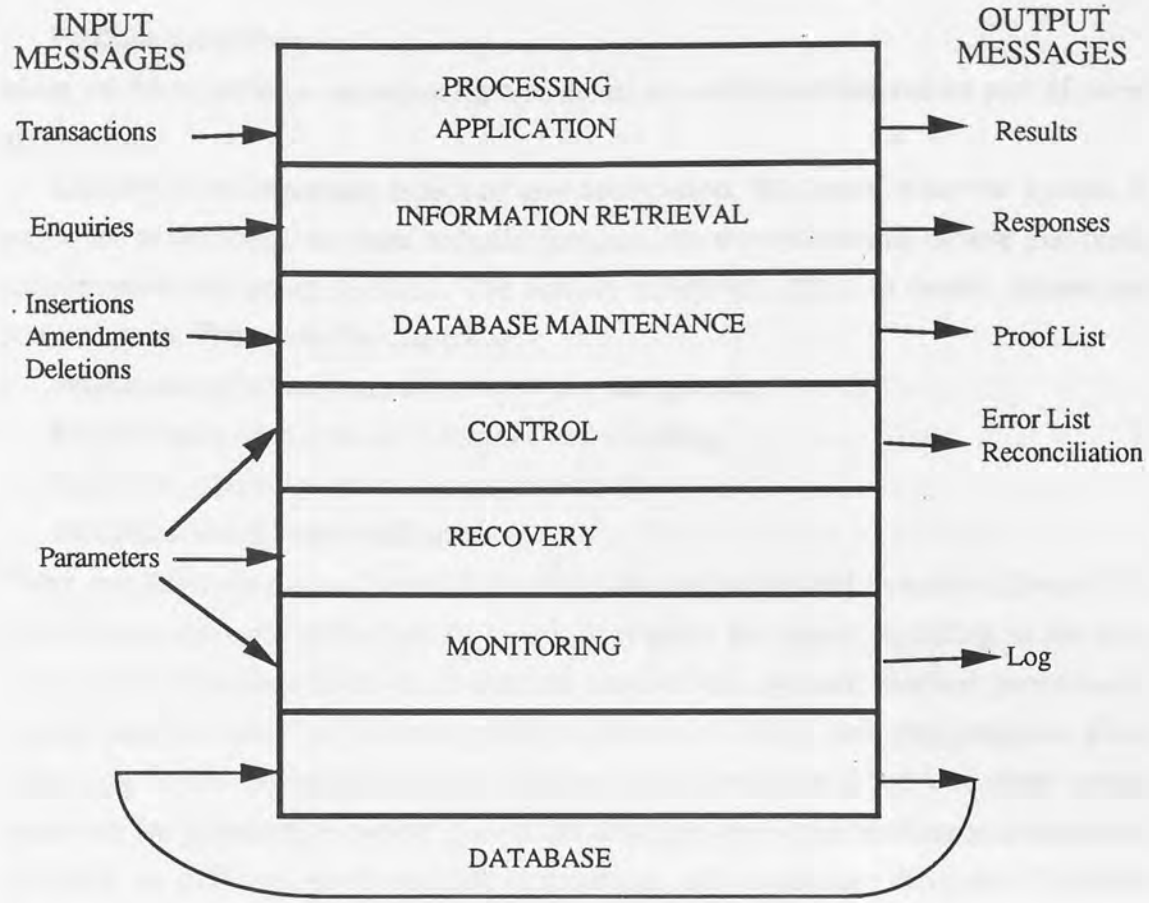


Fig. 3.9. Outline of the requirements for the technical specification

provided. A query language permits the user to formulate enquiries using a formal vocabulary and syntax, but without requiring the level of detail normally associated with coding computer programs.

An information system is built around a database of some sort. This may use a sophisticated database management system or be a conventional collection of files. In this part of the design phase, we are concerned with setting up the skeleton or plan of the database rather than with getting any data into it. The database has to contain all the data required to support the applications and information retrieval. The information needed to design the database comes from the entity model, along with the attributes that have been specified for each entity.

Database maintenance refers to all activities that have to take place in order to keep the data up-to-date and correct. This is concerned with transactions: amendments, deletions and insertions of the applications structure of the data. The responses will give proof that the data has been stored. Procedures need to be provided to:

- Insert new records.
- Amend existing records.
- Delete old records.

- Produce proof lists.

Many of the insertions, amendments and deletions will be carried out as part of some application.

Control is an important aspect of any application. We know what the system is supposed to be doing, we need to build features into the system that ensure that these requirements are being fulfilled. The control subsystem helps to detect, locate and correct errors. These may be caused by:

- People putting erroneous information into the system.
- Programmers who have left mistakes in the coding.
- Operators who have run programs incorrectly.
- Machines which have malfunctions.

There are different sorts of control features that can be applied to each of these. The most useful sorts of control are those which prevent the errors occurring in the first place. The controls will be built into all parts of the system: clerical procedures, application programs, information retrieval, database design, and maintenance. They form part of the completed system. Further, good design will prevent many errors occurring (an appropriate coding system, for example) and good training and education schemes, as well as a good working atmosphere, will encourage diligence. Controls may be established for data entry, to detect program errors, operator errors, machine malfunctions and errors in the database.

Recovery relates to what will happen when the system breaks down. Specification takes into account preparing for the failures and recovering from failures when they have been corrected. The recovery subsystem is related to the control subsystem as it tells users what to do when a mistake has been discovered. If a user makes a detectable mistake during data entry then the control system should set the recovery system to correct the mistake. Recovery from database errors can also be built into the control procedures.

Finally, monitoring relates to the necessity, to know what the system is doing, who is using it, how much time it is operating, and so on. It is very easy for the small system user to see for himself what is happening if he takes the trouble to observe. However, for a large machine, or even for a number of small machines, it is necessary to create a monitoring system.

Even after the five stages of Multiview are complete and the systems builder presents the system for acceptance testing, the analyst and user have to ask the question: "Does the system meet the requirements?" More accurately, they have to ask the following questions, because as we have shown, there are several different sorts of requirements:

- Does the system actually work, or are there 'bugs' in the programs or

incompatibilities in the hardware?

- Does it handle the whole of the information model? Are all the entities and functions in there?
- Does it achieve the stated socio-technical objectives?
- Are the people using the system happy with their interaction with the computer (human-computer interaction)?
- Most importantly, is it supporting the human activity system?

In other words, all the five aspects of Multiview have to be tested thoroughly.

We have seen in this chapter how Multiview has been constructed as a blend of the themes in information systems development methodologies discussed in Chapter 2. For example, it has used aspects of systems approaches and to a lesser extent planning approaches in stage 1; structured approaches and data analysis approaches in stage 2; and participative approaches in stages 3 and 4. But it has done so in a coherent manner which forms a framework, though a flexible framework, to be used in the various problem situations by various analysts. Its use is described in the three cases.

In Chapter 4, the first case is presented of Multiview in action. It shows how Multiview was used and its strengths and weaknesses exposed in a real situation.

Chapter 4
The Distance Learning Unit Case

The first case looks at the Distance Learning Unit (DLU) at South Bank Polytechnic in the UK. This case is fully described in Wood-Harper et al (1985) and it was used to illustrate Multiview in action. It was the first full practical use of Multiview. Examples of the various deliverables are shown as diagrams in this chapter.

Parties which were involved in this application included the Polytechnic itself, the Manpower Services Commission (MSC) and the Paintmakers Association, a professional association for paintmakers. The DLU was attempting to provide educational services to train members of the Paintmakers Association who were scattered in small groups throughout a wide area. The MSC was responsible for monitoring the process. The rich picture, shown as Figure 4.1, provides a summary of the situation.

The rich picture places in perspective the elements of the DLU and its environment - the Polytechnic (in particular the administration), the Manpower Services Commission and the Paintmakers' Association, including the departmental boundaries of the Polytechnic.

The problem owner in the case study is shown in the rich picture as the head of the DLU running an academic support unit of the Polytechnic. He wished to set up the unit to provide a service to the paint industry in the UK. The problem owner and the DLU are central to the case and drawn as such in the rich picture. The problem owner was expected to become the system user (though in fact an administrator was appointed by the Paintmakers Association later).

The factory shape at the top of the page represents the Paintmakers Association of Great Britain. Their members are all paintmakers who are based at factories distributed throughout the UK. This was seen as an important element of the structure of the problem situation, hence the outline of England, Scotland and Wales included in the rich picture. Distance learning is viable because the small number of trainee paint technicians at each site make local college courses uneconomic.

This left the problem owner with three problems:

- Getting funds to establish all the necessary training materials and other requirements of a distance learning situation.
- Maintaining the right academic levels.
- Keeping track of the work of several hundred students across the country.

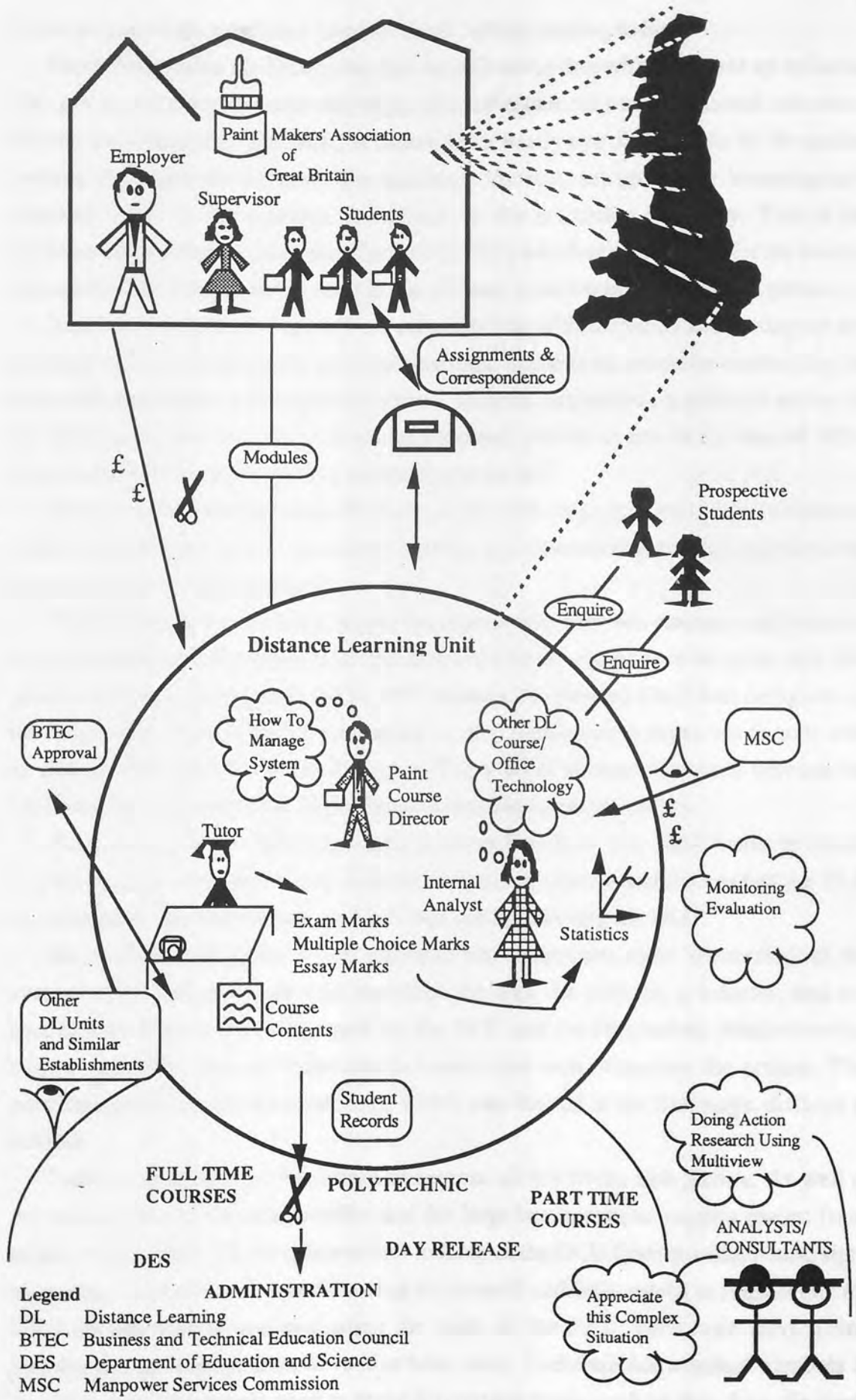


Fig. 4.1. The rich picture of the problem situation

These problems are simplified into the 'think' bubble relating to him.

Funds came from the Manpower Services Commission which was set up to funnel UK government money into the expansion of technical and vocational education beyond the classroom. The MSC is shown by a 'beady eye' in the right of the picture looking for 'value for money'. The academic content, set up by the lecturing staff (marked 'tutor' in the picture), is ratified by the education authority. This is the Business and Technical Education Council (BTEC) which sets standards for the content and teaching of courses at this level in the UK and is shown in the left of the picture.

Another role of lecturing staff concerns dealing with enquiries and setting up and marking written assignments and examinations. Some tests would be marked by the industrial supervisor. Although the internal systems analyst was a problem solver for the system, she was also about to set up a second, similar course in the area of office technology, and therefore she is a potential user as well.

Having looked for elements of structure, the next stage concerns process elements which include tutoring and assessing students, and transmitting student records to the administration of the Polytechnic.

The climate of the situation, that is the relationship between structure and process, highlighted problems coming from the mismatch of the established structure and new processes formed in response to the new situation. As the new DLU was being set up, the Polytechnic administration was trying to stretch its existing structures to cope with an activity for which it was not designed. The symbol of crossed swords between the DLU and the Administration highlights this possible cause of conflict.

There are other soft facts expressed in the rich picture. The MSC wants to ensure that it is getting value for money in this new training venture and the head of the DLU wants to know the criteria that the MSC will use in assessing the DLU.

An additional problem which emerged was agreement upon 'ownership' of the system. Although the issue was identified through the process, a solution was not agreed. Decisions were being made by the DLU and the Polytechnic administration, and yet the MSC and the Paintmakers Association were financing the project. This political problem made the consensus, which was desired in the first stage, difficult to achieve.

Various symbols have been used to express all this in the rich picture. As well as the crossed swords showing conflict and the large beady eyes to suggest groups (such as the MSC and the DLU's competitors) looking at the DLU from outside, pound signs suggesting money (in this case financing the project) and little graphs to suggest that the MSC are monitoring and evaluating the work of the DLU, have been used. Think bubbles and people symbols have also been used. These are not standard symbols in the same way as symbols used in other diagraming tools, such as data flow diagrams

and entity models. Rich pictures are essentially discussion documents for use by people in the problem situation. Whatever symbols are most appropriate for communicating the ideas, facts, problems, and so on in the problem situation to the people in that problem situation, should be used. However, the symbols here have been used by the author over many projects, and they seem to be understandable and helpful to most people.

From the rich picture primary tasks can be identified. It was felt that the task central to the problem situation was that 'the DLU has to educate students at a distance', and everything else is carried out to achieve this end. There were many issues that the rich picture has highlighted, the major issue being that of identifying the problem owner related to the complex financial support coming from the MSC, the Polytechnic, and the Paintmakers Association (whose members were to be educated by the DLU).

Having thought more about the system for educating students at a distance, it will help to detail the information handling that is required. This will, in turn, help to solve the problem in hand, that is, to create an appropriate information system. It will help in other ways. For example, the paintmakers are interested in solving an industry problem of which educating their trainees is a part. The MSC are interested in the way in which training is administered and delivered. However neither of these 'definitions' is relevant for this purpose. Any light they shed is from the 'side'. The relevant system must be relevant to the process of improving the problem situation, and this situation is one of handling the information processing aspects of the distance learning course.

Throughout the process, the following major concerns were identified:

1. Efficient administration and management within the constraints of the equipment already purchased.
2. Communication with and motivation of physically isolated students.
3. Keeping sufficiently accurate records to satisfy the auditors.
4. Building a good reputation for the DLU.

It was concluded that the issues and primary tasks described above could largely be resolved by the following relevant systems:

- An administration and management system.
- A communication and motivation system.
- An information provision system.

The information provision system was seen as particularly important by the staff of the DLU. The DLU system was new, but it can be compared with a classroom situation. One thing that was most important in the design of the DLU system, over and above the recording of students' progress, was the need for progress reports for the project manager and the MSC on the success of the course itself, just as if it were a product being sold commercially. In a conventional course, the primary concern is to get 25 or so suitably qualified people together at the right time. It does not matter much

if there are 25 who applied or 250, because there is nothing that can be done for the others. Nor is there much that can be done if some of the students prefer some slightly different combination of subjects. With a distance learning course it is possible to be much more flexible in the number of students that can be taught and in the combination of subjects offered. There is therefore a whole new opportunity to do market research relating to who is looking for training and what qualifications they already have. In order to do this, it is necessary to collect a lot more information than is needed to satisfy the strict needs of student registration. This should be reflected in the conceptual model for the DLU.

These three relevant systems are subsystems to support the higher system which is to provide courses to increase technical skills and knowledge in the paintmaking industry by distance learning. On forming the root definition, there was particular difficulty, already mentioned, about deciding on the owner. There was nothing that the analysts could do about this, but at least the actors were aware of the problem. It is sometimes difficult to produce a rigourous root definition because of these political or other problems.

The working root definition formed was:

> *'A system owned by the Manpower Services Commission and operated by the Paintmakers Association in collaboration with the Polytechnic of the South Bank's Distance Learning Unit, to provide courses to increase technical skills and knowledge for suitably qualified and interested parties, that will be of value to the industry, whilst meeting the approval of the Business and Technical Education Council, and in a manner that is both efficient and financially viable.'*

The CATWOE criteria which was used to create, and then check and revise the above root definition was as follows:

Customers	Paintmakers and their employees.
Actors	Polytechnic Distance Learning Unit and administration staff. Staff and students in the paintmaking industry.
Transformation	To provide courses to increase technical skills and knowledge for suitably qualified and interested parties.
Weltanschauung	Both efficient and technically viable (implicit in distance learning).
Owners	MSC (and perhaps the Paintmakers Association)
Environment	The paintmaking industry, BTEC approval (for course validation).

The conceptual model was derived from the root definition. The following aspects and named subsystems were identified:

ASPECT	SUBSYSTEM
Technical skills and knowledge	Course exposition system
Monitor and Control	Report and information provision system, and Monitoring and evaluation system
Administration and management	Administration system, Report and information provision system, and Pre-exposition preparation system

The top-level conceptual model (Figure 4.2) shows these subsystems and the information flows between them and those coming in and out of the system. So as to ensure that the most useful systems have been identified and understood, they are described in more detail in words and diagrammatically through lower level conceptual models (Figures 4.3 to 4.7).

Pre-exposition Preparation System. This relates to the design and organisation of the courses, including organising the tutors to take courses and prepare material, identifying the target student population and designing the course structure. Student motivation needs to be considered when deciding on the sorts of material to provide as well as the technical skills and knowledge to be transmitted. The production of materials and its transmission to the students has to be managed.

Course Exposition System. This is the system which transmits the technical skills and knowledge to the students. It includes their learning and undertaking of examinations and other work for assessment, as well as the provision of materials from which to learn and contact with tutors. It is necessary to consider the communication and motivation problems of the students in a distance learning situation. Communication and feedback needs to be fast in order to maintain the students' motivation.

Administration System. This is the system to administer the sending of work, receiving and marking the work, and keeping records on the students and their progress, and to perform accounts and stock control, as well as answering routine enquiries from students, tutors and employers.

Report and Information Provision System. This system collects information on student marks or particular assessments to produce reports. These will include statistical reports for the various bodies involved.

Monitoring and Evaluation System. This system collects the information from the reports and administration records, and analyses it under various criteria in order to evaluate the viability of the course, its usefulness and so on.

In the first level conceptual model, each of the subsystems is shown in context with the other subsystems (Figure 4.2). The large arrows represent flows of physical things, such as reports, materials, monies, and so on, both within and out of the system. The small arrows represent information flows between the subsystems. Secondly, each subsystem is broken down into its component subsystems with their interactions. This is represented by the level 2 conceptual models (Figures 4.3 to 4.7).

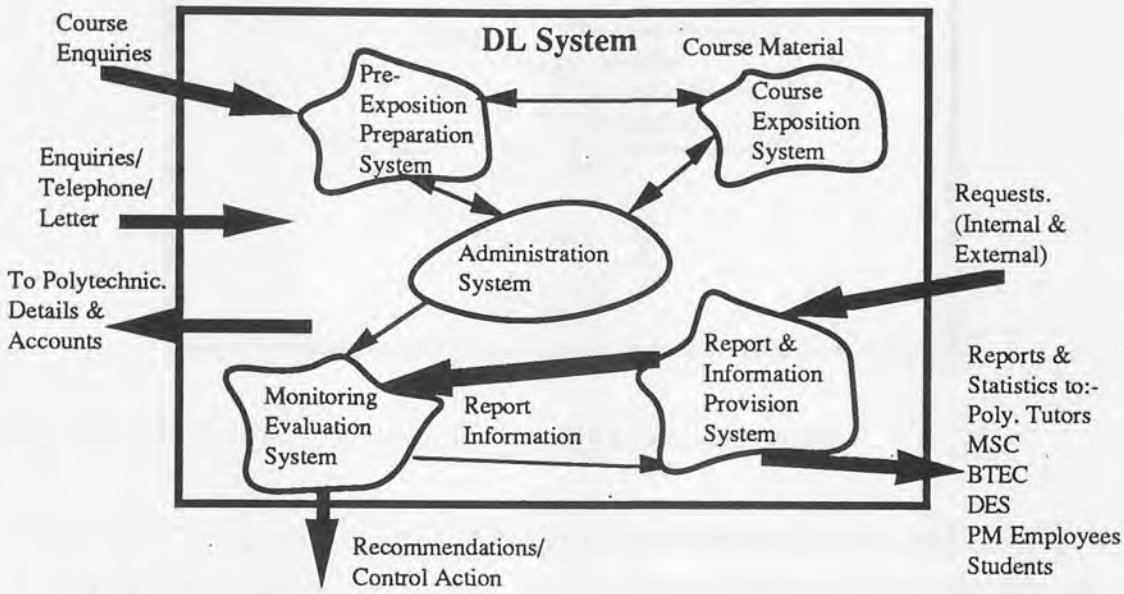


Fig. 4.2. Conceptual model level 1

The next stage is normally to compare the conceptual model with what is actually happening, in order to uncover discrepancies and to recommend changes. In the case of the DLU, this was impossible as there was as yet no real world system in operation. Thus the conceptual model represented the recommendations for the real world organisation. This particular case differs from most other cases in that it was dealing with a situation for which no predecessor existed. In addition, the nature of the MSC constrained the flexibility of technical alternatives that might have been chosen. The MSC required that each project be considered separately, so projects which might share equipment (the future office technology project, for example) could not be considered in terms of their possible total benefits. Sometimes the realities of the situation do not enable the ideal to be followed (in this case elements of the systems approach discussed in Section 2.3).

Because the 'system' did not exist in any form within the Polytechnic, the flexibility of Multiview's model became apparent. No staff existed for the 'system'; they were still to be hired. Nevertheless, a significant amount of participation was available. The internal analyst for the DLU (Lyn Antill, one of the authors of the original book on Multiview (Wood-Harper, Antill & Avison, 1985)) served as a member of the problem

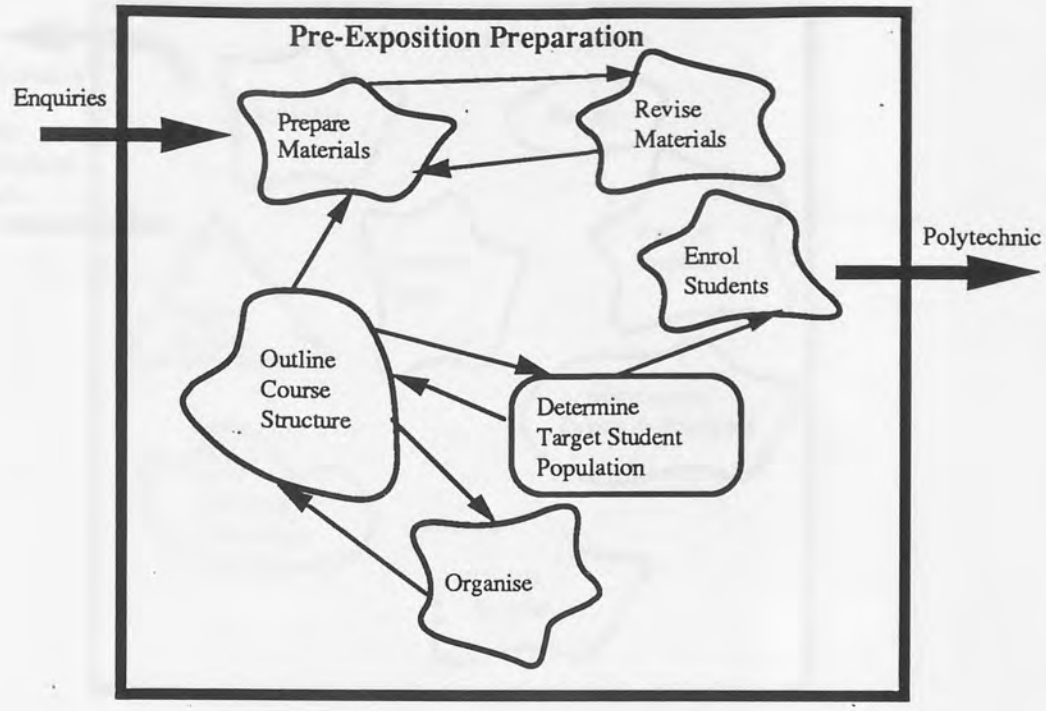


Fig. 4.3. Conceptual model level 2 - pre-exposition preparation

solving team, along with an external analyst (Trevor Wood-Harper) and his team.

The problem solving team appeared to have worked very well and to have been one of the successful dimensions of this case. The external analyst adopted the role of facilitator (Section 2.5). Traditionally, systems analysts have tended to impose their choices on the users.

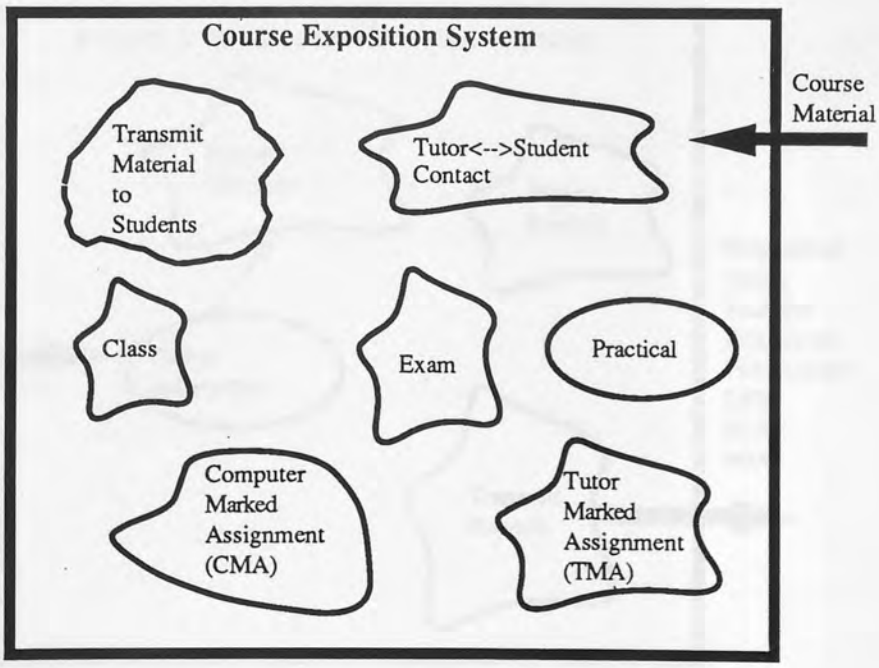


Fig. 4.4. Conceptual model level 2 - course exposition system

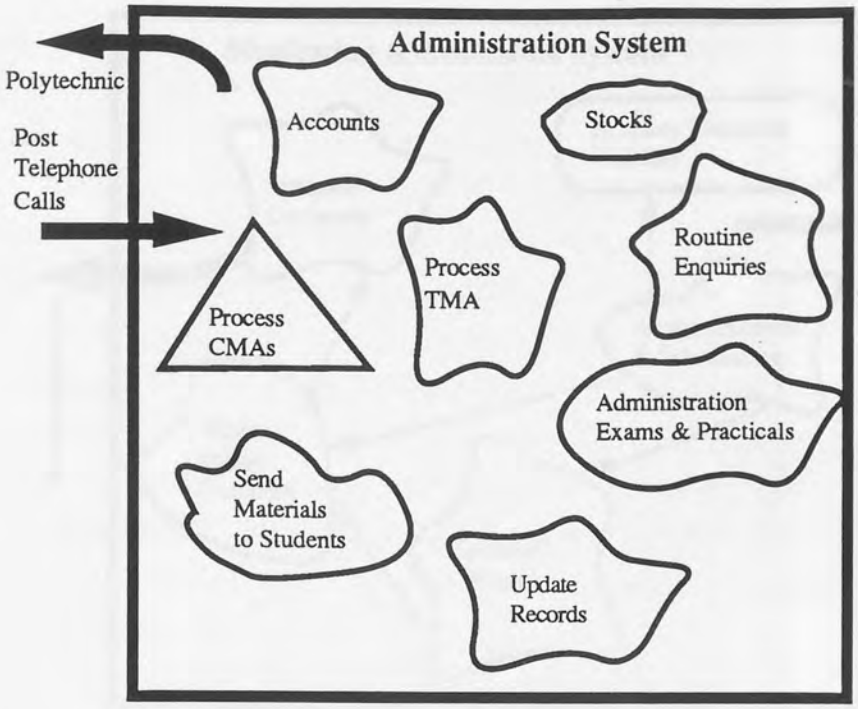


Fig. 4.5. Conceptual model level 2 - administration system

The multiple perspectives identified by the rich picture can be a problem if conflict exists and resolution is not forthcoming. The rich pictures as a representational form, however, worked extremely well. Nevertheless, uninitiated users did not always see the significance of the conceptual models. Sometimes the movement from one conceptual model to one at a different level proved difficult and not clearly directed.

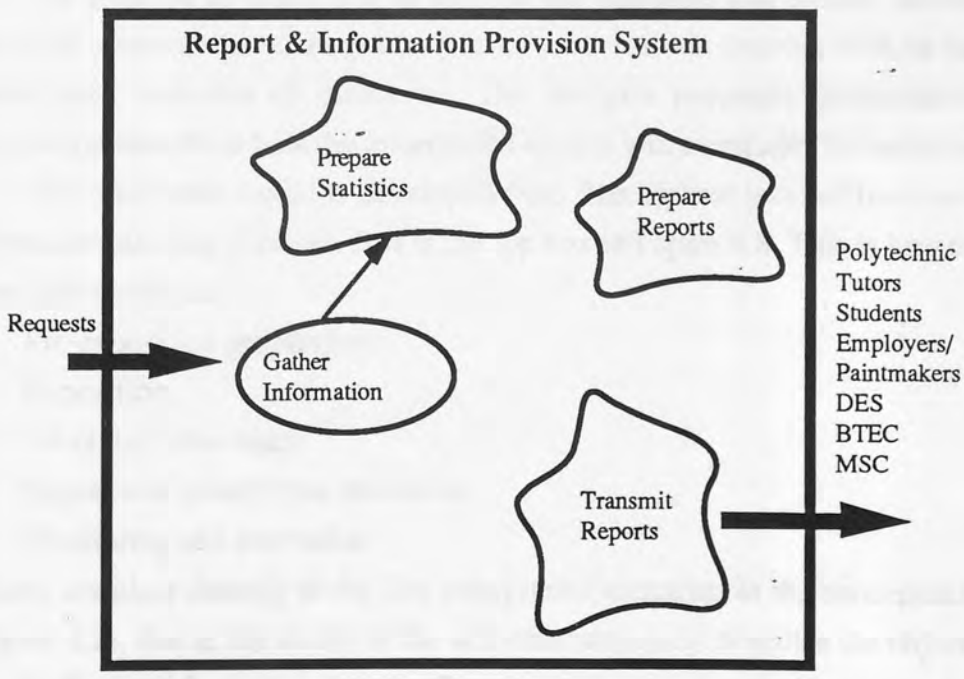


Fig. 4.6. Conceptual model level 2 - report and information provision system

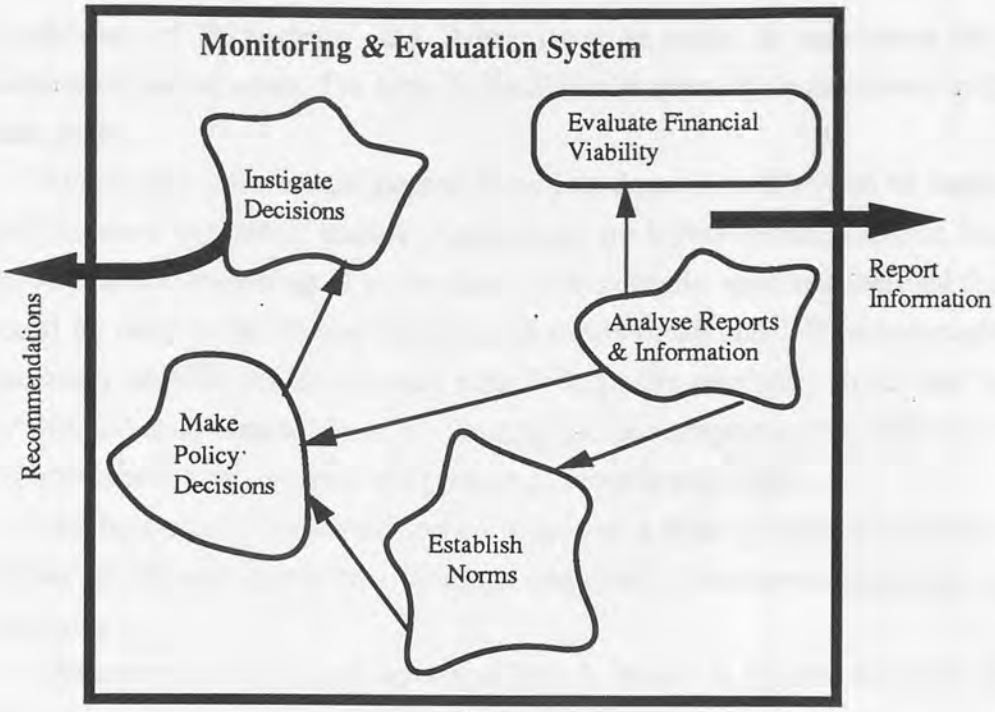


Fig. 4.7. Conceptual model level 2 - monitoring and evaluation system

This case also saw the importance of top management support in the process. They agreed to the use of a novel approach to information systems development and gave their support throughout. Further, the problem solving team were knowledgeable in educational environments and this provided a basis for understanding and communication between the team and top management.

The purpose of Stage 2 is to analyse the functions and entities derived from the notional system, that is, the primary task root definition together with its model and its associated universe of discourse. The analysis proceeds independently of any considerations about how the information system will eventually be implemented.

The functional model is developed first. The highest level of function is 'Operate Distance Learning Course'. This is the top box in Figure 4.8. This is broken down into five sub-functions:

- Pre-exposition preparation.
- Exposition.
- Administration tasks.
- Report and information provision.
- Monitoring and evaluation.

These correlate directly to the five subsystems identified in the conceptual model (see Figure 4.2), that is, the model of the activities necessary to realise the objective system.

In Figure 4.8, we see that the 'Pre-exposition preparation' has been broken down into 'Course preparation' and 'Student preparation'. These are logically distinct. The

breakdown of 'Exposition' and 'Administration tasks' is not drawn on that page because of lack of room. The letter in small circles show the connections to diagrams in later pages.

'Report and information system' is broken down into 'Services to internal bodies' and 'Services to external bodies'. Again these are logically quite distinct, but may turn out in practice to overlap in some places. For example, reports generated for one body could be used to satisfy the information needs of another. To avoid confusion it is necessary to differentiate between what is 'logically necessary to do' and 'the method by which it is convenient to do it'. Sending the same report to two different bodies is a decision about how, whereas the present concern is with what.

The functional decomposition continues to a third level as 'Services to internal bodies' is broken down into 'Student enquiries', 'Employer enquiries' and 'Tutor enquiries'.

The complete functional decomposition is shown in Figures 4.8 to 4.16. The fact that there was no existing system to study meant that it is not possible to have complete confidence in the function charts. There must be things missed out and these will be discovered when the system is designed. However, considerable progress has been made, and this would be unlikely to have been achieved without this modelling technique.

The next stage is to think about what events trigger each of these actions (sometimes referred to as operations), and what information is involved. An event (also called a transaction) is a stimulus that initiates a function. The event 'request by tutor' is arrowed in Figure 4.10). There are two types of events: internal and external. An internal event occurs inside the model and is a function of completing or starting. An external event occurs outside the domain of the model. These events can be depicted on the function hierarchies (through the use of arrows) and on data flow diagrams.

Data flow diagrams give precedence to processes or functions and the use of the data. The analyst constructs these data flow diagrams by examining how information flows in and around the function hierarchies. These diagrams create a good basis for the questions that the analyst must ask the user who is doing the job, and who is the source of the analyst's information. By using this diagram, entries missing from the functional chart can be checked. Formally, this checking can also be done by a Function/Event matrix (described later and shown as Figure 4.20). This chart can be used to check the accuracy and completeness of the models. Each function should be triggered by at least one event and each event should trigger at least one function.

Figures 4.17 to 4.19 are data flow diagrams showing some of the DLU activities. To complete the model we would draw similar diagrams for all the data flows through all the other functions. This is a methodical process which uses up quite a lot of paper.

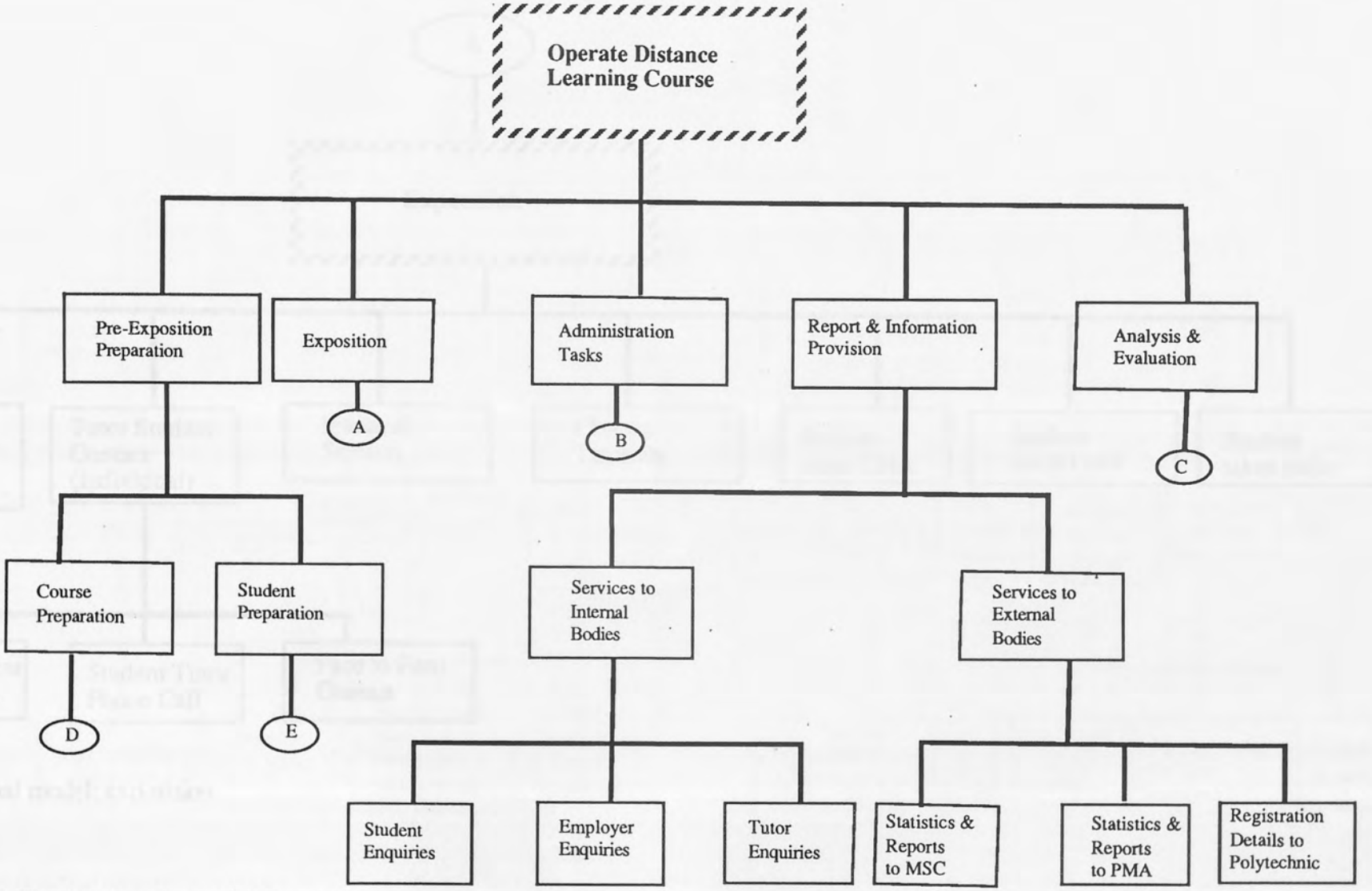


Fig. 4.8. Function model diagram

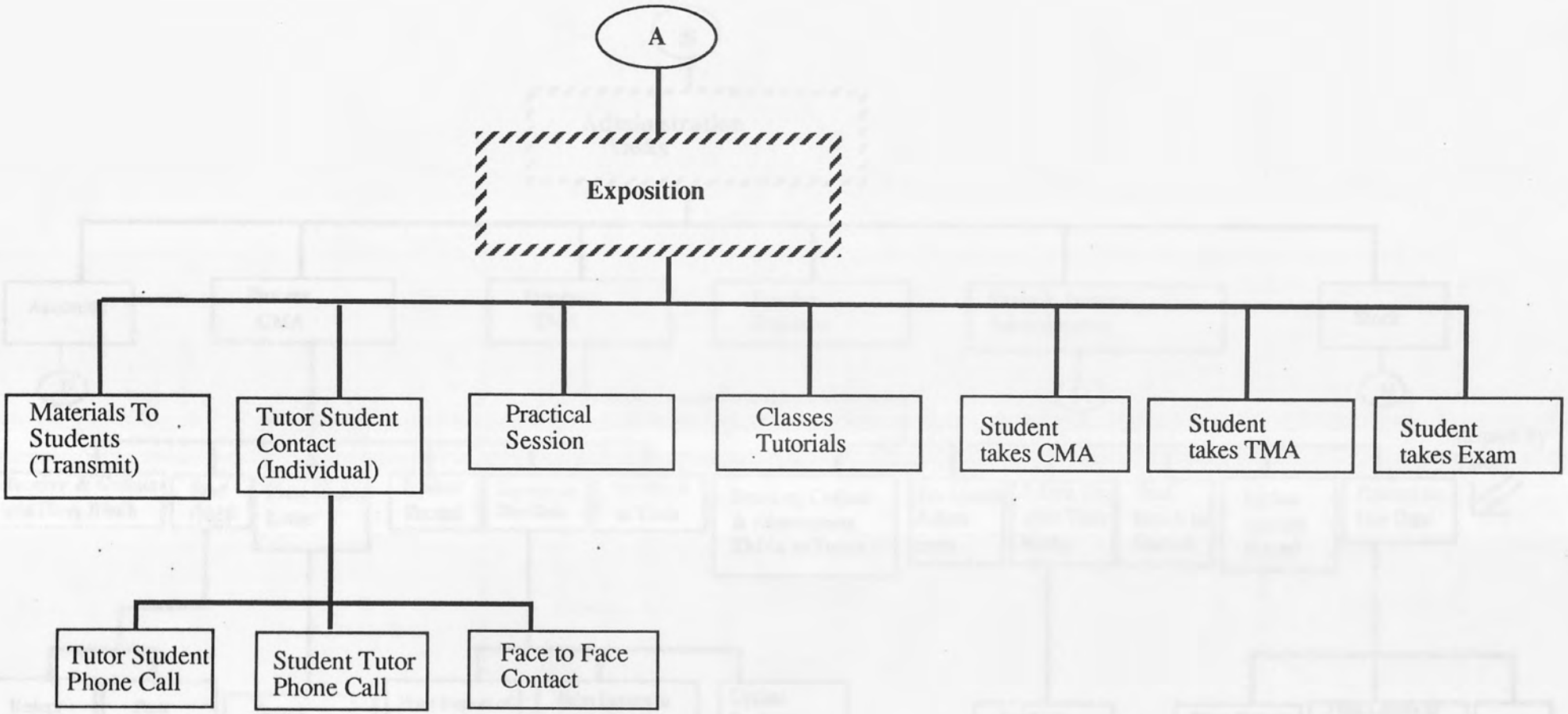


Fig. 4.9. Functional model: exposition

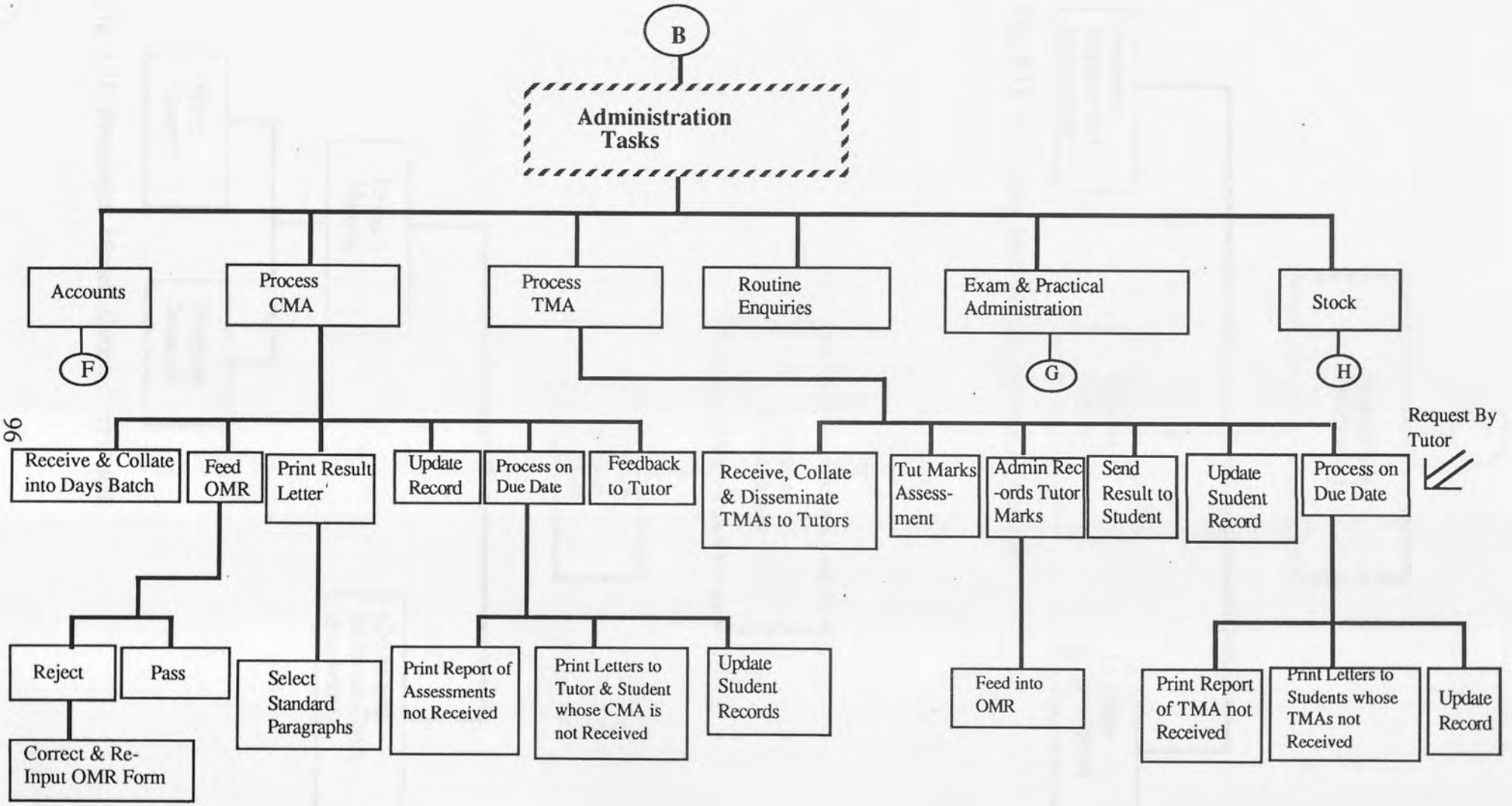


Fig 4.10. Functional model: administration tasks

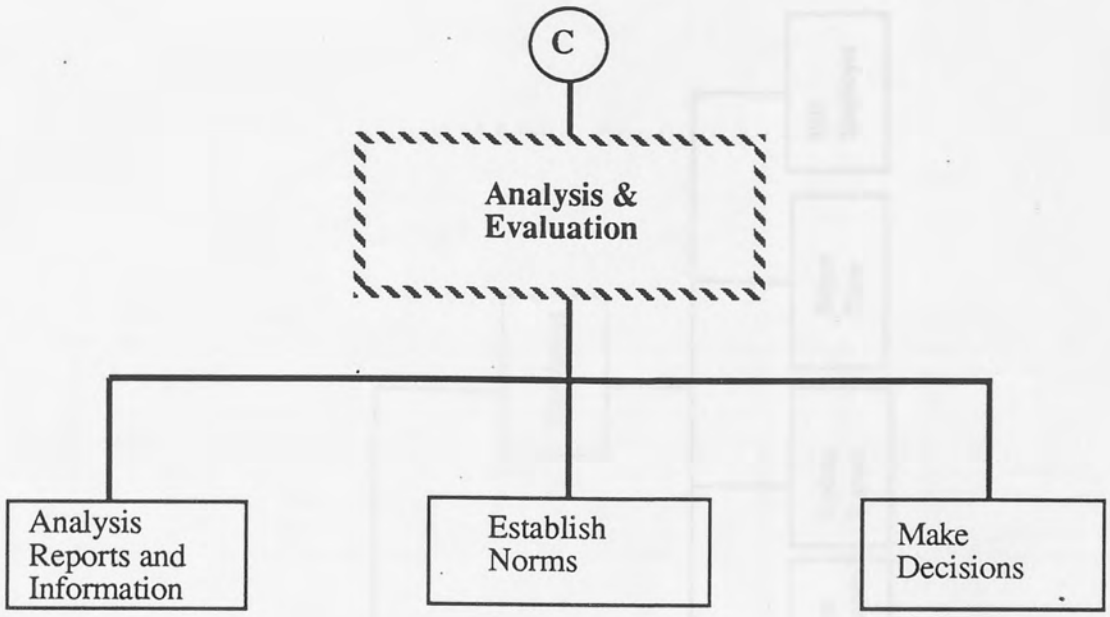


Fig. 4.11. Functional Model: Analysis and Evaluation

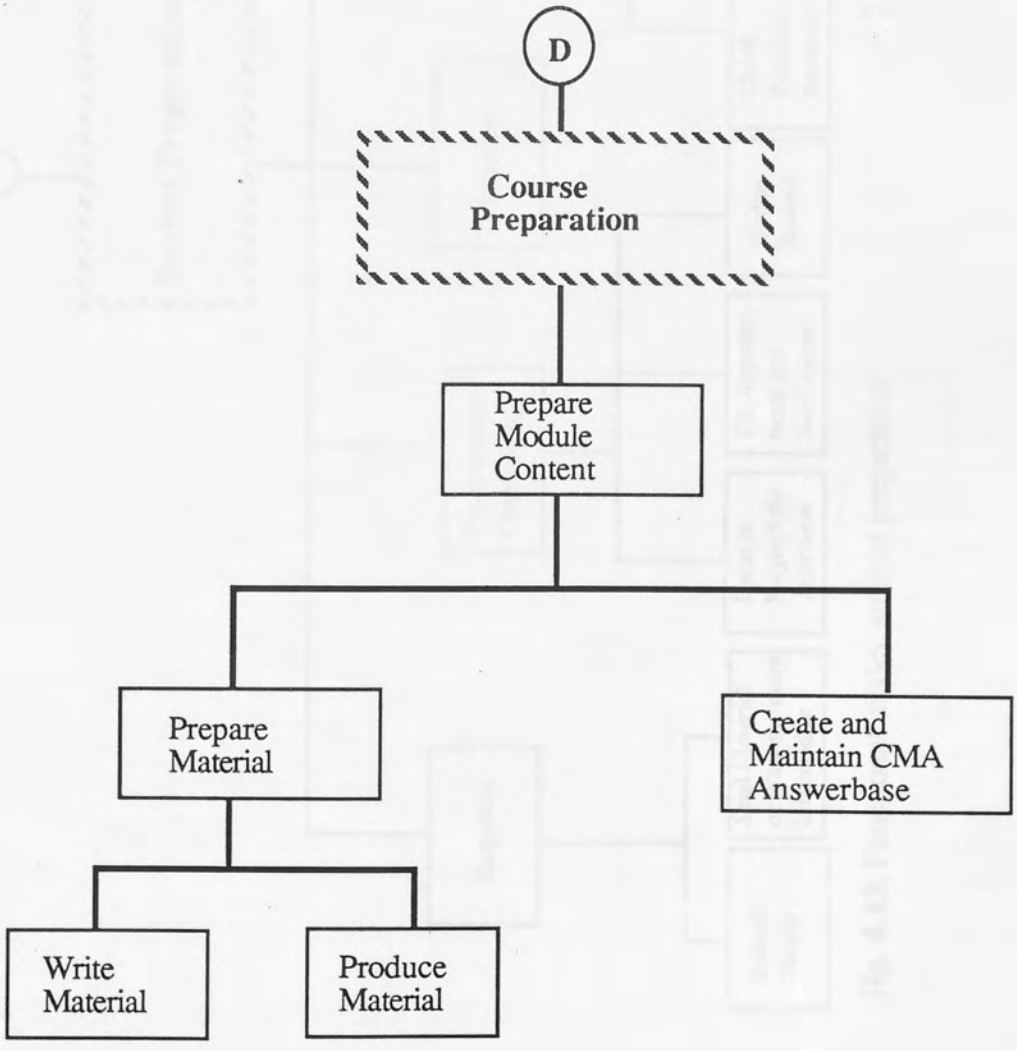


Fig. 4.12. Functional Model: Course Preparation

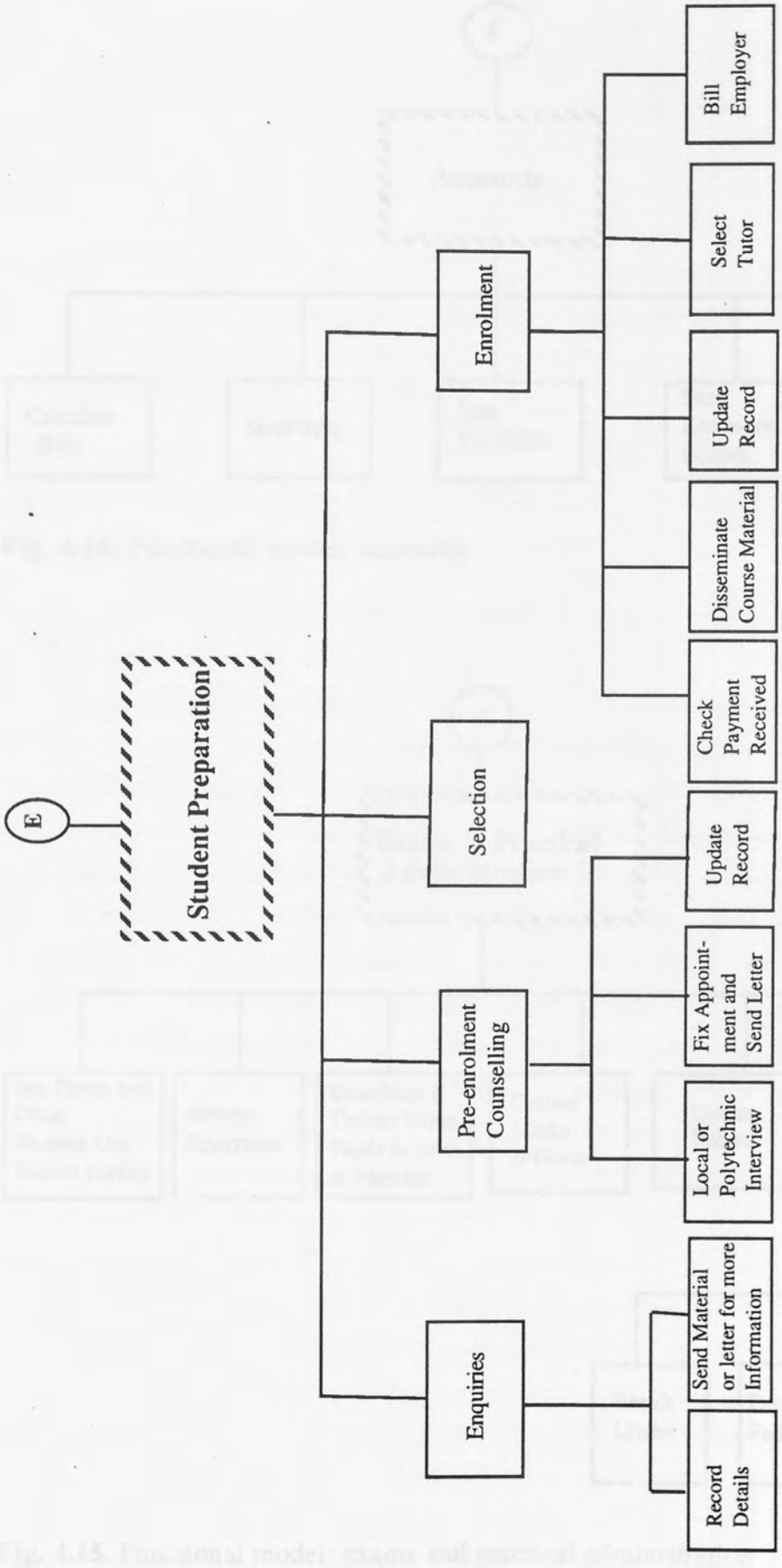


Fig. 4.13. Functional model: student preparation

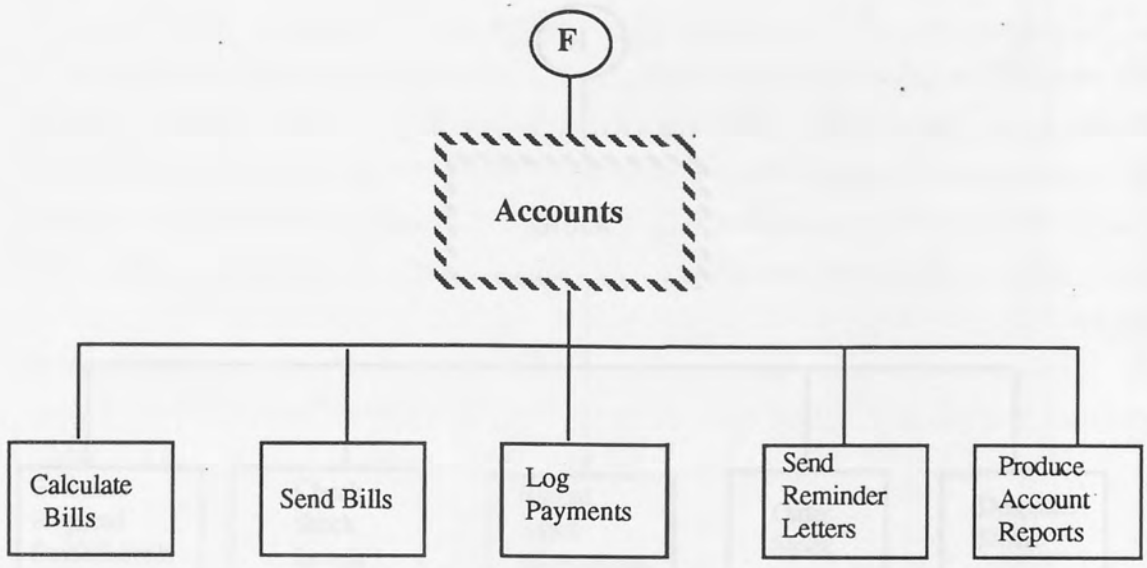


Fig. 4.14. Functional model: accounts

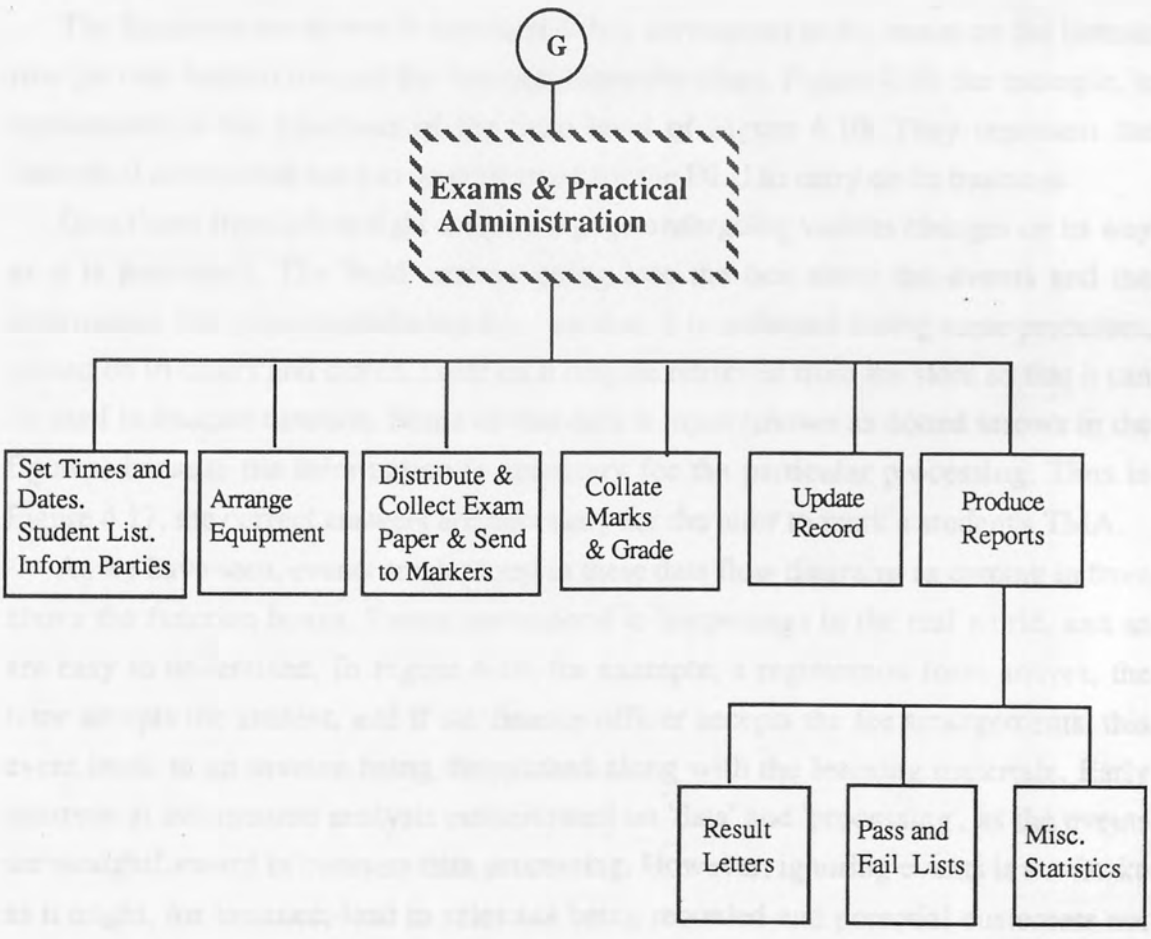


Fig. 4.15. Functional model: exams and practical administration

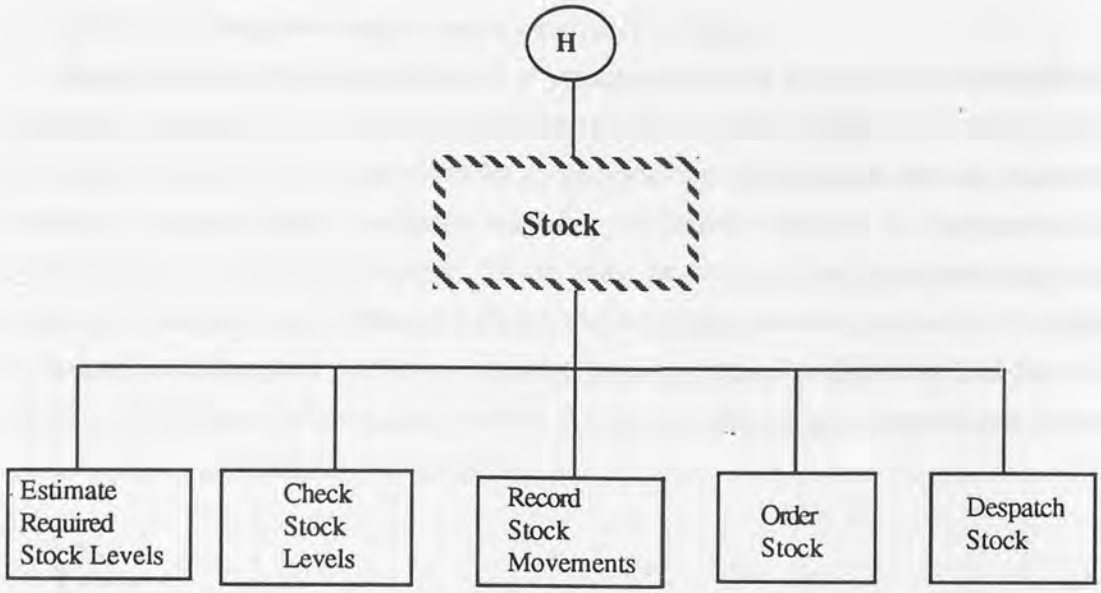


Fig. 4.16. Functional model: stock

The process can be compared with that of completing the engineering drawing when designing a machine or building.

The functions are shown in boxes, and they correspond to the boxes on the bottom row (or near bottom row) of the function hierarchy chart. Figure 4.19, for example, is represented in the functions of the third level of Figure 4.10. They represent the individual actions that need to be performed for the DLU to carry on its business.

Data flows from left to right across the page undergoing various changes on its way as it is processed. The 'bold' arrows going into the box show the events and the information that is captured during that function. It is collected during some processes, passed on to others and stored. Later on it may be retrieved from the store so that it can be used in another function. Some of this data is input (shown as dotted arrows in the figures) because the information is necessary for the particular processing. Thus in Figure 4.17, the correct answers are necessary for the tutor to mark a student's TMA.

As we have seen, events are depicted in these data flow diagrams as coming in from above the function boxes. Events correspond to happenings in the real world, and so are easy to understand. In Figure 4.18, for example, a registration form arrives, the tutor accepts the student, and if the finance officer accepts the fee arrangements, this event leads to an invoice being despatched along with the learning materials. Early attempts at information analysis concentrated on 'data' and 'processing', as the events are straightforward in business data processing. However, ignoring events is a mistake as it might, for instance, lead to sales not being recorded and potential customers not being served. In computer controlled systems, such as those at oil refineries and nuclear power stations, the data and processes to be programmed are usually relatively straightforward, but a failure to respond correctly to an event could result, in these

examples, in a dangerous explosion or a radioactive leak.

Some functions that are performed at present may have an external justification. For example, external bodies may lay down certain rules about student enrolments. One of the objectives of the system must be to provide the information that is required by statutory bodies. Other functions may be performed because 'it happens to be a convenient way of doing things'. These may be changed on implementing a new system if a better way is found of doing these things. The analyst has to distinguish between functions that 'serve an organisation's purpose' and others that 'serve the needs of a previous information system'. Analysts must design systems that serve the needs of the organisation, not its habits.

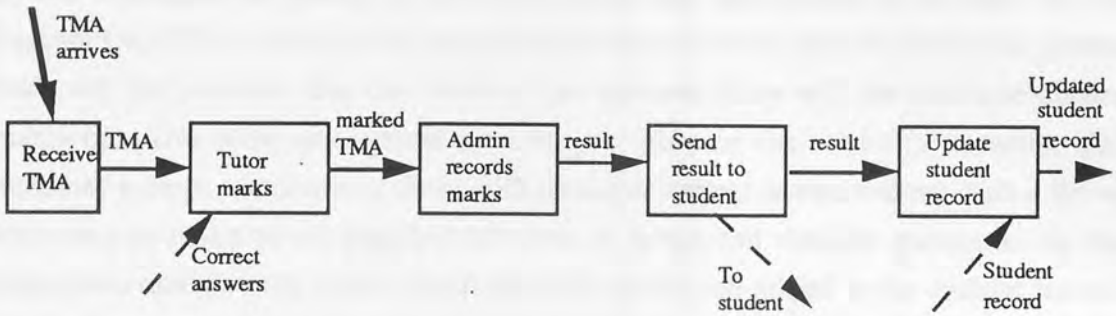


Fig. 4.17. Simplified data flow diagram 'Process TMA'

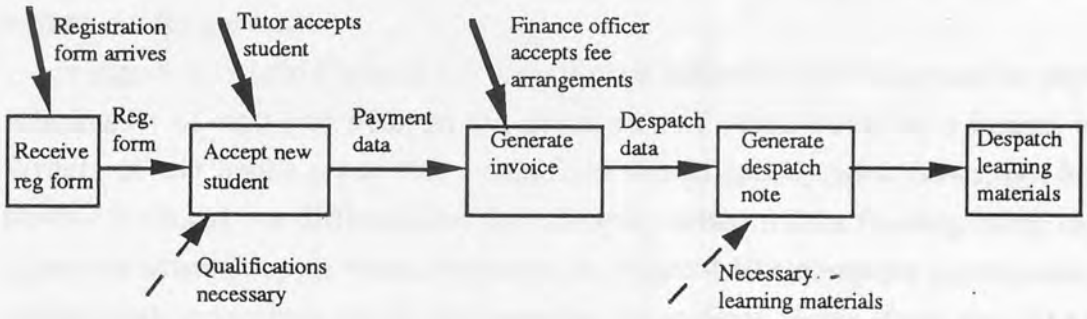


Fig. 4.18. Simplified data flow diagram 'Enrolment'

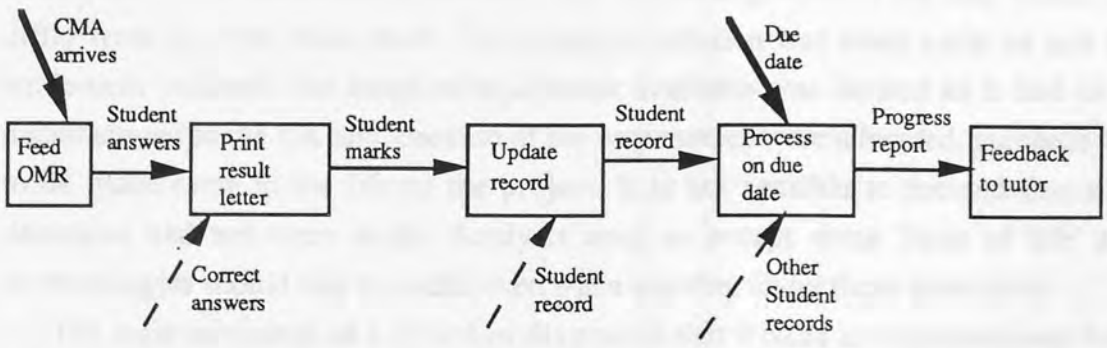


Fig. 4.19. Simplified data flow diagram 'Process CMA'

Figure 4.17 shows what happens when a tutor marked assignment (TMA) arrives. No receipting or batching process has been shown at the start, although these may well need to be carried out. On the diagram the TMA is passed to the tutor who marks the assignment. The marked assignment is passed to the administration which sends the result to the student. Data about the marks are also added to the student record. On the date that the assessment is due for completion, all the student records can be scanned to find out who has completed the work and who has not, and students who have not completed the work are sent letters to remind them that they have not submitted by the due date.

Figure 4.19 shows what happens when a computer marked assignment (CMA) arrives. Again, no receipting or batching process has been shown at the start. On the diagram the CMA is read by the optical mark reader (OMR) and the computer system reads off the answers that the student has marked (they will be multiple-choice questions). The computer system now has the data on the student's answers. The computer system can compare these with its list of correct answers along with a list of responses to make to all possible answers. A letter can then be generated on the assessment and give the marks. Data about the marks are added to the student record. On the date that the assessment is due for completion, all the student records can be scanned to find out who has completed the work and who has not, and who has passed the assessment and who has to make a second attempt. Finally, a progress report can be written out for the tutor.

In Figure 4.19 (and Figure 4.10) there is clear indication as to the possible physical solution (or at least part of it) in the description of what should be a logical view. Writers of text books argue that a data flow should specify what flows, not how it flows - it should not differentiate, for example, between data flowing using carrier pigeon or twisted copper wires. However, in Figure 4.19 a computer system using an optical mark recognition device for inputting the students' marks (from the CMA) has been suggested and the diagram shows the processing that follows its arrival. Considerations relevant to the implementation have been depicted in the logical description. This shows one of the ways that the contingencies of the 'real world' can differ from the 'text book ideal'. The computer solution was made early on and the equipment ordered. The range of equipment available was limited as it had to be manufactured in the UK and, because of the way monies were allocated, purchase had to be made early in the life of the project. It is not sensible to pretend that such decisions had not been made. Analysts need to accept these 'facts of life' and methodologies should still be useful even when working under these constraints.

The main advantage of a data flow diagram is that it takes an important step from the infinite subtlety of the human being to the simplistic approach of the computer.

When one person is explaining a system to another it is possible to move gradually from one phase to another, explaining in more depth where it seems necessary. Computers need every small step to be programmed, the programmer cannot 'assume' steps in a way that one human can when talking to another. The construction of these data flow diagrams represents a phase in the process of breaking complex processes into logical flows of simpler steps. They also identify the information that is passed between these steps and that obtained from files. Finally, they identify the events that call the processes into action. In Figure 4.19, 'feed OMR' and 'print result letter' are two of the steps; 'student answers' and 'student marks' represent information passing between steps; 'correct answers' and 'student record' represent information coming from files; and 'CMA arrives' and 'due date' represent events.

One way of checking the accuracy and completeness of the model is to create a matrix of events and functions and put an X at the intersection between an event row and a function column if that event triggers that function. Every event should trigger at least one function and every function should have a triggering event. If there are any gaps, then the analyst must find out what has been forgotten. Note that Figure 4.20 has only been partially completed.

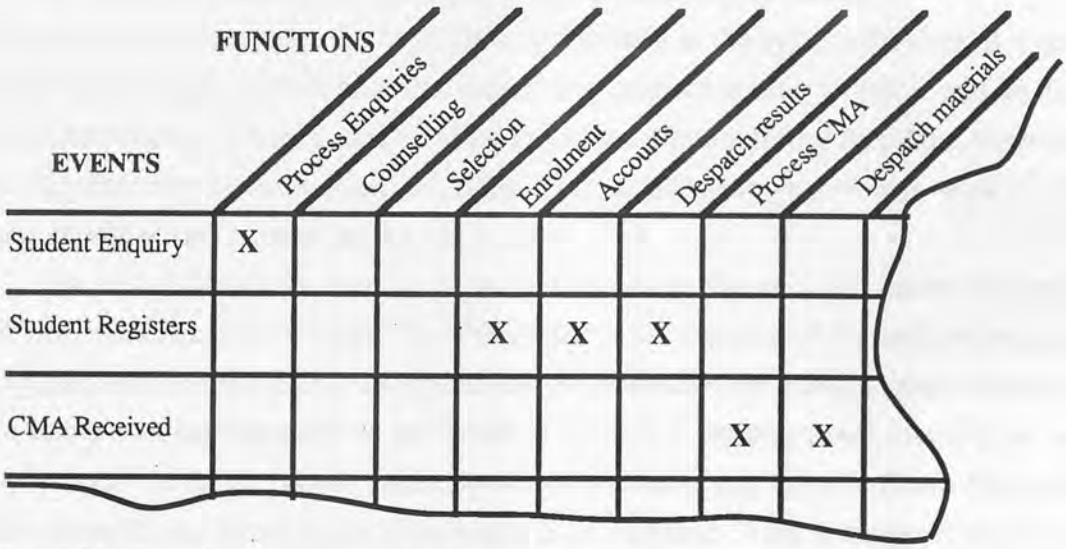


Fig. 4.20. Function/event matrix for DLU case (part)

The next step is to analyse the data and construct an entity model (Figure 4.21) and list the attributes and an identifier associated with each entity. For example, the student reference number may uniquely identify each entity occurrence of the entity 'student' (the student name is unlikely to be unique for all students).

Although it is a more formal representation than the rich picture, it is not too difficult to see what is going on by looking at the entity model - at least those aspects of the problem situation that it is supposed to represent.

When verifying the model, a good way to start the testing is to go to all the people who will use the system and ask them to list the sorts of information that they want to collect and the questions that they will want to be able to answer. Such questions might include "Have you marked the CMA I sent in last Friday?" or "What proportion of students are dropping out before they take the exams?" or "Has Ann Jones completed module 1 yet?". Armed with a list of questions, the analyst looks at the function chart, the data flow diagrams and the entity model to see if the necessary information was collected. Further, this process will detect information on the model that is not used.

Having created the entity model and listed the attributes, the next stage is to draw an entity life cycle which shows the changes of state that an entity goes through over time. It is a technique which represents a dynamic view of the system. For example, Figure 4.22 traces the life cycle of an occurrence of the entity 'enquirer' in the DLU system. There will always be an entity type 'enquirer', even if there are no enquirers presently in the system. If we look at a particular enquirer (an entity occurrence of the entity enquirer) then it will change state if any of the following happens:

- The enquirer enrols as a student.
- The enquirer withdraws by saying that he is not interested.
- The tutor declares that the enquirer is not suitable for the course.

If none of these happens, the enquirer could remain in the system for ever. It would be sensible therefore to declare a time limit for enquiry data after which it will be deleted from the system. There could be good reasons for maintaining this data, however, as an enquirer may be encouraged to re-apply in the following year or may want to hear of new modules and courses as they develop.

The entity life cycle chart, such as that shown in Figure 4.22, shows the progress of any particular type of entity. The things that cause the state of the entity to change are functions and events. Entity life cycle analysis identifies the various possible states that an entity can legitimately be in. There is always a starting point, usually an event, which sets the entity into its initial state and a terminating point to finish the cycle. In the example, the initial state of the entity is as 'enquirer'. This is triggered by an event, which in the example is the receipt of an enquiry. The entity changes state as a result of the receipt of an application form, the admissions function and whether the course has been completed.

The next stage of Multiview begins the process of moving away from the analysis discussed so far towards design issues. The first of these concerns the balancing of the social and technical needs of the organisation.

The major social objectives of the distance learning unit were:

- To provide quick feedback to students on their progress, including answering telephone calls.

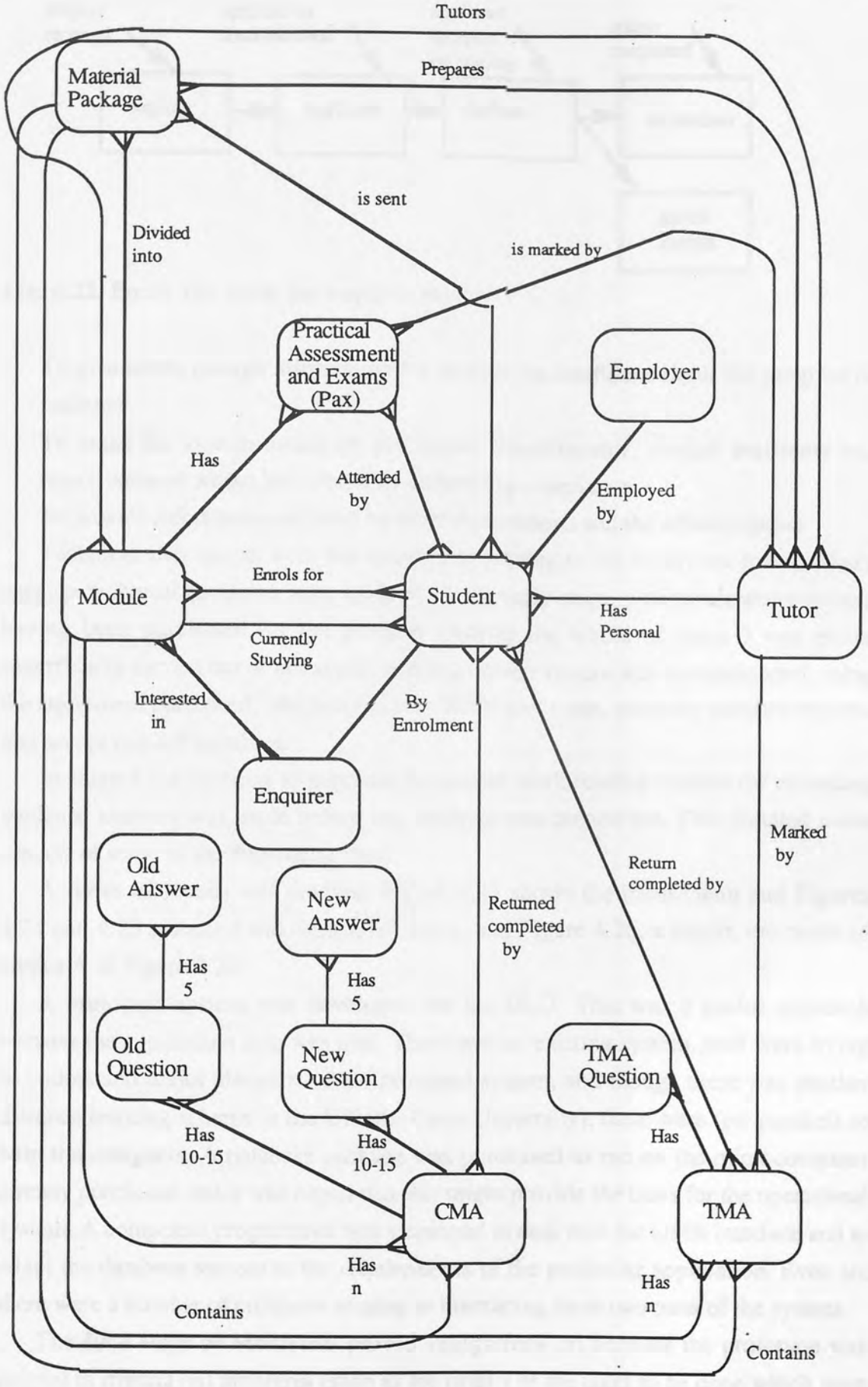


Fig. 4.21. Entity model for Distance Learning Unit (part)

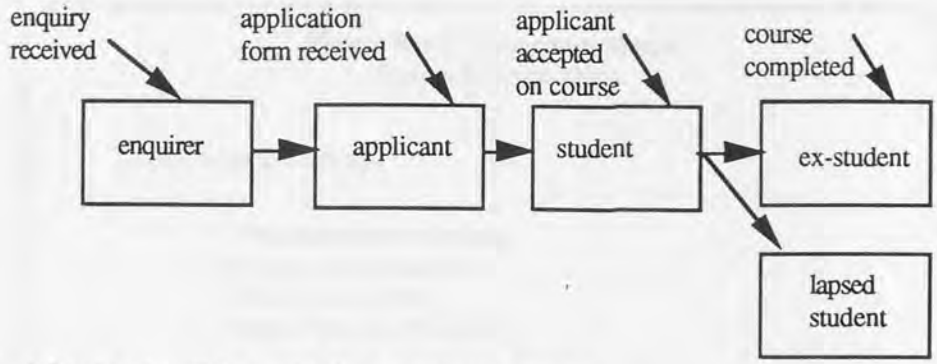


Fig. 4.22. Entity life cycle for enquirer record.

- To give tutors enough information for them to be confident about the progress of students.
- To make the system usable by the course administrator, clerical assistants and tutors, some of whom have had little computing experience.
- To provide information required by other departments and the administration.

Technical alternatives were not assessed according to the Multiview methodology because technical decisions were made at a very early stage, a microcomputer system having been purchased for the purpose (indeed the whole of stage 3 was either superficially carried out or not at all). A menu driven system was recommended, using the equipment purchased, which could take batches of input, generate standard reports, and accept one-off enquiries.

In stage 4 the decision to purchase an optical mark reading system for recording students' answers was made before any analysis was carried out. This dictated some aspects of some of the documents used.

A series of menus was devised. Figure 4.23 shows the main menu and Figures 4.24 and 4.25 a second and third level menu, and Figure 4.26, a report, the result of choice A in Figure 4.25.

A prototype system was developed for the DLU. This was a useful approach because the application area was new. There was no existing system, staff were trying to understand major elements of the proposed system, and though there was another distance learning scheme in the UK (the Open University), there were few parallels to help the designers. A database package was purchased to run on the microcomputer already purchased and it was hoped that this might provide the basis for the operational system. A competent programmer was employed to deal with the OMR interface and to adapt the database system to the requirements of the particular application. Even so, there were a number of problems relating to interfacing these two parts of the system.

The final stage of Multiview proved straightforward because the prototype was helpful in ironing out problems (such as the details of the tasks to be done which were outlined in the function charts; and the data storage requirements from the entity

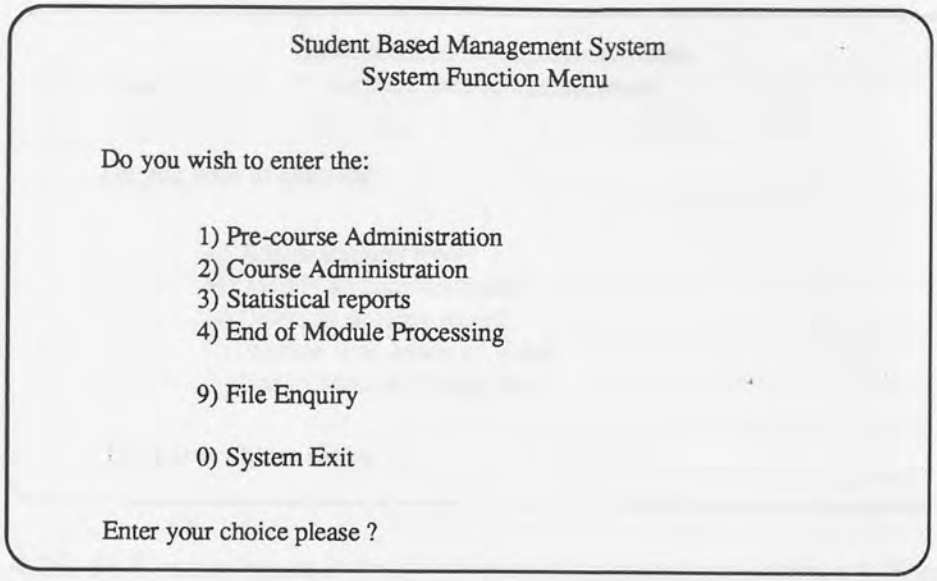


Fig. 4.23. DLU main menu

modelling stage). Furthermore, it proved adequate for the operational version. In other words, once the data was collected and validated, the users had had their requests for modification carried out and had been fully trained, the prototype 'became' the operational system. Again, however, some steps were necessarily omitted. For example, not all the people who would use the system were in place, and therefore training and testing was not carried out for them. Furthermore, some criticisms of the system were not dealt with, in particular an implementation of a by-pass system for menus, so that experienced users could go quickly to the desired part of the system.

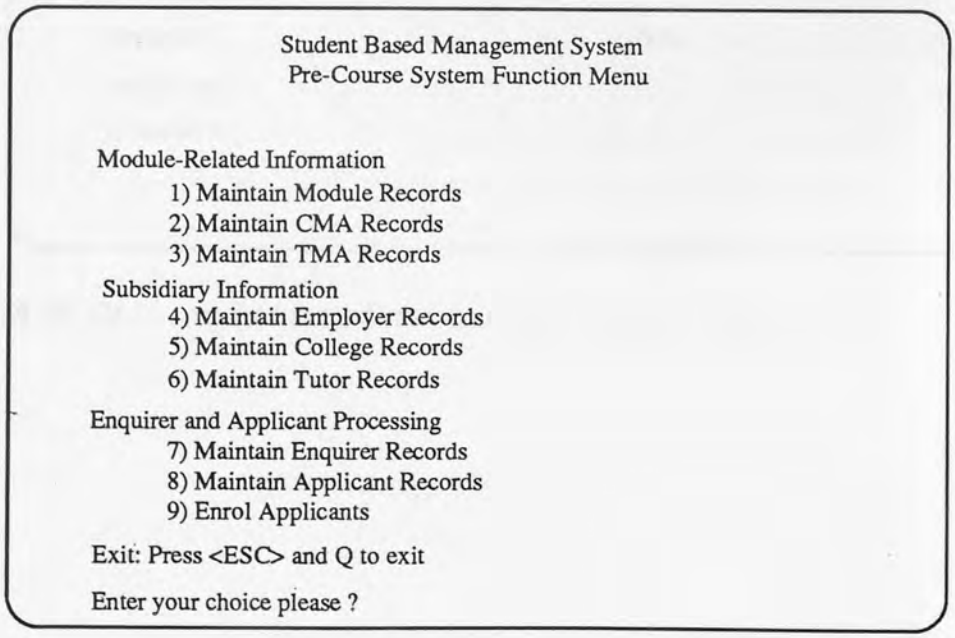


Fig. 4.24. DLU main menu 2: Pre-course administration (choice 1 Figure 4.23)

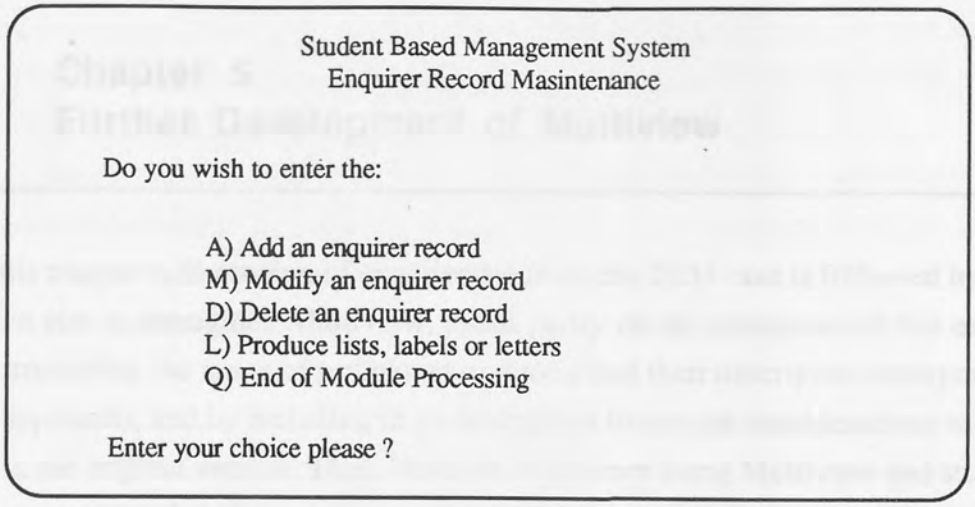


Fig. 4.25. DLU main menu 3: enquirer record maintenance (choice 9 Figure 4.24)

An analysis of the experience gained from using Multiview in the DLU case is given in Section 5.1. This is followed by sections describing those aspects of Multiview which should be strengthened on the basis of this experience.

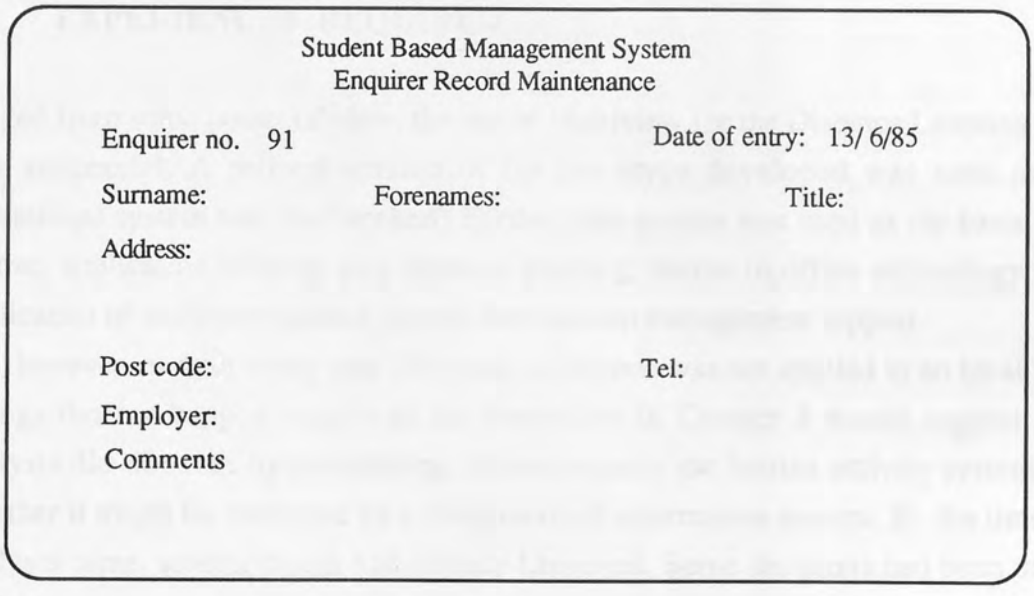


Fig. 4.26. DLU enquirer record maintenance (choice A Figure 4.25)

Chapter 5
Further Development of Multiview

In this chapter a discussion of experiences from the DLU case is followed by sections which aim to strengthen Multiview, based partly on an assessment of this experience (by improving the range of techniques and tools and their description incorporated into the approach), and by including in its description important considerations which were not in the original version. Thus, based on experience using Multiview and study of the literature since that time, a new version could be compiled. The updated approach is described fully in Avison & Wood-Harper (1990). The use of this new version is seen in two cases (described in Chapters 6 and 7). Some lessons, conclusions and suggestions for further work are found in Chapter 8.

5.1 EXPERIENCE FROM THE DLU CASE AND FURTHER EXPERIENCES REQUIRED

Judged from some points of view, the use of Multiview for the Distance Learning Unit was successful. A refined version of the prototype developed was used as the operational system and this 'worked'. Further, this system was used as the basis for a second application relating to a distance learning course in office technology. The application of Multiview gained general user and top management support.

However, as with every 'real life' case, Multiview was not applied in an ideal way. Things did not happen exactly as the exposition in Chapter 3 would suggest. The analysts did not start by considering dispassionately the human activity system and whether it might be improved by a computerised information system. By the time the analysts came, several things had already happened. Some decisions had been made, such as that to use optical mark recognition equipment for assessing some of the students' work and the contract to create a particular computerised system had been awarded, a price agreed and some hardware already delivered. Thus a significant amount of the analysis and design process had already been undertaken. This was not ideal (because the analysts would normally want to assess various hardware options following full investigation), but this was a 'fact of life' relating to the way monies were allocated.

It was therefore necessary to do some inventive thinking about hardware requirements. Some initial work was sketched out relating to files, programs, storage volumes and the amount of printing that needed to be done. Further, the socio technical

aspects amounted to only a quick appraisal leading to the view that the computer would need to be run in multi-user mode in order to cope with the variety of tasks necessary when the student numbers built up.

A further difficulty concerned the inability to take a 'systems view' with regard to recommendations. It would have been useful to take into account other application areas, in particular, a second distance learning project in information technology for office workers. However, because of the way the project was financed, purchases had to be justified on the basis of this one application.

Another consideration was that some aspects of the system were not as yet in place: there were no students nor clerical staff to comment on the proposals. The human-computer interface for some users had to be analysed by guesswork, as not even advertisements for some posts had been drawn up. Further, the Department of Education and Science had yet to decide what statistics and reports it wanted. The entity and function models also required some guesswork along with the knowledge of existing requirements for student records. Prototyping the system proved invaluable, and improvements were made as users were added to the system and comments were forthcoming or difficulties encountered.

Despite all these difficulties in the problem situation - complications such as these will always be found in the real world - Multiview was helpful. Many systems development methodologies assume that everything will go according to plan and to an 'ideal' pattern. This is rarely the case. Multiview is designed to be flexible and the elements of the framework adapted to the problem situation.

Many of the techniques provided in Multiview worked well in the case. Rich pictures, somewhat innovative at the time, proved an excellent communication tool in the problem situation. They were particularly useful in this situation as there was no previous system and therefore there needed to be a good communication and analysis tool. The rich picture presented as Figure 4.1 encapsulates a very complex situation, with many actors and externalities. It proved to be a good communication tool with people discussing, refining, disagreeing, but above all thinking about this very complex situation. Problems emerged, for example, some people objected to the conflict situations (crossed swords) being represented in the diagram, preferring to 'keep quiet' about disagreements.

However, in general, users liked the format and were relieved that the first communication tool used between 'computer people' and themselves were human-oriented rather than computer-oriented. They could see themselves and many other aspects of the problem situation documented. This ability of rich pictures to include the actors, problems, roles, climate, and other aspects in one diagram is unusual.

One problem was that of identifying the system owner. This was not resolved, but

it was useful to expose the problem. The lack of consensus over ownership (between the Polytechnic administration, the MSC and the Paintmakers' Association) later led to a 'battle over copyright', the software produced being highly marketable.

The other aspects of Checkland's methodology incorporated into Multiview, root definitions and conceptual modelling, also proved useful. There was a natural progression from rich pictures to root definitions and conceptual models, and the construction of the relevant system and root definition enabled an identification of the various concerns, and the separation of the major concern from side issues. The relative success through using these techniques was particularly heartening as the DLU system was new. Most conventional methodologies assume that a system of some sort already exists. Further, the use of these techniques was new to most of the participants.

The use of conceptual models proved more difficult. Some users did not see their significance and users found it difficult to go from one level to others. There was also initial confusion regarding the term itself, as the conceptual model is sometimes the name given to database models. For example, Avison (1985), Date (1986) and Korth & Silberschatz (1986) take this view. In Multiview, the term is used as in Checkland (1981) and as illustrated in Chapter 4. It was not possible to compare the conceptual models with the present real-world system, which would normally be done in order to uncover discrepancies and recommend changes. The conceptual model therefore represented recommendations for the real world operation.

Other techniques, such as entity modelling, though complex, were successful in this application partly because they were well known and the systems analyst was familiar with their use. On the other hand, some other techniques specified in Multiview were either not used or proved inadequate. This was particularly true of some aspects of Stage 2, functional analysis and entity modelling. Functional decomposition, used in a number of techniques, proved difficult for this group of users (this was not an experience which occurred in all applications of Multiview). The function models lent themselves to decomposition with greater ease than other techniques, at least to this group of users. There was also some difficulty and confusion over differentiating between what is logically necessary to do from the method by which to do it, in other words, the separation of logical representation and its physical solution.

Further, users found that some techniques, such as data flow diagrams and entity life cycles, were poorly explained in the original exposition of Multiview. Some documentation was also inadequate. For example, there were no forms or advice on documenting attributes, entities and relationships. It is important to address these weaknesses. This could be done by specifying them more clearly, incorporate more recent and improved versions of the techniques, and to suggest other documentation aids.

Further guidance on other concerns would also have been useful. For example, an important aspect of this case was the purchase of microcomputer packages, including a database management system, and yet no advice was given in the original exposition of Multiview for comparing application packages and, in particular, database systems. This also needs to be addressed. Other important aspects of the case, for example, tools supporting prototyping, were also poorly described in the original version even though the use of the prototyping approach did prove successful. The prototype, following several iterations of user comment and improvement, was adopted as the operational system.

One aspect of this research was therefore to refine the explanation of some techniques and tools for users of the methodology. On some occasions, techniques and tools which were omitted from the original exposition of Multiview would have helped the users in the DLU case and these need to be included into the exposition of Multiview. These are described in this chapter.

Part of this research concerns the problems of teaching information systems. Multiview might be used as a teaching tool in courses in information systems. A criticism of the original text on Multiview has been that it dealt with some techniques either superficially or not at all. An improved explanation of the techniques and tools, along with the exposition of a blended contingent approach to information systems development, is a contribution to this end.

The DLU case did not follow exactly the methodology as described in Chapter 3. This was due to cost, time and political constraints applying in this one problem situation. Some phases were omitted or reduced in scale and others carried out in a different sequence. The socio-technical phase was not fully explored, mainly because many of the technical decisions had already been made. The political dimension transcended the methodology rationale, in particular in that purchase decisions were made without consideration of the factors implied by the methodology. Other difficulties included the problem of identifying the interest groups or stakeholders, and in some aspects this proved impossible in that they were not in place as yet.

This leads us to stating a second major weakness of the original exposition of Multiview (Wood-Harper et al, 1985): that in a contingency approach one case study is not sufficient to illustrate its use. It would have been a weakness of this research if only one case had been used to illustrate its use and to expose any problems. This view is further evidenced by assessing the inspection copy replies from academics and researchers, where it is seen that the one major criticism levied at the text was the lack of further case studies to explain the application of the approach. As well as to improve the exposition of a number of techniques and tools of Multiview, a second major function of the research described in this thesis, therefore, was to use Multiview in

other situations and, in general, gain more experience of Multiview in action, in order to refine it further. Two other major cases are described in this thesis and formed major aspects of this research. They are described in Chapters 6 and 7. Lessons learnt and conclusions reached following assessment of all three cases are discussed in Chapter 8.

5.2 IMPROVEMENTS AND ADDITIONS TO TECHNIQUES

This section describes improvements to the various techniques discussed in the original version of Multiview, in particular data flow diagrams and entity life histories, and also techniques not described in Wood-Harper et al (1985), for example, normalisation, decision tables, decision trees, structure diagrams and structured English. Only a brief overview of each technique is given here. They are further described in Avison & Wood-Harper (1990) and in the various texts and papers quoted. Further, their use is generally described in relation to the cases in the two chapters following. Where they are not illustrated in the description of the cases, an example of their use is given here.

This section first gives an improved description of data flow diagrams. They proved useful in the first case, but the description was found wanting. The second technique described is that of normalisation which is now a standard method of data modelling aiming to make the data model more useful for database design, for example. Entity life cycles, like data flow diagrams, were poorly described in the original exposition of Multiview and these are described next. Finally, four techniques are described which were not in the original exposition of Multiview: structured English, structure diagrams, decision trees and decision tables. All four techniques help in analysing, documenting and specifying processes. Structure diagrams are in fact extensions to function diagrams so that design as well as analysis considerations are included. Not all these techniques will be used in any application of Multiview, but they add to the techniques available to be used where appropriate.

5.2.1 Data Flow Diagrams

Data flow diagrams describe the transformation of inputs into outputs, providing a convenient overview of the main functional components of the system (Yourdon, 1989). They were first used in the software engineering field as a notation for studying systems design issues (Stevens, Myers & Constantine, 1974). The data flow diagrams described in the original text and in this thesis in the DLU case have been modified because the diagrams did not specify in detail nor in a well documented form the various components. Figures 6.4, 7.10, 7.11 and 7.12 give examples using the diagrammatic technique based on Gane & Sarson (1979). These data flow diagrams (DFD) have four basic constructs: data flow, process, data store and external entity.

The data flow: Data flow is represented by an arrow (as it is in the simplified model) and depicts the fact that some data is flowing or moving from one process to another. A number of analogies are commonly used to illustrate this. Gane and Sarson suggest that we think of the arrow as a pipeline down which 'parcels' of data are sent, and Page-Jones (1980) states that data flow is like a conveyor belt in a factory which takes data from one 'worker' to another. Each 'worker' then performs some process on that data which may result in another data flow on the conveyor belt. These processes are the second element of the DFD.

The processes: The processes or tasks performed on the data flows are represented by a soft box (a box with rounded edges) as against the 'hard box' of the earlier model (Figures 4.17, 4.18 and 4.19). The process transforms the data flow by either changing the structure of the data or by generating new information from the data. The conventions used for the process symbol in Figure 7.11 are as follows. The top compartment contains a reference number for the process and a lower one might contain the description of the process (which was not completed in the examples used in the cases). A process must have at least one data flow coming into it and at least one leaving it. There is no concept of a process without data flows, a process cannot exist independently.

The data store: If a process cannot terminate a data flow because it must output something, then where do the data flows stop? There are two places. The first is the data store, which can be envisaged as a file, although it is not necessarily a computer file or even a manual record in a filing cabinet. It can be a very temporary repository of data, for example, a shopping list or a transaction record. A data store symbol is a pair of parallel lines with one end closed and a compartment for a reference code and a compartment for the name of the data store.

The source or sink (External Entity): The second way of terminating a data flow in a system is by directing the flow to a sink. The sink may, for example, be a supplier to whom we send an order for stock. The supplier is a sink in the sense that the data flow does not necessarily continue. Sinks are usually entities that are external to the organisation in question, although they need not be, another department may be a sink. It depends on where the boundaries of the system under consideration are drawn. The original source of a data flow is the opposite to a sink, although it may be the same entity. For example, a customer is the source of an order and a sink for a despatch note. Sinks and sources are represented by the same symbol which is a square (thickly lined on two sides) in this convention. Sources and sinks are often termed 'external entities'.

One of the most important features of the data flow diagram is the ability to construct a variety of levels of data flow diagram according to the level of abstraction required. This means that an overview diagram can be consulted in order to obtain a

high level (overview) understanding of the system. When a particular area of interest has been identified, then this area can be examined at a more detailed level. The different levels of diagram must be consistent with each other in that the data flows present on the higher levels should exist on the lower levels as well. In essence it is the processes which are expanded at a greater level of detail as we move down the levels of diagram. This 'levelling' process gives the technique its top-down characteristic. This can be seen from Figures 7.10, 7.11 and 7.12.

The details of errors and exceptions are not shown on high level diagrams as it would confuse the picture with detail that is not required at an overview level. What is required at an overview level is 'normal' processing and data flows. To include errors and exceptions might double the size of the diagram and remove its overview characteristics.

The problem is that it is sometimes difficult to decide what constitutes an error or an exception, and what is normal. Some common guidelines suggest that if an occurrence of a process or data flow is relatively rare, then it should be regarded as an exception. However, if it is financially significant, it should be taken as part of normal processing. Overall, it depends on the audience or use to be made of the data flow diagram as to exactly what is included. At the lowest level, all the detail, including errors and exceptions, should be shown.

5.2.2 Normalisation

Normalisation helps to ameliorate the anomalies when inserting, deleting and updating entity occurrences. A normalised relation has a simpler structure than an unnormalised one and is easier to change (see Date, 1986, for the rationale and details of the process). Normalisation is in a number of methodologies to improve database design, and is a feature of most data analysis approaches (Section 2.8). It is used in Multiview following entity analysis. The example of the normalisation process chosen below is taken using data from the DLU case, though normalisation was not carried out in that particular case. The process described in Multiview has three steps (there are further steps in normalisation, described by Date as 'advanced', but that described here has proved sufficient for most problem situations):

1 *First normal form*: Ensure that all the attributes are atomic (that is, in the smallest possible components). This means that each has only one possible value and not a set of values. For example, if an entity is presented as:

COURSE, COURSE-NAME, MODULE-DETAILS

module-details has to be defined as a set of atomic attributes, not as a group item, and has to be broken down into its constituents. This has been done in the entity course details in Figure 5.1. Another aspect of first normal form is the filling in of the details, also seen in Figure 5.1. Whereas the order of the tuples (each row of the relation) is insignificant in the second relation, the order is important in the unnormalised version because some of the information would be lost if the tuples were ordered differently. The key (or identifier), which uniquely identifies each tuple occurrence, of the entity course details consists of course and module (a composite key).

COURSE DETAILS (UNNORMALISED)

COURSE	COURSE-NAME	MODULE	M-NAME	STATUS	M-PTS
C14	PAINT FOUNDATION	M17	DESIGN-1	BASIC	4
		M19	CHEMISTRY-1		
		M22	TECHNOLOGY-1		
		M34	COLOUR	INT	6
C15	PAINT QUALIFIER	M41	DESIGN-2	INT	6
		M46	TECHNOLOGY-2		
		M55	COMPUTING	ADV	8

COURSE DETAILS (FIRST NORMAL FORM)

COURSE	COURSE-NAME	MODULE	M-NAME	STATUS	M-PTS
C14	PAINT FOUNDATION	M17	DESIGN-1	BASIC	4
C14	PAINT FOUNDATION	M19	CHEMISTRY-1	BASIC	4
C14	PAINT FOUNDATION	M22	TECHNOLOGY-1	BASIC	4
C14	PAINT FOUNDATION	M34	COLOUR	INT	6
C15	PAINT QUALIFIER	M41	DESIGN-2	INT	6
C15	PAINT QUALIFIER	M46	TECHNOLOGY-2	INT	6
C15	PAINT QUALIFIER	M55	COMPUTING	ADV	8

Fig.5.1. First normal form

2 *Second normal form*: Check that all non-key attributes belong fully to one entity type, that is, they are fully dependent on - give facts about - all of the key. This is called functional dependency. If this check fails, entity types (new or existing) are created where the attributes are fully dependent on the whole key. The products from this process are called 'second normal form'. For example, course details, in first normal form, is not in second normal form because m-name, status and m-pts

are functionally dependent on - they are facts about - module, which is only part of the key (course and module). They do not represent facts about course. Thus two relations are formed from the first normal form relation of Figure 5.1. The entity module in Figure 5.2 will have the attribute 'module' as the key and we are left with a second relation consisting of course and module (the composite key) and the non-key attribute course-name.

If we look at the second relation in Figure 5.2, we see that it is also not in second normal form because course-name is functionally dependent on course, which is only part of the key. We therefore have to separate out this relation leading to Figure 5.3. We finish with three relations in second normal form, module (Figure 5.2) and course and course-module (Figure 5.3). The relation course now only has two entries, as we do not want entries which are exactly the same. The relation course-module is 'all key', that is all the attributes are key attributes. Information would be lost if we left out this relation, that is, the information that modules M17, M19, M22 and M34 are included in the course C14, and M41, M46 and M55 are included in the course C15.

MODULE

MODULE	M-NAME	STATUS	M-PTS
M17	DESIGN-1	BASIC	4
M19	CHEMISTRY-1	BASIC	4
M22	TECHNOLOGY-1	BASIC	4
M34	COLOUR	INT	6
M41	DESIGN-2	INT	6
M46	TECHNOLOGY-1	INT	6
M55	COMPUTING	ADV	8

COURSE DETAILS

COURSE	COURSE-NAME	MODULE
C14	PAINT FOUNDATION	M17
C14	PAINT FOUNDATION	M19
C14	PAINT FOUNDATION	M22
C14	PAINT FOUNDATION	M34
C15	PAINT QUALIFIER	M41
C15	PAINT QUALIFIER	M46
C15	PAINT QUALIFIER	M55

Fig.5.2. Towards second normal form

COURSE

COURSE	COURSE-NAME
C14	PAINT FOUNDATION
C15	PAINT QUALIFIER

COURSE-MODULE

COURSE	MODULE
C14	M17
C14	M19
C14	M22
C14	M34
C15	M41
C15	M46
C15	M55

Fig.5.3. Second normal form

3 *Third normal form*: Check to see if the value of a non-identifier attribute can be deduced from the values stored within other non-identifier attributes in the same entity occurrence or occurrences in the same entity type (this is known as transitive dependency). This check is to ensure that redundancy is not present due to dependency upon non-identifier attributes. If this redundancy occurs then the dependent attribute can be removed to new entity types. The resulting products from this process are called the 'third normal form'. In Figure 5.2, the attribute m-pts is transitively dependent on status (not a key) in the relation module and the third normal form version therefore has two relations, status and module, for the relation module in second normal form.

MODULE

MODULE	M-NAME	STATUS
M17	DESIGN-1	BASIC
M19	CHEMISTRY-1	BASIC
M22	TECHNOLOGY-1	BASIC
M34	COLOUR	INT
M41	DESIGN-2	INT
M46	TECHNOLOGY-2	INT
M55	COMPUTING	ADV

STATUS

STATUS	M-PTS
BASIC	4
INT	6
ADV	8

Fig.5.4. Third normal form

The relation status had repeating entries, so these were not included. The final set of relations in third normal form includes module and status (from Figure 5.4), course-module and course (from Figure 5.3).

5.2.3 Entity life cycle

Again, the entity life cycle diagram used in the original version of Multiview was somewhat rudimentary, and a more sophisticated version is now incorporated. The convention used is that from D2S2 (Macdonald & Palmer, 1982), a precursor of the methodology Information Engineering, but the technique is used in many methodologies, for example SSADM - where it is called an entity life history - (Downs et al, 1988) and Jackson Systems Development (Jackson, 1983).

The entity life cycle is not, despite its name, a technique of data analysis, but more a technique of functional analysis. This is because the things that cause the state of the entity to change are functions and events, and it is these changes that are being analysed. The objective of entity life cycle analysis is to identify the various possible states that an entity can legitimately be in over time. It is a technique which shows some dynamic aspects of the application (most techniques used in information systems express static aspects of the system). The sub-objectives, or by-products, of entity life cycle analysis are to identify the functions in which the entity type is involved and to discover any functions that have not been identified elsewhere. It may also identify valid (and invalid) sequences of functions, not identified previously, and it can form the outline design for transaction processing systems. Thus, as well as being a useful analysis technique in its own right, it is a useful exercise to perform as a validation of other function analysis techniques.

The documentation of the states of an entity in a diagram is one of the most powerful features of entity life cycle analysis. The diagram provides a pictorial way of communication that enables users to validate the accuracy or otherwise of the analysis.

Figure 5.5 shows the different symbols used and Figure 6.11 shows an example. There is always a starting point, usually an event, which sets the entity into its initial state. There is also always (or should always be) an end or terminating point to finish the life cycle. In between, there may be many different states of the entity.

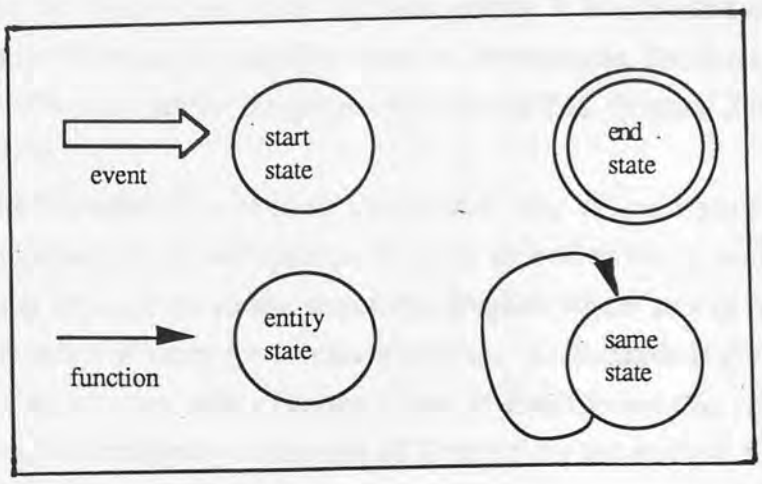


Fig. 5.5. Entity life cycle symbols

The technique is useful in identifying the states of an entity, the functions that cause the states of an entity to change, and any sequences that are implied. It is also important to identify the terminating states of the entity. Some systems have not always done this and found that at a later date they have no way of getting rid of entity occurrences. One example of this, from Crowe & Avison (1980), is provided by a vehicle spare parts system for a large organisation. In this system vehicles require specific parts, but what is not known is which parts are used in specific vehicles. The result is that when vehicles become obsolete there is no way of withdrawing the parts that are used in that vehicle only. It is too dangerous to withdraw all the parts required by the obsolete vehicle, as many of these parts will be common to other vehicles. This results in a database which is continually growing as new vehicles and their parts are added. If there had been an entity life cycle analysis performed on the entity part, it would have been discovered that the entity occurrence did not terminate. The likelihood is that the organisation would then have designed a function to associate parts with vehicles and thus be able to terminate the entity. The process of normalisation (Section 5.2.2) also facilitates the resolution of this particular problem.

5.2.4 Structured English

No language was specified for formulating the technical specification in the original version of Multiview. Such a language can be useful when setting up a prototype (indeed the language may be used as the input to a program generator). Structured English is very like a 'readable' computer program, it aims to produce unambiguous logic which is easy to understand and not open to misinterpretation. It is not English, which is ambiguous and therefore unsuitable. Nor is it a programming language, though it can be readily converted to a computer program. It is a strict and logical form of English and the constructs reflect structured programming. It is a useful technique to express logic in a system, though the decision tree or decision table are more suitable tools where the system has many decision points. It is a feature of most structured methodologies (Section 2.7) and described in, for example, DeMarco (1979) and Gane & Sarson (1979), but similar languages appeared as PSL (Section 2.6) and PDL (Caine & Kent, 1975).

Structured English is a concise and precise way of specifying a process, and is readily understandable by the systems designer as well as being readily converted to a program. It is appropriate to use structured English where sets of operations may be repeated a number of times for certain conditions. An example is given in Figure 6.14. Structured English uses only a limited subset of English and this vocabulary is exact. This ensures less ambiguity in the use of 'English' by the analyst. Further, by the use of text indentation, the logic of the process can be shown more easily. (As with all

these techniques, however, though the logic can be formally expressed, there is no guarantee that the expression in structured English is correct. That will depend on the investigation in the first place.)

Structured English has an:

IF condition 1 (is true)
THEN action 2 (is to be carried out)
ELSE (not condition 1)
SO action 1 (to be carried out)

construct. Conditions can include equal, not equal, greater than, less than, and so on. The words in capitals are keywords in structured English and have an unambiguous meaning in this context. They include SO, REPEAT and UNTIL, as well as the aforementioned IF, THEN and ELSE keywords. The logic of a structured English construct is expressed as a combination of sequential, decision, case and repetition structures.

Functional decomposition can be supported in structured English by an:

IF condition
THEN do a named set of operations
(specified at a lower level)

type construct.

In structured English any logical specification can be written using the four basic structures of :

- *Sequencing*, shows the order of processing of a group of instructions, but has no repetition or branching built into it.
- *Selection*, through the decision process facilitates the choice of those conditions where a particular action or set of actions (or another decision and selection) are to be carried out.
- *Cases*, represent a special type of decision structure (a special kind of selection), where there are several possibilities, but they never occur in combination. In other words, they are mutually exclusive.
- *Repetition*, or loop instructions, facilitate the same action or set of actions to be carried out a number of times, depending on a conditional statement.

DeMarco (1979) discusses structured English at length and he argues that though there are many advantages of its use, in particular its ability to describe many aspects of analysis, its conciseness, precision and readability, and the speed with which it can be written, there are disadvantages as well. DeMarco highlights the time it takes to build up skills in its use and the fact that it is alien to many users (despite being English-like). Indeed, this aspect might be misleading to users, because the structured English meanings are not exactly the same as their natural language counterparts.

5.2.5 Structure Diagrams

This technique can be seen in Yourdon & Constantine (1978), DeMarco (1979) and a number of other methodologies use them, particularly structured approaches (Section 2.7). A structure diagram or chart is a graphical tool for representing hierarchy. Whereas a data flow diagram is a statement of requirement, declaring what has to be accomplished, a structure chart is a statement of design, declaring how the requirement might be met. It is therefore an extension of the function diagrams from analysis to include design aspects.

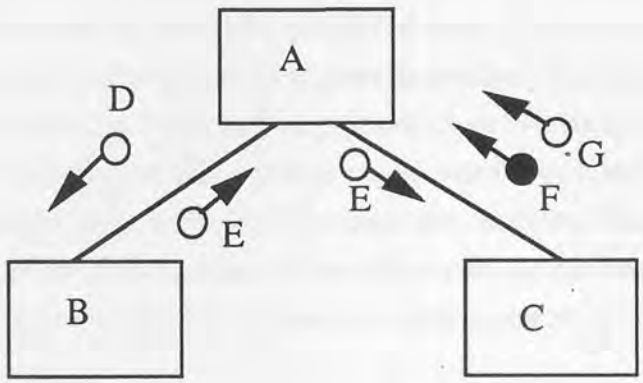


Fig.5.6. Structure diagram

The structure chart is a series of boxes (representing processes or modules) and connecting lines (representing links to subordinate processes) which are arranged in a hierarchy. The basic diagram is shown as Figure 5.6. This structure chart shows that:

- Module A can call module B and also module C. No sequencing for these calls nor whether they actually occur is implied by the diagraming notation. When the subordinate process terminates, control goes back to the calling process.
- When A calls B, it sends data of type D to B. When B terminates, it returns data of type E to A. Similarly, A communicates with C using data of types E and G.
- When C terminates, it sends a flag of type F to A. A flag is used as a flow of control data.

A full example is given in the third case study (Chapter 7) in Figures 7.5, 7.8 and 7.9.

5.2.6 Decision trees

Decision trees and decision tables are tools which aim to facilitate the documentation of process logic, particularly where there are many decision alternatives. No such documentation aids were described in the earlier exposition of Multiview. They are both standard techniques of structured approaches (Section 2.7). A decision tree illustrates the actions to be taken at each decision point. Each condition will determine the particular branch to be followed. At the end of each branch there will either be the

action to be taken or further decision points. Any number of decision points can be represented, though the greater the complexity, then the more difficult the set of rules will be to follow.

When constructing a decision tree, the problem must be stated in terms of conditions (possible alternative situations) and actions (things to do). It is often convenient to follow a stepwise refinement process when constructing the tree, breaking up the largest condition to basic conditions, until the complete tree is formulated.

An example of a decision tree is given in Figure 5.7. At the first decision point, the customer is classified into one of two types, private or trade. These are two conditions. If the customer is trade, then a second decision point is reached. Has the customer been trading with us for less than five years or five years or more? - two further conditions. If the customer has been trading for five years or more, then the customer can obtain up to £5,000 credit, otherwise only up to £1,000 credit can be given. These are the two actions corresponding to the combinations of conditions. If the customer was deemed private at the first decision point, then the action is to offer no credit at all.

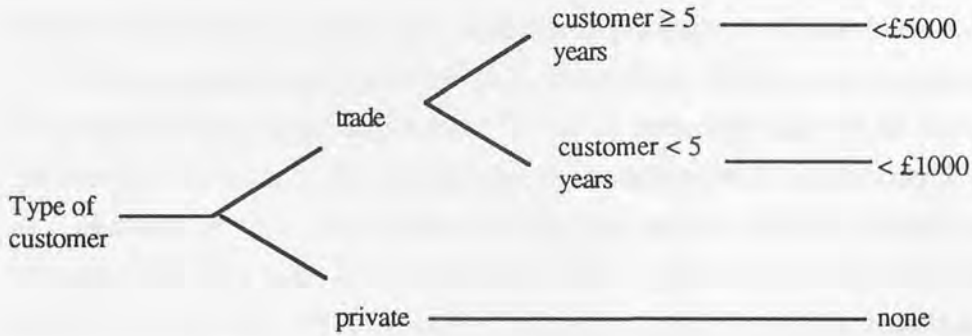


Fig.5.7. Example of a decision tree

Decision trees are constructed by first identifying the conditions, actions, and unless/however/but structures from a narrative statement (or direct from users) of the situation being analysed. Each sentence may form a 'mini' decision tree and these could be joined together to form the version which will be verified by the users. Sometimes it is not possible to complete the decision tree because information has not been given in the narrative or statement. For example, one branch of the decision tree may be identified, but no indication is given on the action to take on this branch. In such cases the analyst has to carry out a further investigation by interviewing staff or by using another method of systems investigation.

Decision trees prove to be a good method of showing the basics of a decision, that is, the possible actions that might be taken at a particular decision point and the set of

values that leads to each of these actions. It is easy for the user to verify whether the analyst has understood the procedures.

Sometimes it is possible to associate probability scores for each branch once the decision tree is drawn. With this information it will be possible to compute expected values of the various outcomes and hence to evaluate alternative strategies. For example, in Figure 5.7, research may have indicated that trade customers outnumber private customers by 5:1 and that 90% of trade customers are less than five years' standing. By extrapolating this information with the total number of customers, it will be possible to evaluate the amount of total credit made available to customers. A description of this technique incorporating probabilities is given in Crowe & Avison (1980).

5.2.7 Decision tables

These are less graphical, when compared to decision trees, but are concise and have an in-built verification mechanism so that it is possible to check that all the conditions have been catered for. Again, conditions and actions are analysed from the problem situation. Decision tables can be used as computer input, programs being produced directly from them, and there are a number of packages available for this purpose.

When constructing a decision table, the various actions to be executed are listed in the bottom left hand part of the table known as the action stub. In the top left hand part, the conditions that can arise are listed in the condition stub. Each condition is expressed as a question to which the answer will be 'yes' or 'no'. All the possible combinations of yes and no responses can be recorded in the upper right hand part of the table. Each possible combination of responses is known as a rule. In the corresponding parts of the lower right hand quadrant, an X is placed for each action to be taken, depending on the rule of that column.

Figure 5.8 shows the decisions relating to the treatment of customers with bad debts with a company. The condition stub (upper left section) has all the possible conditions. Condition Entries (upper right quadrant) are either Y for yes (this condition is satisfied) or N for no (this condition is not satisfied). Having three conditions, there will be 2 to the power of 3 (2 x 2 x 2 = 8) columns. The easiest way of proceeding is to have the first row in the condition entry as YYYYNNNN, the second row as YYNNYYNN and the final row as YNYNYNYN. If there were four conditions, we would start with eight Ys and eight Ns and so on, giving a total of 2^4 (2 x 2 x 2 x 2 = 16) columns.

All the possible actions, are listed in a concise narrative form in the Action Stub (bottom left). An X placed on a row/column coincidence in the Action Entry means that the action in the condition stub should be taken. A blank will mean that the action

should not be taken.

Once the table is completed, rules which result in the same actions can be joined together and represented by dashes, that is, 'it does not matter'. The result of this is a consolidated decision table. Figures 5.8 and 5.9 illustrate an example decision table before and after the consolidation process. In the decision table (Figure 5.9), the third and fourth rule of the decision table seen in Figure 5.8 have been merged into a consolidated rule 3, expressing a 'don't care' condition. This is because the same process needs to be executed whether or not condition 2 is 'yes' or 'no'.

			√	√				
INVOICE > £300	Y	Y	Y	Y	N	N	N	N
ACCOUNT O'DUE > 3 MONTHS	Y	Y	N	N	Y	Y	N	N
NEW CUSTOMER	Y	N	Y	N	Y	N	Y	N
INFORM SOLICITOR	X		X	X	X			X
WRITE 1ST REMINDER LETTER		X	X	X	X			X
WRITE 2nd REMINDER LETTER					X			X
CANCEL CREDIT LIMIT			X	X		X	X	

Fig. 5.8. Example decision table

			√				
INVOICE > £300	Y	Y	Y	N	N	N	N
ACCOUNT O'DUE > 3 MONTHS	Y	Y	N	Y	Y	N	N
NEW CUSTOMER	Y	N	—	Y	N	Y	N
INFORM SOLICITOR	X		X	X			X
WRITE 1ST REMINDER LETTER		X	X	X			X
WRITE 2nd REMINDER LETTER				X			X
CANCEL CREDIT LIMIT			X		X	X	

Fig. 5.9. Example consolidated decision table

In systems analysis, there are likely to be requirements to specify actions where there are a large number of conditions. A set of decision tables is appropriate here. The first will have actions such as 'GO TO Decision Table 2' or 'PERFORM Decision Table 2'. Each of these may themselves be reduced to a further level of decision tables. The technique therefore lends itself to functional decomposition.

Sometimes the values of conditions are not restricted to 'yes' or 'no', as defined in the limited entry tables described. There can be more than two possible entries, and extended entry tables are appropriate. For example, the credit allowable to a customer could vary according to whether the customer had been dealing with the firm for 'up to 5 years', 'over 5 and up to 10 years', 'over 10 and up to 15 years', and so on. The rule for obtaining the right number of combinations will need to be modified. If condition 1 has two possibilities and condition 2 five possibilities, then the number of columns will be 2 x 5, that is, ten columns.

Whereas decision trees are particularly appropriate where the number of actions is small (although it is possible to have large decision trees), decision tables are more appropriate where there is a large number of actions as they can be decomposed into sets conveniently, in other words decision tables can better handle complexity. However, decision trees are easy to construct and give an easily assimilated graphical account of the decision structure. Decision tables have good validation procedures and, further, can be used to generate computer programs which carry out the actions according to the rules. Here the processing is specified by the analyst in terms of decision tables. These are transferred to computer readable format, and the programs generated automatically.

5.3 IMPROVEMENTS AND ADDITIONS TO TOOLS

In the previous section, additions and improvements to the descriptions of techniques used in Multiview have been given. In this section, improvements to the tools are described. Many of these have become much more suitable as part of a methodology since 1985, particularly fourth generation systems and application workbenches. Others, such as database management systems, data dictionaries and project control tools were available in 1985 but not described in the original exposition of Multiview. They will be available to any user of Multiview and the analyst chooses from this 'tool kit' as appropriate to the problem situation. Many of the considerations that apply for choosing an application package (Section 5.4.1) also apply when choosing a software tool. Quang & Chartier-Kastler (1991) suggest fifty criteria when choosing a software tool supporting the systems development methodology, Merise, and use these to assess ten design support tools for Merise.

5.3.1 Database Management Systems

In the original exposition of Multiview it is stated that:

> 'An information system is built around a database of some sort. [The term database] for the user ... just means a collection of information, and Multiview uses the term in this more general sense'.

A database and associated database management system are a major feature of many information systems, and therefore more information should be provided in the exposition of Multiview relating to this tool. This loose description of a database given above is no longer appropriate.

A database is an organised and integrated collection of data. It will be used by a number of users in a number of ways. In some methodologies the whole organisation (or that part which is of interest in the problem situation) is modelled on a database. The data analysis techniques described in Multiview can be used as a precursor to creating the database. A large part of information systems development concerns the best way of using the data resource as a whole, as against designing, creating, validating and updating files especially for a particular application.

The software to manage the database is the database management system (DBMS). This will store the data and the data relationships on the backing storage devices. It must also provide an effective means of retrieval of that data when the application systems require it, so that this important resource of the business, the data resource, is used effectively. Efficient data retrieval may be accomplished by computer programs written in conventional programming languages accessing the database. It can also be accomplished through the use of a query language (Section 5.3.2) which is designed for use by people who are not computer experts.

Avison (1985) details the reasons for adopting the database approach as a basis for information systems development:

- Reduce data duplication.
- Increase data integrity.
- Increase speed of implementing systems.
- Ease file access.
- Increase data independence.
- Provide a management view.
- Improve standards.

DBMS are software packages which manage complex file structures. DBMS make databases available to a variety of users and the sharing of data can reduce the average cost of data access as well as avoiding duplicate and therefore possibly inconsistent or

irreconcilable data. Databases hold large amounts of data and operations required to use the database are complex. Correspondingly, DBMS are large, complex pieces of software. Users of databases do not directly access the database. Instead they access the DBMS which interprets the data requirements into accesses to the database, makes the accesses required, and returns the results to the user in the form that the user requires.

The various users accessing the database via the DBMS may be user department managers and clerks as well as data processing professionals and they may well have different views of the database. They may access the database using 'user friendly' query languages on their workstations. Access may also be made via user programs written in a conventional programming language such as Cobol or PL/1 which act as host languages. Many analysts will set up a database as part of the information system. It is therefore appropriate to include a description as part of an information systems development methodology.

Many DBMS provide report generators as an alternative way of using the database. Report generators are usually designed for unskilled users. They are orientated towards providing large volume printed reports rather than displays on the screen although many systems will also display the same report on the screen. The report format should be easy to design. Sometimes the DBMS will also provide program and system generation facilities. These can be used to build more complex systems, containing significant processing facilities. These features are normally associated with fourth generation systems.

An important human intermediary will be the database administrator (DBA) who will be responsible for the design of the overall data structure and for ensuring that the required levels of privacy, security and integrity of the database are maintained. The DBA, or more accurately the DBA team, could be said to be managers of the database and, because the design of the database involves trade-offs, they will have to balance these conflicting requirements and make decisions on behalf of the whole organisation, rather than on any particular user or departmental objective. The DBA can give crucial help to systems analysts designing and developing information systems in a database environment, so that the information system is appropriate to all users. The role of the database administrator is described in Avison (1985).

When setting up the database, the data structures have to be mapped onto the DBMS. In a database environment, data structures may have been formulated using entity modelling techniques, and the resultant model mapped on to the database. However, DBMS differ in the way that they require the data structures to be formulated. Usually these structures will be hierarchical, network, or relational (Date, 1986).

The entity model can be mapped on to any of the above types of database model (Quang & Chartier-Kastler, 1991) and therefore proves a good base on which to develop an information system using a DBMS as a tool. Many methodologies assume that the applications will be implemented using a DBMS. Although those that incorporate entity modelling might claim that the technique is useful whatever tools are used for implementation, the applications are nevertheless usually implemented in this way. Indeed entity modelling gained prominence because it proved to be a very useful preliminary step to database implementation.

5.3.2 Query Languages

Query languages were originally designed to enable speedy and flexible access to information in a database. They were particularly aimed at non-professional users requiring ad-hoc access without the necessity of having a special program written for them. Originally query languages enabled only fairly simple queries to be asked but more and more complex searching, sorting and matching facilities are now provided. Further, it is quite common, given that the user has the correct authorisation, for query languages to facilitate updating of the database as well as the retrieval of information.

An example simple query might be as follows:

"LIST ALL students IN ASCENDING ALPHABETICAL ORDER OF name
THAT HAVE AVERAGE OF grade1, grade2, grade3 GREATER THAN 65%".

This may look very free format and natural language like, but in fact there is a fairly fixed syntax and only a certain number of key words can be used. The words of this particular hypothetical query language are in capitals and the words in lower case refer to elements in the database. This query accesses the student records file, calculates the average of the specified fields (grade1, grade2 and grade3) for each student and lists the students records that satisfy the condition (average greater than 65%) in the specified sequence (ascending alphabetic order of name).

This is a simple query in that it only requires access to one file (the student file). More complex queries would require access to two or more files or tables in the database.

Query languages are, in general, non-procedural which means that the users specify what they require from the system rather than how to get it. Most traditional programming languages are procedural and the user has to specify how to achieve what is required, that is, the procedure. Obviously non-procedural languages are easier for non-specialists to use.

Query languages are designed to:

- Provide ad-hoc and on-line access (and update) to information in the database.
- Provide an English language-like, non-procedural interface to enable flexible access.
- Handle only relatively small data volumes.

Query languages for non-relational databases differ considerably one from another and are specific to the particular DBMS that they are using. In the case of relational databases the situation is somewhat different. The basic query language of relational DBMS is the relational algebra (Date 1986). This consists of a basic set of primitives for manipulating relations. It is in fact a procedural language and not particularly easy for non-specialists to use, that is, it does not really meet the criteria required of a query language. For this reason a more fitting query language is often employed that is easier to use and non-procedural, known as the relational calculus. The relational calculus is converted into the relational algebra primitives prior to execution on the relational database.

Relational calculus-based languages differ one from another although there are some common characteristics. So once again there is no standard query language defined for use on all relational databases, although many people now believe that SQL is beginning to emerge as a de-facto standard now that IBM's DB2, their relational database product, uses it.

When compared to the relational algebra, the relational calculus is more orientated towards expressing the user requests in a way that the user may construct, because it is non-procedural. This means that the user specifies his needs rather than having to construct the procedures to retrieve the required data. 'Pure' relational calculus is difficult for non-mathematicians, but more user-orientated languages which are based on the relational calculus are easier to use.

SQL, is a calculus-based language. The main operator has the following basic structure:

SELECT/FROM/WHERE

so that a request:

LIST DETAILS OF EMPLOYER NUMBER 756

would be formulated as:

```
SELECT   EMP-NO, NAME, DEPT, SALARY  (data items - a 'target list')
FROM     EMPFILE                     (a relation involved in the query)
WHERE    EMP-NO=756.                 (qualifier or 'condition')
```

The qualifications on the data retrieved can include ANDs, ORs and NOTs, along with < (less than), > (greater than), and so on. The results can be ORDERED according to users' requirements and operations such as average, minimum, and maximum vales calculated. The user has to know the names of the relevant attributes and key them in explicitly. Alternatively, the sets of commands can be 'programmed' and stored when developing the information system and called in as required when the system is operational. SQL also provides a CREATE VIEW command which sets up alternative views of the data derived from other tables and selected rows and columns. This is useful when setting up different sub-schemas (which adhere to the security and privacy requirements defined by the database administrator), that is apply to different users within the same information system. A full account of SQL is found in Date (1984).

Query By Example (QBE), which was proposed by Zloof (1978), is designed for users with little or no programming experience. This cannot be said of the relational algebra and it is more 'user friendly' than a straight relational calculus language such as SQL. The intention is that operations in the QBE language are analogous to the way people would naturally use tables. Further, as the name implies, the user can use an example to specify his query. The user enters the name of the table and the system supplies the attribute names in a skeleton table. The system details the attributes in which the user has indicated an interest (see Figure 5.10).

Empl. No.	Name	Status	Birthday	Tax to date	Salary

(a) Skeleton table for QBE request

Empl. No.	Name	Status	Birthday	Tax to date	Salary
P.756	P	P			P

(b) Table completed by user:
756 - record to be addressed
P - details requested

Empl. No.	Name	Status	Birthday	Tax to date	Salary
756	SMITH	FULL			£12,030

(c) Table completed as requested

Fig. 5.16. QBE example

Many systems are either menu-based or form-based and these prevent even the necessity of recalling the relation names. This means that minimal training is necessary and therefore is usefully associated with information systems development by users or with user participation. However there are severe limitations to retain flexibility unless the database is not very complex. Some aspects of a natural language interface, which are still in a somewhat experimental phase, are discussed in Codd (1974).

With database management systems and query languages it becomes easier to translate user requirements into working systems in an information systems methodology. A third piece in this jigsaw of tools in a database environment is the data dictionary which is discussed in the next section. These three tools will also be part of a fourth generation system (Section 5.3.4), designed specifically for applications development.

5.3.3 Data Dictionaries

Data dictionaries were not included in the original exposition of Multiview (nor indeed in the version propounded in Avison & Wood-Harper, 1990) yet are a very useful interface with database management systems, and the database maintenance, control, and the recovery and monitoring aspects of the technical solution. There has recently been a significant increase in the use of data dictionary systems (DDS). This is mainly due to the corresponding growth in the use of entity analysis techniques, DBMS and fourth generation products. A DDS can ensure standard and consistent definitions for all data in the organisation, essential if the output from entity analysis is not to cause confusion, as a DDS helps to control and use the information so gathered. Many DDS are integrated with a particular DBMS or offered as a possible 'extra' to a DBMS.

A DDS is a software tool for recording and processing data about the data (meta data) that an organisation uses and processes. Originally DDS were designed as documentation tools, ensuring standard terminology for data items (and sometimes programs) and providing a cross-reference capability. They have evolved as an essential feature of the information systems development environment (Curtice & Dieckman, 1981).

A DDS is a central catalogue of the definitions and usage of the data within an organisation. Information systems development is helped by the clear definition of what data is already in the database. The DDS can be accessed by each new information system as it uses the database and therefore the DDS eases the sharing of data. If used alongside a DBMS it could be said to be a directory of the database, 'a database of the database', although this relationship might be better expressed as an 'information source about the database'. Thus DDS are used chiefly as documentation aids and as control points for referencing data. A DDS will hold definitions of all data items, which

may be any objects of interest, and their characteristics. It will hold information on how this data is used as well as how it is stored.

A DDS can also be used as a storage base of logical file designs and programming code (sub-programs) and these sub-programs may be used in a number of application programs. Many DDS are therefore more than mere points of reference about data, and they hold information about processes as well as data.

The DDS pervades all aspects of information systems work, and its use could lead to improved documentation and control, consistency in data use, easier entity analysis, reduced data redundancy, simpler programming, the enforcement of standards and a better means of estimating the effect of change. Of course some of these advantages could be equally said to be the result of using a database with a DBMS and an effective database administration team. Indeed, they all contribute to an effective data processing environment. They are as useful to the systems analyst developing the information system using a methodology (such as Multiview) as they are to the user requiring the information provided by the system so developed.

The conceptual and implementation views for data and processes are shown in Figure 5.11. The four quadrants of the figure represent the components of the DDS. There is a fifth component which indicates the cross-referencing between components. It will be possible to identify entities, functions, and programs using a simple referencing system which will be an effective tool in maintenance, as well as help when designing the information systems. An example of its use might be in answering the question, 'What processes use this data item?'.

	DATA	PROCESSES
CONCEPTUAL	entities attributes relationships	events operations
IMPLEMENTATION	records files data items sets areas	subroutines modules programs systems

Fig. 5.11. The four quadrants of a DDS

For each data element, the DDS will contain amongst other things, the following:

- The names associated with that element.
- A description of the data element in natural language.

- Details of ownership.
- Details of the users that refer to the element.
- Details of the systems and programs which refer to or update the element.
- Details on any privacy constraints that should be associated with the item.
- Details about the data element in data processing systems and programs.
- The security level attached to the element in order to restrict access to it.
- The storage requirement.
- The validation rules for each element.
- Details of the relationship of the data items to others.

With so much detail needed to be held on the data dictionary, it is essential that a referencing and cross-referencing facility is provided by the DDS. This will help the systems analyst in the design process and it provides a check for completeness.

Most commercially available DDS now go beyond these basic facilities and the DDS can be seen as a tool with a number of objectives:

1. To provide facilities for documenting information collected during all stages in the development of an information systems project.
2. To provide details of applications usage and their data usage once an information system has been implemented, so that analysis and redesign may be facilitated as the environment changes.
3. To make such access easier by providing cross-referencing and indexing facilities.
4. To make extension of the DDS easy.
5. To encourage systems analysts to follow structured methodologies.

This last objective could point, for example, to data dictionaries holding details of the contents of the sources and destinations (sinks) of data flows, outlined in a data flow diagram (Section 5.2.1). This shows the importance of data dictionaries as tools of process definition as well as data definition.

The scope of a DDS is therefore broad. It is not merely a documenting aid: it is a support to all stages in the development and maintenance of an information system and the control of the database. A more sophisticated DDS would have a number of extra facilities such as the following:

1. Automatic input from source code of data definitions.
2. The generation of source code relating to items in the data dictionary to a sort program or applications program.
3. The recognition that several versions of the same programs or data structures may exist at the same time.
4. The provision of on-line facilities to aid the interrogation of the database.
5. The provision of an interface with a DBMS.
6. The provision of security features, such as password systems.

7. The provision of facilities to generate application programs and produce reports and validation routines and the provision of test file generation facilities.
8. The provision of data definitions and sub-routines for application programs enforce some standards on programming, making programs more readable and consistent.
9. The provision of an estimate of costs of any proposed change of use of the data and an estimate of the time scale for making such changes.

Thus, DDS can be used as a dynamic tool of the system development process. The DDS has also become the basis of many analyst and programmer development workbenches. Here, the DDS has had a number of analysis and design functions added to it, and in some cases the addition also of implementation tools. This concept of a DDS has come a long way from its origins as a simple store of data about the data in the database, and, as an indication of this, the term System Encyclopedia is often now used instead of DDS.

5.3.4 Fourth Generation Tools and Methodology Workbenches

There is no generally agreed definition on what constitutes a fourth generation system. It is often argued that development work should be carried out by people who are not programmers, and this, perhaps, lies at the heart of the tools which are usually together known as fourth generation systems. But they can be as useful to professional computer people developing information systems. Indeed, Martin (1982a, 1982b and 1983) distinguishes between end user and professional fourth generation systems.

The basic aim of fourth generation systems is to speed up the work of developing new systems and hence reduce the applications backlog. A further benefit accrues to applications developed using fourth generation systems, for they are not just developed more quickly but, when they require changing, as they inevitably will, the changes can be made speedily and accurately.

In fourth generation systems, the program flow (procedures) are not designed by the programmer but by the fourth generation software itself, in the form of previously constructed, parameterised algorithms. Each user request is one for a result rather than a procedure to obtain this result. Compared to third generation languages, they require less lines of coding, are quicker to write and test and easier to maintain. They are designed for interactive use and the dialogue between user and software enables errors to be corrected as the application is being developed. Mnemonics, which require learning and remembering, are replaced by menus, semi-natural language, and other easy-to-use facilities.

Most fourth generation systems use a mixture of graphics and text which eases the specification of user requirements. Some use a technique whereby the user 'fills in the blanks' and rapid development is further helped through the sensible setting of default

options, which are followed if the user does not specify otherwise. 'Intelligent' defaults, might include, for example, the place and format of the date on a report. If the users specify and implement the fourth generation solution, they can also maintain it without involving the data processing professionals and thereby reduce the large maintenance costs of traditional computing.

The tools that make up a fourth generation system will normally have a common user interface, very often the WIMP computer-user interface.

One obvious advantage of the users developing their own systems is that it avoids some of the communication problems that arose in the past between users and developers. A major criticism that users make of data processing professionals is that they have little understanding of the business and do not pay enough attention to users' requirements. Also, if the development is speeded up, the likelihood of the user requirements changing during the development process is reduced. Further, programming and systems analysis expertise is a scarce resource in most organisations and due to the demand there is frequently a long lead time before projects are implemented.

There are a number of characteristics of a system orientated towards the user to which fourth generation systems would be expected to conform (see also Nelson, 1985):

- They should be easy to learn and use effectively.
- As well as good written documentation, there should be on-line 'help' facilities and debugging aids.
- They should be available for use interactively on a terminal connected to a mainframe computer or using a microcomputer.
- They should be robust, so that they do not 'crash'. Such systems should be designed to be 'tolerant' of mistakes in data entry.
- They should self-document any work produced, so that the operational systems are easy to maintain.

As well as producing final applications, fourth generation systems can also be used to generate prototypes (Section 2.6). Performance criteria are less consequential when developing a prototype, which can be looked on as a working model, a means of refining the exact form of a working model, of the operational system. The system may produce skeletal programs and the user can experiment with various options. Principles and user requirements can be tested for viability so that costly mistakes in solving a complex data processing problem can be avoided. Fourth generation systems can produce fairly cheap and effective prototypes.

A fourth generation system is likely to have the following modules (described in Avison & Fitzgerald, 1988a):

- Database Management System.
- Data Dictionary.
- Query Language.
- Report Generator.
- Screen Generator.
- Program Generator.

Some fourth generation systems also have:

- Business Graphics.
- Spreadsheet.
- Word processing
- Mailing.
- Investment analysis.

Indeed, a fourth generation system is likely to give the user a comprehensive 'tool kit'.

In view of the claims made by vendors of fourth generation systems regarding their benefits, it is interesting to ask why they have not currently made a great impact on reducing the application backlog (Jeffery & Low, 1989). Possible reasons include the following:

- They are comparatively new and still in the early development stage.
- They require a considerable change in the systems development culture of an organisation, and not all organisations are yet ready to make this change.
- The type of application that successfully used fourth generation system developments has been limited to fairly small, one-off type developments.
- Fourth generation systems are often less efficient in operation than code produced by third generation programming, but any inefficiency, in terms of excess running time and memory required, must be traded off against increased productivity.

Thus, most large transaction processing systems are still developed in traditional ways. Nevertheless the impact of fourth generation systems will increase, but whether they are ever used for all developments seems unlikely. Indeed, it is somewhat an irrelevant debate because as long as a significant number of developments are undertaken using fourth generation systems, then most objectives will have been achieved. Surveys of fourth generation systems and methodology workbenches can be found in Horowitz (1985), Lobell (1983), Tozer (1984) and Pergamon (1987).

Whereas fourth generation systems are tools for general use on a number of methodologies, methodology workbenches are designed to support a particular methodology or methodologies. There is no workbench designed for Multiview, though there are some which would help the user of Multiview.

Methodology workbenches are sometimes separated into two parts and termed

analyst workbenches or programmer workbenches. There is no complete agreement on terminology and the term CASE (Computer Aided Software Engineering) is quite common for what we term programmer workbenches, as is IPSE (Integrated Project (or Programming) Support Environment).

Programmer workbenches support the design and construction of programs. The objective is to provide an environment in which software can be speedily developed which adheres to the standards of structure and the principles of software engineering. Within this framework there are usually two distinct support features: code generation and sub-program provision. A code generation facility generates source code in a traditional third generation language from a high level specification language, such as structured English. The second support feature is the provision of building blocks of code that have been pre-written for functions and processes which are frequently required and can be incorporated into a final application.

A workbench is likely to include test tools and diagnostic aids. A subroutine or program can be tested independently of other subroutines or programs. These facilities might be used to trace the execution of the subroutines or programs so that intermediate results can be validated. For this reason, the workbench can also be used to maintain programs and systems as well as creating new ones. Workbenches aim to attack the applications backlog by reducing maintenance time as well as development time.

Analyst workbenches support the systems analysis and design tasks. In particular they usually support the logical data modelling processes by automating the construction and maintenance of entity models, and their conversion to a variety of target DBMS schema designs (including normalised data). On the process modelling side they usually support the construction and maintenance of DFDs and function charts. These facilities would help the user following Multiview. There may well also be support for screen and report designs and a menu and dialogue design facility. Analyst workbenches will have some kind of data dictionary or system encyclopedia as the central core of the facility. (A review of a number of analyst workbenches is found in Pergamon, 1987.)

Inevitably analyst and programmer workbenches have been combined to form integrated project workbenches which addresses the whole of the development of a project from analysis through to construction, using a common set of tools and user interfaces.

An example of such a workbench product is the Information Engineering Workbench (IEW) jointly developed by Arthur Young and Knowledgeware. This is designed to support the Information Engineering methodology and so is properly classified as a methodology workbench. The core of the IEW is the Encyclopedia, which contains information about all the objects in the development project, be these

entities, processes, relationships, attributes, or whatever. The Encyclopedia is in two parts, one dealing with the organisation's information and data, and the other with the organisation's functions or activities. Information is also kept that relates one part to the other, for example, what data are used by what function.

At the user end, IEW provides access to a series of diagraming support tools similar to those described above. These are for DFDs, Entity Models, Decomposition Diagrams (which are kinds of structure charts) and Action Diagrams, described in Avison & Fitzgerald (1988a).

Sitting between these diagraming support tools and the Encyclopedia is something called the Knowledge Co-ordinator. This is an 'expert system' containing the rules of the Information Engineering Methodology. It enforces the correct standards for completeness, consistency, and accuracy of the diagram content.

5.3.5 Project Management Tools

One of the most important tasks of a manager of an information systems project is project planning and control. Projects which take longer than scheduled cause a loss of money as well as embarrassment, particularly if they are unexpected or cannot be explained easily. Most methodologies split the information systems task into subdivisions and to activities within these. This ensures that there are a series of checkpoints that can be readily identified so that systems development staff can work to these and hence control the project. These checkpoints can be used to provide the interface to a project control tool. The use of a project planning package can ensure that projects are scheduled at the earliest possible date, with the least drain on resources, and that there is a good chance that this date will be met. If there are delays, then at least the managers have information about them. Project management should pervade the information systems development project. Although control was discussed in the original exposition of Multiview (Wood-Harper et al, 1985) tools were not included in this discussion. Project management is also discussed in Avison (1990b).

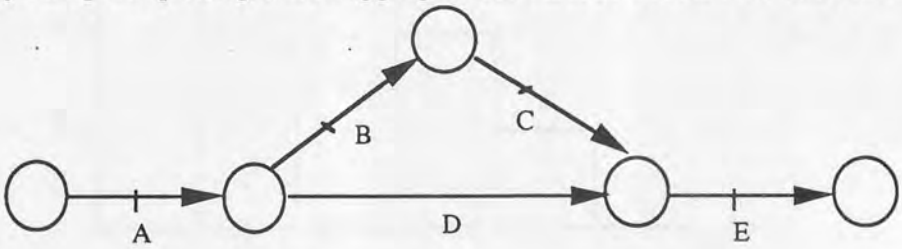
Project control techniques start with an attempt to break down the large and complex project into tasks, normally called activities. Once the activities have been identified, a time and resource requirement is assigned to each of these, and the inter-relationships between them established. In other words, those tasks which are dependent on the completion of other tasks are identified. These activities, and information about them, can be entered into a project control package. A project control tool may also be a part of the tool kit included in a fourth generation system, or more likely, an analyst workbench.

Using this information, the computer package can draw up a network. In a network, the activities are represented by arrows which join the nodes. These represent

events, that is, the completion of activities. Figure 5.12 shows a network. The arrows represent the activities, though the length of the arrow does not indicate the time taken for each task. Arrows drawn in parallel indicate tasks that can be carried out simultaneously. Arrows following others indicate tasks that are dependent on the completion of those other tasks.

The manual development of networks is lengthy and project control packages can make the task much easier. They can draw the network and highlight critical activities on which any slippage of time will cause the whole project time scale to suffer. The path of the critical activities joined together forms the critical path, and it is useful for the package to highlight these activities. In Figure 5.12, the activities A-B-C-E are on the critical path. If it is possible to reduce the time of these activities, possibly by moving resources allocated from other activities to them, then the overall project time should decrease.

Many computer packages will aggregate the various resources, such as the number



Activities on critical path

Activity	Dependence	Duration	Start	Finish	Slack
A	—	10 days	—	10	—
B	A	6 days	11	16	—
C	B	16 days	17	32	—
D	A	8 days	11	18	14
E	C,D	6 days	33	38	—

Fig. 5.12. Project control - network and critical path

of people working on the activity, and to attempt to level the use of these resources throughout the project. It is usually better to use resources as smoothly as possible in the lifetime of the project. Once this has been done, a bar chart showing the resource allocation over time can be displayed and printed (see Figure 5.13).

Many packages will report on inconsistencies within the network, such as the same

resource being used at the same time. The package will normally convert days into calendar dates and to allow for weekends, bank holidays, and other holidays.

Normally there is a trade-off between time and cost, in other words the more resources allocated (and the more costly the project), the quicker it can be finished. The user may like to input the minimum, the most likely, and the maximum resource availability so as to get three different results for time/cost comparisons. Such an exercise would be very laborious if drawn by hand.

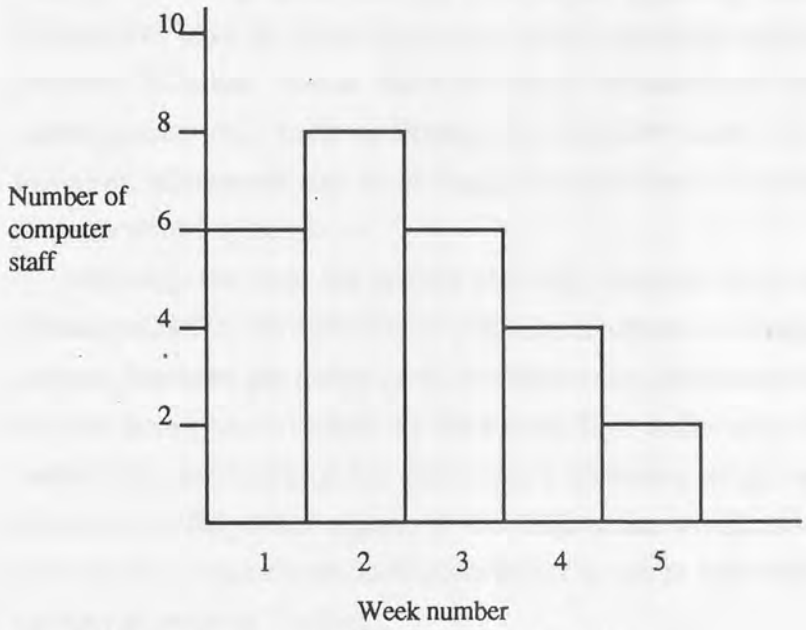


Fig. 5.13. Project control - resource schedule

Project planning packages may permit the user to ask 'what if?' questions so as to see the consequence of certain actions, for example of a re-allocation of staff, a holiday period, and a machine breakdown, and so on. The user can also use the package to highlight the results of following different work patterns or changing other aspects of the business. Useful reports from the package include a list of activities presented in order of latest starting date, earliest starting date, by department, by resource and by responsibility.

Once the project has started, there will be progress reporting. This can be used to:

- Compare the schedule with progress made.
- Detect problem areas.
- Provide a historical record which can be used for future project planning.

The system may also give information about how to act so as to put right any deviation from the schedule. This may be achieved by, for example, increasing the resources on some activities and re-scheduling others. This goal-seeking type of analysis, almost impossible manually, is achieved by a number of computer packages.

Any change in a manual system will require the chart to be re-drawn - an inconvenience at best; more often a task left undone, which makes it likely that any previous planning is ineffective.

The development of an information system is likely to be a large scale project, particularly where the overall plan has a number of inter-related sub-systems that need to be completed separately but integrated later. 'Large' here refers to time, cost, people, and equipment resources, as well as to the complexity of the inter-relationships between activities in the project. The use of a project planning tool makes an up-to-date and informative plan far more likely, and such large scale integrated information systems projects feasible. Some methodologies recommend using a particular project management tool, such as Prompt for SSADM users. To set against these gains, however, allowance has to be made for the costs of maintaining and updating the project control systems.

Although the need for project planning tools has been stressed here, according to Grindley (1987), the difficulty of meeting deadlines on computer systems development projects has been the major problem concerning information technology managers over the last seven years (when the data were first collected). Although project planning tools help, particularly by enforcing a planning stage, and thereby enabling the discovery of important aspects of the project that would otherwise go unnoticed (such as conflicting objectives), their main effect seems to have been in measuring our lack of success at meeting deadlines.

One would expect that the use of integrated tools or a fourth generation system would lead to workable systems being produced more quickly. According to Grindley's statistics, however, the problem is worsening. Demand is outstripping the supply of improved information systems. Perhaps the problem is insoluble, the supply of information systems simply feeds further demand. The faster systems are developed using fourth generation tools and others, the faster users will expect them, and want new ones. No project control package can help in a fundamental way to solve this problem. The answer might be to let users develop their own systems and let *them* make the choices. They can then see the repercussions of these decisions. This might make for a more realistic framework for information systems development.

5.4 IMPORTANT CONSIDERATIONS PREVIOUSLY OMITTED

It is claimed that Multiview is suitable for small companies and microcomputer use, and yet the first exposition of the methodology gave little help the process of evaluating and selecting application packages and no guidance on end-user computing. Wiser choices on application package selection might have lessened some of the interfacing problems

experienced in the DLU case. End user development was not envisaged in the first edition of Multiview (although user involvement and participation in general was stressed). With new tools, for example, fourth generation systems (described earlier in Section 5.3.4) user development becomes a real possibility. This is also discussed in this section. Although evaluation of the final system was discussed in the early exposition of Multiview, there was very little advice given on how to assess user reaction. Techniques for this are also suggested in this section. A further omission in the original exposition of Multiview was documentation and documentation standards. Some suggestions are discussed in the final part of this section.

5.4.1 Application Package Selection

In the last few years, partly due to the impact of microcomputers, package solutions have been advocated as a cost-effective way of developing information systems. However, most methodologies still assume that they will be developed in-house and tailor-made. Systems development methodologies should address the problem of package evaluation and selection (Avison & Fitzgerald, 1988b).

An overview analysis of the application area may have concluded that the application is of a certain general type, such as stock control or sales ledger, and that the volumes of data and transactions were within the range likely to be catered for by an off-the-shelf package. Such a solution seems to be a straightforward matter, but there are a number of pitfalls.

In order to select suitable software, the requirements need to be specified in more than outline fashion, and some of these requirements may relate to technical matters, for example the way in which data should be transferred from one program to another. There is no standard way of describing software so that the buyer of a package knows exactly what he is getting. The manual provided with the system may help, and it may be possible to buy a copy of the manual before finalising any decision about purchase. But even the manual may be vague and ambiguous about certain points or else optimistic about the capabilities of the system. Some 'capabilities' may refer to aspects that have not been implemented as yet.

Difficulties in evaluating an application package may be caused by not knowing what questions to ask of it. A very common reaction of users to packaged software is to go through the following stages:

- Fantasy about what problems a computer system would banish for them.
- Anxiety at the thought of coping with a new system, because they feel as though they are on alien territory.
- Delight as they start to get the basics working and discover that they understand quite a lot about it.

- Horror when they discover that the package cannot do some of the things that they need for the system to be effective.

The advantage of having been through the Multiview analysis stage is that there is a series of models of how the system needs to work in practice. The salesman and the manual can be probed to ensure that the package will be able to cope with the situations that have been written into the model. Thus, the package must be able to deal with the information processing associated with all the events in the information model. The package should also be looked at in the light of the following questions: "what sort of organisation was this package intended to work in?" and "what sort of people are supposed to be able to operate it?". It may be possible to have the package free on loan for a trial period, so that it can be evaluated fully.

If there is doubt about what is wanted from a computer system, then it is obviously useful to look at what is available. A package that is potentially useful may be seen and even if such a package is not found, then more information about what might meet the needs of the organisation has been gained. It is a common observation that people are better at seeing what is wrong than what is right. If this is true then a reasonable specification may be "what I want is the features of the XYZ package, plus ....., but excluding .....".

It is always worth looking at what packages are available as it is possible to learn about particular applications and the ways in which they are usually handled. This will be useful even if a solution will eventually be tailor-made. Useful features, that were not originally thought of, may be used. These features could include good ways of presenting menus, layouts, or error handling routines. Analysts with a strong computer background often over-estimate the user's knowledge of what computers are and what they can do. Demonstrations of packages might be very helpful to the users so that they can see what is available and for the analyst to gauge the capabilities of the users.

When choosing a package, there are a number of questions that should be asked. The first of the following is the most obvious (but even this is sometimes ill-considered), and there are other important criteria that ought to be followed (Avison, 1990b):

- Does it meet the functional requirements?

  This is of course the issue of fundamental importance. If the package does meet the functional requirements then:

  (a) Is all the input required by the package readily available?

  (b) Is its capacity large enough for present use or too restricting for the future?

  (c) Does it process the data fast enough?

  If only some of the requirements are fulfilled then some other questions should be asked:

(a) What proportion of the requirements are fulfilled without amending the application package?
(b) Are the limitations of the package acceptable?
(c) How easily can the extra requirements be fulfilled?

- What resources are required to buy and run the package?
  (a) What is the basic cost, maintenance cost and the cost of extra hardware and support required?
  (b) What labour is required to set up and run the system?
  (c) Can the package be run on other computers (which may be important later)?
- How many people are presently using the package?
  (a) Is it possible to get their reactions to it?
  (b) Were there many setting-up and teething problems?
  (c) Has it proved reliable?
  (d) Are they presently happy with the system?
  (e) Are they happy with the help provided by the supplier when requested?
  (f) What would they have done differently now?
- What is the quality of the documentation? Is it geared to computer experts, or are the users of the system likely to understand it? Is it well written? The documentation should be judged on its appropriateness for the people who are going to use the package. Is it good for:
  (a) Reading (for example, to give a general overview of the system).
  (b) Learning (for example, so that the user can use the package without too much trouble).
  (c) Teaching (for example, so that the trained user can teach others how to use the system).
  (d) Referring to (for example, so that the format of a particular command can be checked).
  (e) Reminding (for example, to allow the user to look at an overview of the system quickly).
  (f) Diagnosing problems (for example, so that the user can soon correct any mistake).
- Are the 'help' facilities provided by the package when using the system on the computer good? If the user does make an illegal command or response, are the messages sensible and does the system provide help to make the correction? This is a facility which is often overlooked when buying a package but a good help facility is vitally important when learning about the system.
- Is there a disk tutorial system? Such a system should lead the user through the use of the package, with carefully chosen examples, in a structured way.

- Can the package be implemented without the need to employ computer professionals? This should avoid expensive setting up costs.
- Are other training facilities provided? These may be in-house or provided at the supplier's base.
- Is it a well-established product? Products that have been on the market for some time are likely to have fewer problems.

One possible way of evaluating packages is to develop a weighted matrix (see Figure 5.14) which shows how packages measure up to requirements (weighted to ensure that the most important requirements are given more importance). In the example, package C is the best option.

CRITERION/ (WEIGHT)	PACKAGE A	B	C	
1 (2)	6 (12)	9 (18)	8 (16)	
2 (10)	6 (60)	0 (0)	9 (90)	
3 (10)	7 (70)	0 (0)	9 (90)	
4 (1)	7 (7)	10 (10)	0 (0)	
5 (2)	0 (0)	10 (20)	0 (0)	
TOTAL WEIGHTED MARK	149	48	196	

Fig. 5.14. Comparing application packages

5.4.2 User Development

In the original exposition of Multiview, it was assumed that the computer solution could be carried out by a technically-oriented designer or programmer (albeit with user involvement). However, this is not the only possibility. Even though users are unlikely to be computer experts and unwilling to spend the many hours necessary in designing, writing and testing programs in a computer programming language, this does not preclude user development. For example, they sometimes choose a package without help from systems analysts or any computer people. An application package may perform something like 80% or 90% of the requirements in a particular application, leaving 10% or 20% of the work undone. It may be that these tasks can be done manually or that a consultant can be hired to design and write the software necessary to

execute these tasks. Some computer packages provide another alternative: the ability to enable the users to write their own procedures. It is frequently not beyond an interested user to write the procedures necessary to fulfil the rest of the required tasks.

Packages adopt one of two approaches. In the first, the user writes procedures in the programming language used by the writers of the application package. But this is likely to be difficult for users. The second approach is to provide a 'very high level language' which is easy to use and which the package can execute. A specially designed set of procedures is usually much easier to learn and to use than a conventional programming language. These commands should give the user the flexibility necessary to allow the production of routines necessary to 'customise' the package so that it can meet all the user's requirements. Even so, it is probably necessary for the user to have attended a training course in the use of the package, usually lasting a few days, to gain a fair command of the system's capabilities.

However, some applications are unique to a company or at least are so different that it is impossible to find a suitable application package available. If there is no in-house software specialist, there is still an alternative: that of using an applications generator, user workbench or fourth generation system (Section 5.3.4). As we have seen, the facilities that they provide are likely to include:

- Prototyping.
- Screen formatting.
- Database handling.
- Test data generation.
- Report generation.
- Data dictionary handling.
- Documentation generation.

Many of the systems advertised under these banners are designed for applications programmers - aids to speeding up their work - particularly the more mundane work such as screen and report design. However there is a way that even specialist tools can be helpful, for users and programming people can sit together at a workstation and try out a prototype. The prototype can be modified a number of times, until a system is developed which is satisfactory to the user. This process of developing application systems has two main advantages: firstly, it is quick, and secondly, because the user is involved, it is likely to produce a system that meets the user's requirements.

This section will concentrate on the attributes of those systems designed especially for the user. There are packages with facilities similar to those described above but usable by the untrained person. These systems will therefore be:

- Easily learned.
- Require few statements to do a particular task.

- Adopt default settings should the user not specify certain details.
- Provide productivity and cost gains over programs written by programmers in conventional programming languages.

These facilities form a user workbench (as against the programmer workbench and analyst workbench). Many systems have only a few of the facilities discussed below, but it is likely that newer systems will contain most of these.

- A screen painter (or formatter). This enables the user to set up a screen layout for such requirements as menus and screen reports.
- Database handling. The user may not have to collect the data for the application, it may be already in the database. However, it is necessary to specify what data is required. The system may do this by asking the user a set of questions or via a soft copy form.
- Code generators. From the user requests, the system will generate code. This code may well be in a high level language but there is no guarantee that the code generated is efficient.
- Report generation. This may also be specified by completing a form giving details of data required, headings, titles, totals, page breaks and sort and print criteria.
- Data dictionary. This gives details of the data stored.
- Data entry into the database. The user should also be able to specify data validation procedures, so that data entering the database is valid.
- Menu generation. The ability to create menus for future users as well as reports and forms. The menu may consist of a series of options expressed in English phrases or icons.

Another principle of many of these systems is that of development by example. The user 'ticks' those items on the screen that are required. A report screen could also be presented to the user who specifies areas where more detail is required (a 'zoom in') and other areas where a summarised format would be more appropriate.

Although such systems may be used to develop applications, they may also document the systems produced, or at least ease its production. For example, initial 'skeletons' of document types can be provided. These will not only help and speed up the production of documentation, but also ensure that standard layouts are followed and that the documentation is complete. Indexes can also be built up to cross reference elements of the documentation and the system itself. This also means that documents can be retrieved conveniently and altered if necessary.

5.4.3. Testing User Reaction and Evaluation

One of the most critical issues in information systems management concerns how to assess the information systems in an organisation. Assessment or evaluation is an act of

placing a value on an information system. Evaluation can take place from a number of different points of view, for example, those of the owners of a system, the users, or the system designers.

The owners of a system are most likely to be interested in economic assessment criteria and less interested in technical or operational aspects. They will be concerned with effects at an institutional level such as the way the organisation does business and how interactions with other organisations occur. System users will be interested in operational aspects and perhaps impact on work quality at an operational level. System designers may be concerned with the impact of an information system on the computing resources of the organisation.

Assessment should take into account effectiveness, that is, the proportion by which the system meets organisational objectives and goals, and efficiency, that is, a measure of the 'mechanical' aspects of the system such as the accuracy, timeliness and speed of access to information.

Hawgood & Land (1986) describe three fundamental difficulties in assessing information systems. Firstly, effectiveness is a subjective concept and different people will have different views on the effectiveness of a system. Secondly, information systems are immensely diverse in the functions that they carry out and in the objectives that they seek to meet. Thirdly, an evaluation should really take place by comparing the running of the organisation with and without the aid of the computer-based information system. In reality, this comparison is not usually possible.

Despite these drawbacks there is an important need to quantify the benefits of an information system and perform evaluation. A multi-disciplinary approach is needed to do this because evaluation can take place from a technical, economic, organisational or societal perspective. Evaluation is usually confined to the economic and technical aspects of systems, with the justification of the project at the feasibility stage being the most common stage. As Blackler & Brown (1988) point out, post implementation studies are less common, and where they exist are usually used to monitor costs and performance. Concentration on the economic and technical aspects may cause important organisational and social factors to be omitted, such as job satisfaction. Possible areas for study include functionality, operational factors, the structure of the organisation, the infrastructure supporting the system, ergonomic considerations and the use of skills (Avison, Catchpole & Horton, 1989).

Possible benefits of evaluation include improved understanding of the system, knowledge to extend the facilities provided and improved communications between users and computer people (Ginzberg & Zmud, 1986 and Etzerodt & Madsen, 1988).

In the case described in Chapter 6, it was decided to evaluate the prototype by soliciting opinions from the system owners, who are senior managers in the

organisation, and system users, who operate, provide data or make use of information from the system. It is also possible for users to be owners of the system. System owners will evaluate the effectiveness of the system, and system users will evaluate the efficiency aspects. In this case we wished to take into account efficiency and effectiveness and also to review the training and design process. This is essentially a review of the participative and prototyping approach to information systems development that Multiview recommends and was used in this instance. Opinions about the system were solicited from the senior managers as well as user groups and individual users. The methods used to assess the system were questionnaires with follow-up interviews and group discussions.

Where possible, comparison of the prototype was made relative to the previous equivalent manual system which was replaced by the prototype, and both of these comparisons were related to theoretical objectives, goals and needs which were identified by the design group. Costs and benefits associated with the old and new systems were identified, and then qualitative and quantitative items were evaluated covering such issues as services to clients, value added features, changes in the decision making process, and so on.

The questionnaires devised for the case study in Chapter 6 have four sections: effectiveness (objectives fulfilled, problems addressed, needs addressed, costs and achieved benefits, impact on work and clients, and impact on decision-making and control); efficiency of the system (attributes of the information produced, aspects of using the information system, value placed on information generated, data collection methods, working practices of staff, and the personal aspirations of staff); the design process (in terms of perceived 'success' of the implementation, contribution made by 'participation' procedures, and personal benefits and gains of staff); and finally the training process. Four of the fourteen tables derived from responses from users in that case study are given in Chapter 6 (Figures 6.20 to 6.24).

5.4.4. Documentation

Information regarding some significant documentation was omitted in the first exposition of Multiview. For example, there was no help as to the design of forms on which to document entities, attributes, relationships, events and operations. The basic forms for these have been designed and three completed forms are shown in Chapter 6 (Figures 6.7 to 6.9). Furthermore, there was no general advice in the original presentation of Multiview on narrative documentation. This is now shown as Figure 5.15. Of course, the standards presented here can be adapted to suit the house rules of the particular organisation.

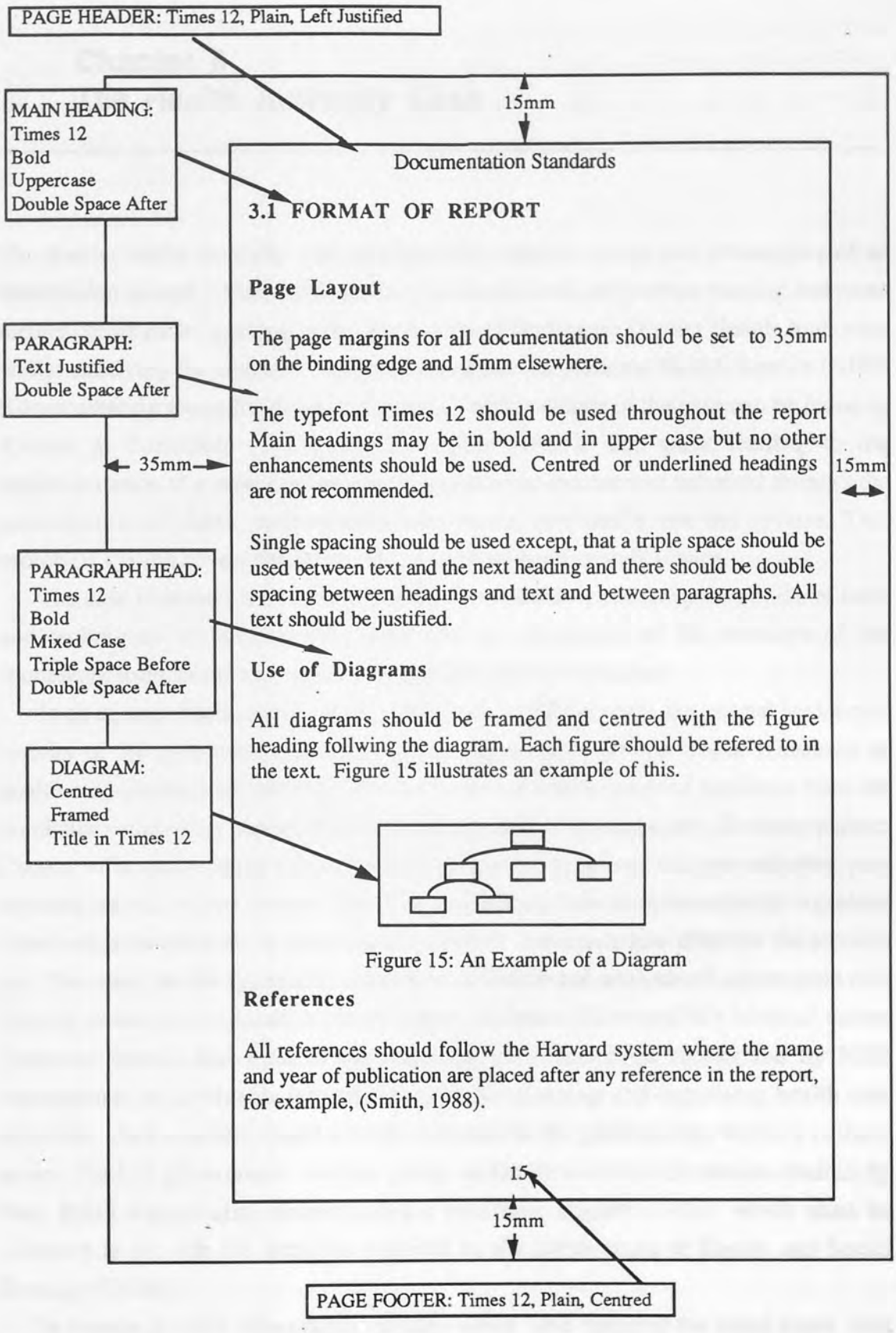


Fig.5.15. Suggested documentation standards

Chapter 6
The Health Authority Case

The district health authority case concerns the analysis, design and prototyping of an information system which fulfils some of the local needs of fourteen nursing and para-medical professions working in the community of Darlington District Health Authority, whilst satisfying the statutory requirements of the UK National Health Service (NHS) Körner steering group for those professions. Further details of the case can be found in Avison & Catchpole (1987) and Catchpole (1987). The work leading to the implementation of a prototype computer information system was achieved through the participation of those professionals who would eventually use the system. This prototype system covers the chiropody and school nursing staff groups.

The case illustrates the contingency aspect of Multiview, that is, the choice of tools and techniques within the framework and the adaptation of the structure of the framework itself to suit the needs of a particular problem situation.

In an average health district in the UK (there are 220 in total), the normal health care activity of the community (non-hospital) health services creates tens of thousands of health care interactions annually. These relate to a whole range of activities from the domiciliary midwifery service to the domiciliary care of the terminally ill elderly patient. Current information about these interactions is stored in several discrete and often very separate manual record systems. There is surprisingly little comprehensively organised information available about them and it is difficult to evaluate how effective the services are. The need for the systematic collection, collation and analysis of information of a 'service' nature (type, duration and frequency of interactions) and of a 'clinical' nature (patients' details and reasons for requiring care) has been recognised by NHS management as a valuable information base for planning and organising health care activities. Such a system would provide feedback to the professionals working in these areas. The UK government steering group on health service information chaired by Mrs. Edith Körner also recommended a minimum amount of data which must be recorded to provide the statistics required by the Department of Health and Social Security (DHSS).

In current discrete information systems which have operated for many years, data has been collected on forms relating to the provision of services in the community. Individual practitioners submit data on their activities to managers, usually on a monthly basis. A characteristic of this data is that it is of high volume, which means

that it is very difficult if not impossible to process manually into meaningful information. Data collection is often duplicated by different staff groups. Returns made by staff are frequently stored away for many years without use being made of them. The recording of this data is often carried out merely to fulfil statutory requirements or for no good reason at all.

An increasing emphasis is being placed on caring for clients in the community situation and the potential of community information systems is large. At this particular health authority the number of beds in the district general hospital is 750. One community service, school nursing, has 20,000 registered clients. This data could provide valuable epidemiological and demographic information. The opportunity is presently being lost.

The remit was to analyse, design and implement a prototype of an appropriate information system for the community health services of the health district, whilst fulfilling the related statutory requirements of the NHS Körner steering group.

The first phase of Multiview is an analysis of human activity systems. The familiarisation process had two strands. A survey of computer applications for the community aspects of the NHS, documented in Catchpole (1985), was carried out and professionals working in community health for the health authority were interviewed and accompanied on visits to clients. It was obvious from the survey that there was no existing information system that could be used 'off the peg'. It was necessary to develop one.

Having established this, possible frameworks for developing information systems were discussed with management. This included the more popular information systems methodologies such as SSADM and IE, but although there are advantages to be gained from standardisation, this type of methodology can be rigid. The view that carried most weight was that this approach tends to preclude the real (rather than lip-service) participation of the users, the methodology being a strait-jacket preventing them from contributing effectively, and that people ought to have control over their work environment wherever possible, including the computer systems that they use. For this reason the Multiview methodology (or more accurately, methodology framework) was thought appropriate for the application area and flexible enough for user groups to have a real influence on decisions made.

It was felt that the framework could lend itself to participation of user groups and at all stages of the development of the informations system, not only the design of the human-computer interface. A second and related requirement was for prototyping which again is enabled using Multiview. This facilitates the demonstration of aspects of a potential system to be made without any irreversible work being done on the real system. A quick delivery of a skeletal working system can be made which will test out

design principles of a system with users, who can then be involved participatively in the construction of the system. A prototype can be used as a tangible starting point for discussions. As seen in the first case, it may be possible to use the prototype as a basis for the operational system - it 'becomes' the operational system after many iterations when users regard it as being satisfactory. In the health authority case, the prototype was more of a fact finding tool which was used as one base for the Comcare system that followed. A final requirement of the methodology was to enable the setting-up of the database in such a way that it could be useful to the various professional groups and managers at the health authority and fulfil the Körner requirements. The data analysis/database approach discussed in Avison (1985) was used as a basis for this aspect of the Multiview framework.

The framework for the methodology to develop the information system therefore was an adaptation of Multiview, and is shown in Figure 6.1. We give an outline of the methodology below (the numbers refer to the boxes in the diagram).

1. *Familiarisation and boundary definition:*

Investigate the organisation, in particular by discussing the application area with members of staff, and draw boundaries around the area affected by the information system.

2. *Körner analysis:*

This activity is specific to this application area, and involves analysing the various reports (Körner, 1982, 1984a, 1984b) which are relevant to community health and extracting the requirements in terms of data collection and reports and other outputs. Participation is not appropriate here.

3. *Process analysis:*

This involves the construction of a requirements definition, again in terms of data collection and outputs, set out by the system's stakeholders. People with a stake in the project include the staff groups, management and clients. User views are recorded via interviews and group sessions led by the users themselves.

4. *Consolidation:*

The requirements of the various users are merged along with the Körner recommendations to form a consolidated requirements set. The data collected which is needed to support these requirements is also merged to form the basis of a data dictionary.

5. *Human-computer interface:*

The reports and displays identified at Stage 4 are designed in accordance with the users' experience and background. Users are encouraged to experiment with prototypes to determine themselves which approach would be the most appropriate.

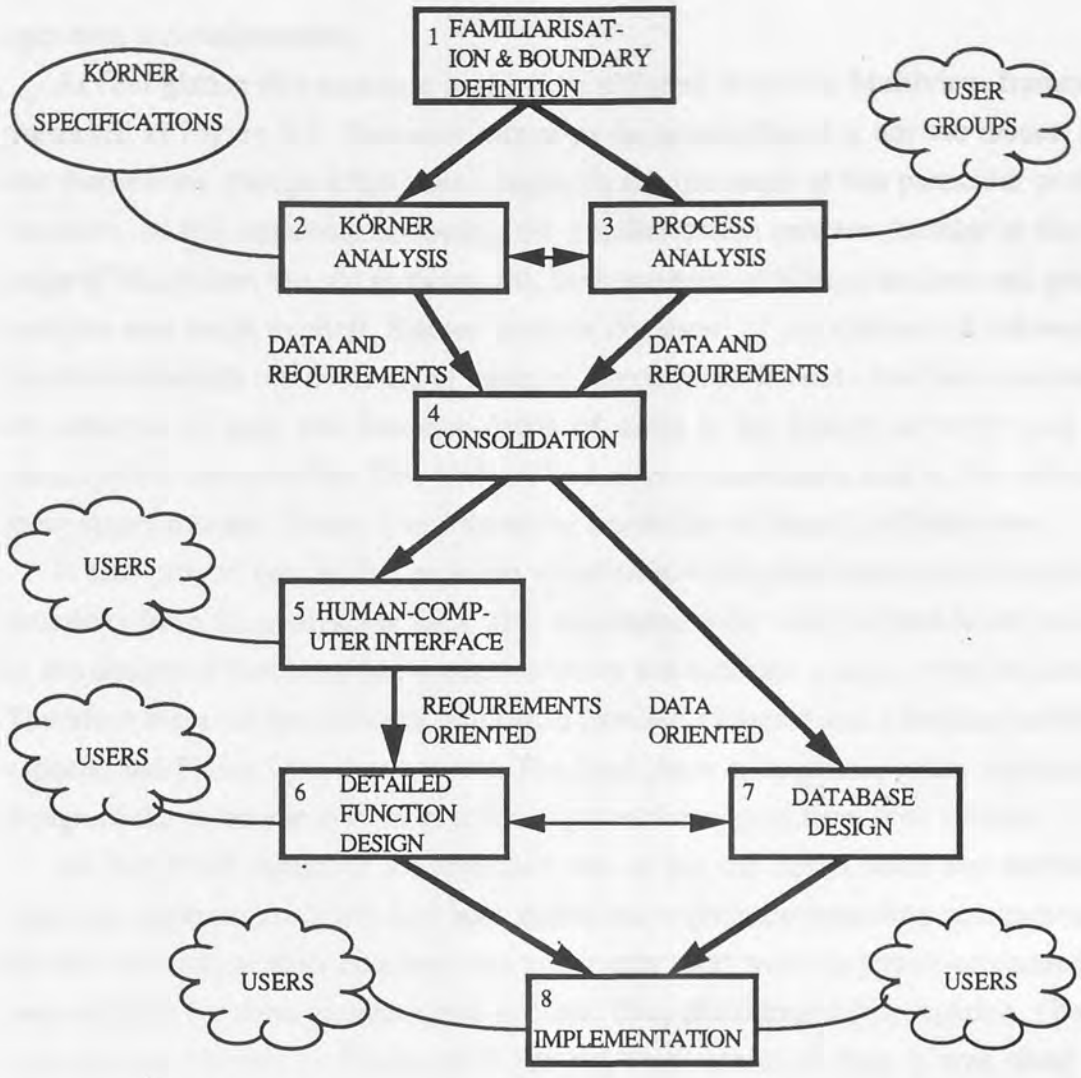


Fig.6.1. The methodology framework

6. *Detailed design of functions:*

The functions outlined in Stage 3 are analysed in greater depth and the menus, dialogues and reports designed in detail according to the principles established in Phase 5. Again, the participation of the users is essential to ensure that they get the system that they want.

7. *Detailed design of the database:*

The data requirements identified in the consolidation phase are redefined in a form suitable for the target database management system, which was chosen partly because it could offer the range of user interfaces required in Stage 5. The operational system, unlike the prototype, needs to be designed to fulfil performance, security, privacy and other criteria to a satisfactory standard.

8. *Implementation:*

This phase covers the processes towards changing to the new system and its regular

operation and maintenance.

At first glance this structure looks very different from the Multiview framework presented as Figure 3.2. However, closer study reveals that it it not too distant from that framework, though it has been adapted to suit the needs of this particular problem situation. In this situation, following the familiarisation process (similar to the first stage of Multiview, but not as thorough), the separation of Körner analysis and process analysis was made explicit. Körner analysis consisted of the analysis of information required to satisfy the demands of national government. Process analysis consisted of the analysis of data and function needs of users at the health authority and here participation was possible. This analysis had to be consolidated, that is, the outputs of these stages merged. Phases 2 to 4 therefore are similar to Stage 2 of Multiview.

It also proved best in this problem situation to explicitly separate the analysis of functions from the analysis of data. This was because the users wished to be involved in the design of functions but wanted to leave the database design to the facilitators. Therefore there are two streams running in parallel: Phases 5 and 6 looking at process aspects, and Phase 7 the data aspects. The final phase of implementation included the design of the prototype system, and this required the outputs from both streams.

An important aspect of the approach was to use the newer tools and techniques wherever appropriate. Users had been rather unsympathetic regarding computers, and the documentation tools that they had previously used were computer-orientated and very difficult for them to understand and use. They discouraged participation. The first rich picture (shown as Figure 6.2) proved very useful in that it was used as a discussion document to stimulate contributions from users.

Because of the wide area of concern and complexity of this particular case, a series of rich pictures were produced as well as other documents (Catchpole 1987). Figure 6.3 shows a sample entity model, Figure 6.4 a sample data flow diagram and Figure 6.5 a function diagram relating to the chiropody service aspects of the community health programme. Forms for specifying entities, attributes and relationships were used and part of the consolidated entity model (Stage 4) is shown as Figure 6.6. This is used as the basis for an entity type specification for the entity 'category-of-programme', shown as Figure 6.7; a relationship type specification for the relationship 'consists-of-2', shown as Figure 6.8, and an attribute type specification for 'category-name' (an attribute of the entity type 'category-of-programme') shown as Figure 6.9. Part of the function/entity matrix is shown as Figure 6.10. Figure 6.11 shows the entity life cycle for one of the entities 'person-in-programme'.

It was recognised that there was a need for those people in the organisation who will eventually use the information system to take part in the decision-

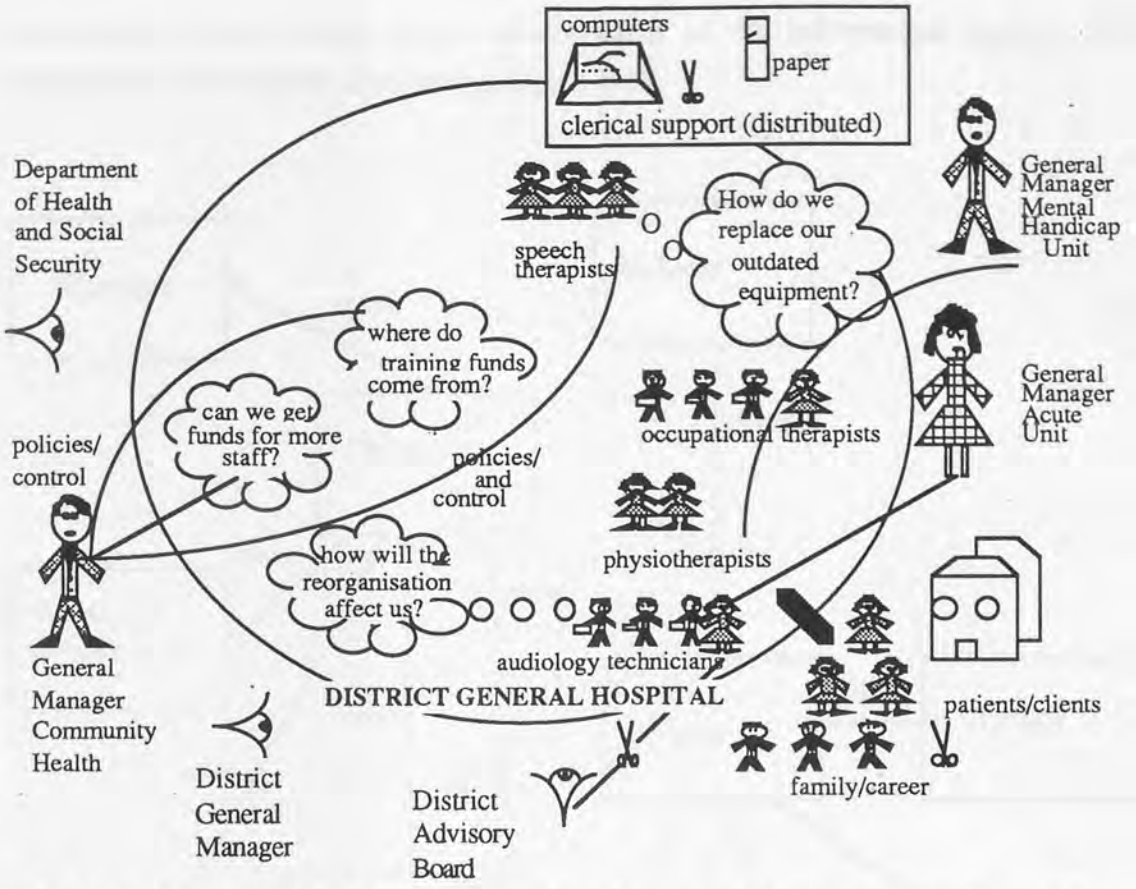


Fig. 6.2. An early draft of a rich picture chart for part of the para-medical services

making relating to analysis and, in particular, design. Of the three approaches described in Section 2.5, the representative approach was thought appropriate and selected. This approach was preferable to consultative participation which still leaves the bulk of design decisions to the systems analyst/designer.

Consensus participation would have allowed all members of the community health services to be involved in the systems design process. However, for this particular problem situation, a continual high quality operational service must be maintained and so the approach would have been impractical in view of the lengthy time scales which were involved.

A design group was formed consisting of various grades of staff from the community health services representing the professional staff groups. Volunteers were used who had an interest in the development of a system from the non-management grades. The agreement of senior management was obtained to enable the members of the design group to participate. An important member of the group was the facilitator who led the group in the tasks to be performed. For this problem situation, the facilitator took on an active role, training the design group in the application of the methodology and also carrying out tasks which were not thought appropriate or necessary for the group as a whole to participate. He also advised on the various

alternatives at each stage in the development of the information system. The composition of the group is shown as Figure 6.12.

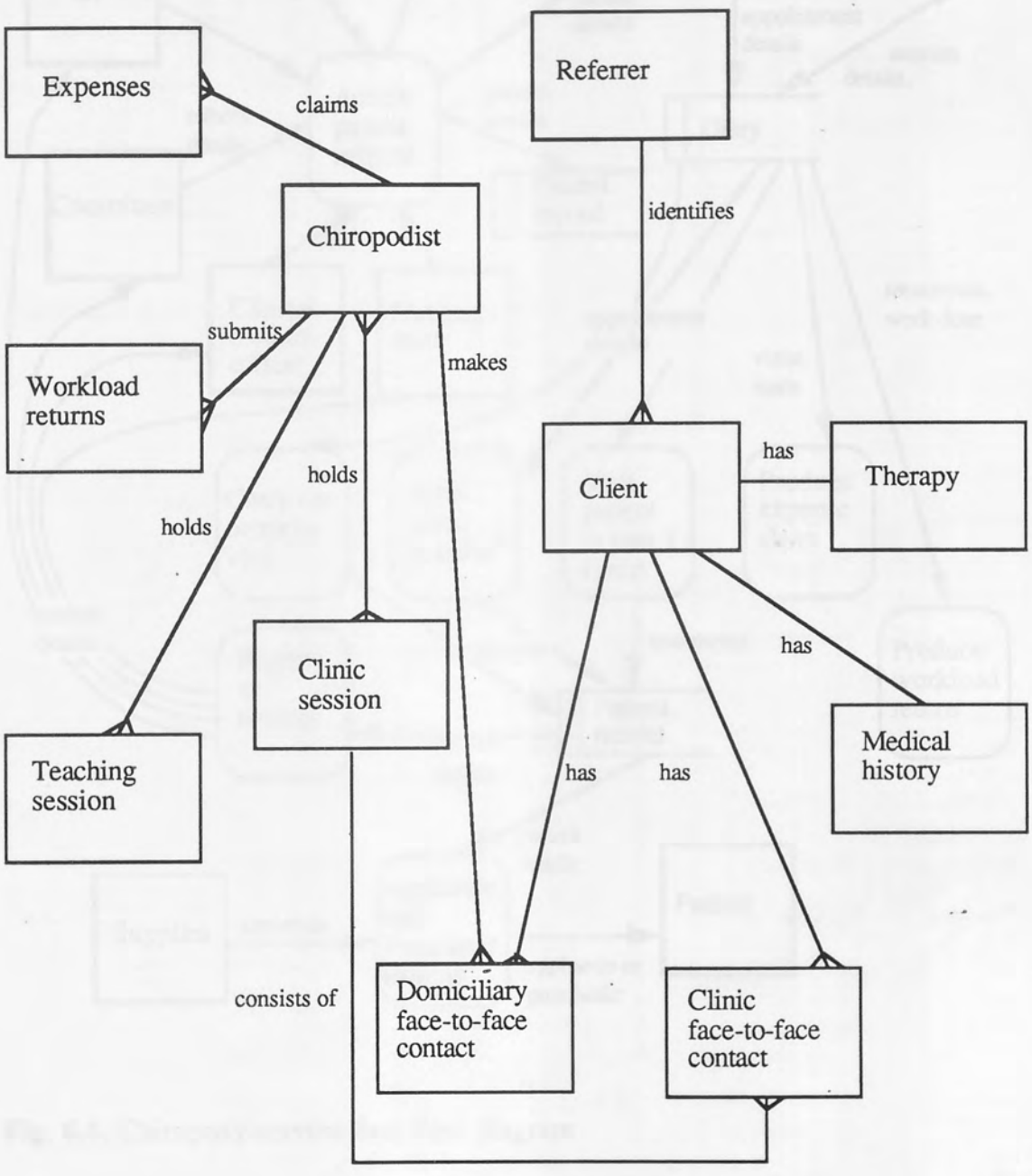


Fig. 6.3. Chiropody service entity model

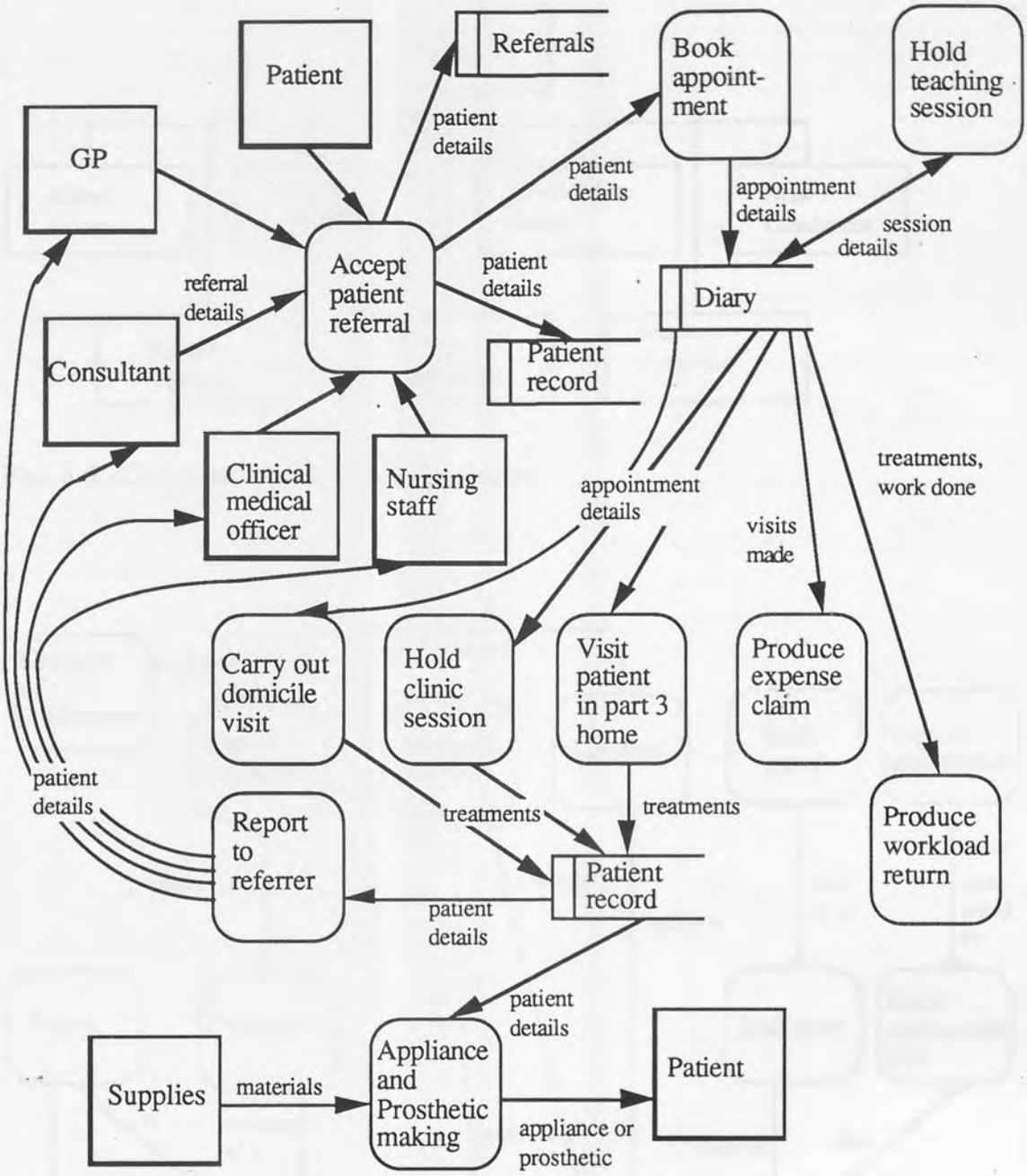


Fig. 6.4. Chiropody service data flow diagram

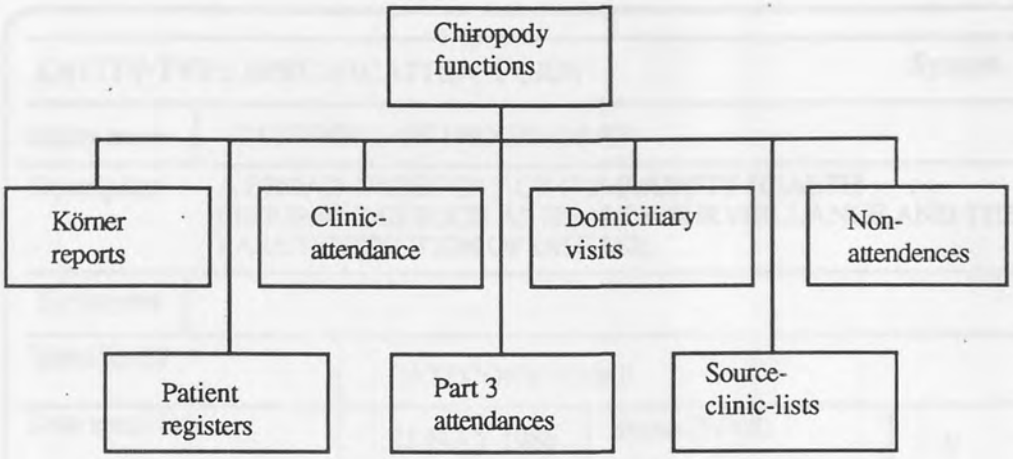


Fig. 6.5. Chiropody service function model

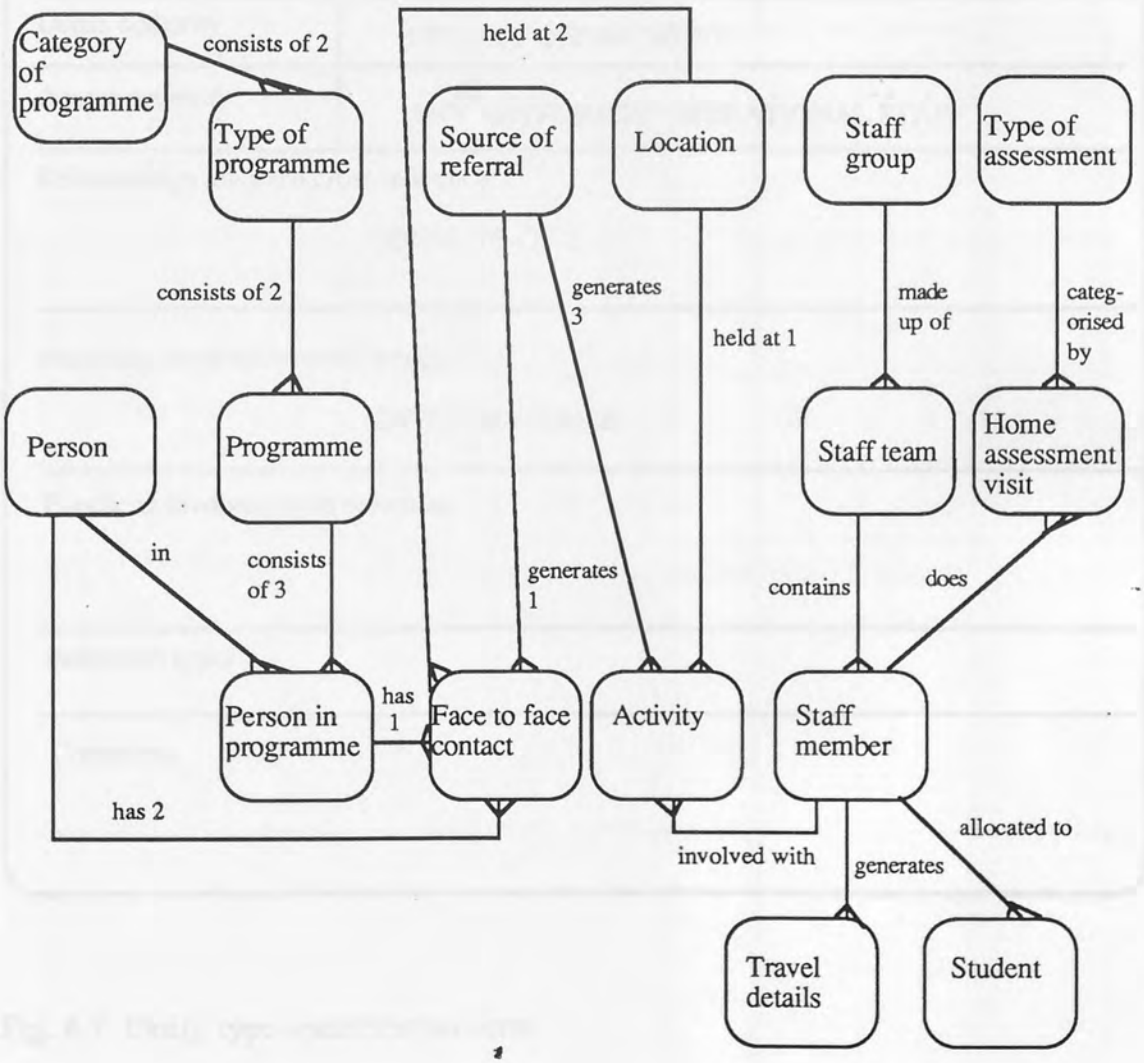


Fig. 6.6. Consolidated entity model (part)

ENTITY TYPE SPECIFICATION FORM System: CHIS

Entity name	CATEGORY-OF-PROGRAMME		
Description	A BROAD CATEGORY OF COMMUNITY HEALTH PROGRAMME SUCH AS HEALTH SURVEILLANCE AND THE EARLY DETECTION OF DISEASE		
Synonyms			
Identifier(s)	CATEGORY-NAME		
Date specified	15 MAY 1986	Status (P/V/I)	V
Minimum occurrences	1	Maximum occurrences	10
Average occurrences	3	Growth rate %	
Create authority	HEAD OF DEPARTMENT		
Delete authority	HEAD OF DEPARTMENT		
Access authority	ANY AUTHORISED OPERATIONAL STAFF		

Relationships involved cross reference

CONSISTS-OF-1

Attributes involved cross reference

CATEGORY-NAME

Functions involved cross reference

Entity sub types

Comments

Fig. 6.7. Entity type specification form

RELATIONSHIP TYPE SPECIFICATION FORM System: CHIS

Relationship name	CONSISTS-OF-2		
Description	A TYPE OF PROGRAMME CONSISTS OF ONE OR MORE INDIVIDUAL PROGRAMMES		
Time state	1 YEAR		
Synonyms			
Date specified	15 MAY 1986	Status (P/V/I)	V
Entities involved	(Owner) TYPE-OF-PROGRAMME (Members) PROGRAMME		
Degree (1:1, 1:m, m:n)	1:m	Optional Contingent Mandatory	MANDATORY
If contingent, state optional entity			
If exclusive, state paired relationship name			
If inclusive, state paired relationship name and first existence relationship name			
Create authority	HEAD OF DEPARTMENT		
Delete authority	HEAD OF DEPARTMENT		
Access authority	ANY AUTHORISED OPERATIONAL STAFF		
Comments			

Fig. 6.8. Relationship type specification form

ATTRIBUTE TYPE SPECIFICATION FORM System: CHIS

Attribute name	CATEGORY-NAME		
Description	NAME OF THE CATEGORY THAT PROGRAMME TYPES FALL INTO		
Synonyms			
Date specified	15 MAY 1986	Status (P/V/I)	V
Entity cross reference	CATEGORY-OF-PROGRAMME TYPE-OF-PROGRAMME PROGRAMME		
Create authority	HEAD OF DEPARTMENT		
Delete authority	HEAD OF DEPARTMENT		
Access authority	ANY AUTHORISED OPERATIONAL STAFF		
Functions involved cross reference			
Comments			

Fig. 6.9. Attribute type specification form

FUNCTION NAME \ ENTITY NAME	STAFF GROUP	STAFF TEAM	STAFF MEMBER	GROUP SESSION	LOCATION	PROGRAMME	CATEGORY PROGRAMME	TYPE OF PROGRAMME	PERSON IN PROGRAMME	PATIENT PERSON
1 GROUP-SESSION-ATTENDANCES	X	X	X	X	X					
2 PROGRAMMES	X	X	X			X	X	X	X	X
3 CONTACTS-TRACED	X	X	X			X			X	X
4 ASSESSMENTS	X	X	X			X			X	X
5 ACTIVITY-SUMMARY	X	X	X	X						
6 ACTIVITY-SUMMARY-OTHER	X	X	X							
7 PROGRAMME-COSTING	X	X	X	X		X	X	X	X	
8 VOLUNTARY-SERVICES	X	X	X							X
9 OTHER-SERVICES	X	X	X							X
10 ACTIVITY-SUMMARY-REFERRAL	X	X	X	X						
11 ACTIVITY-SUMMARY-LOCATION	X	X	X	X	X					
12 CONTACT-ACTIVITY	X	X	X							
13 NON-ATTENDANCES-REFERRAL	X	X	X	X						
14 SOCIAL-SUPPORT	X	X	X							X
15 FUNCTIONAL-IMPAIRMENT	X	X	X							X

Fig. 6.10. Function/entity usage chart

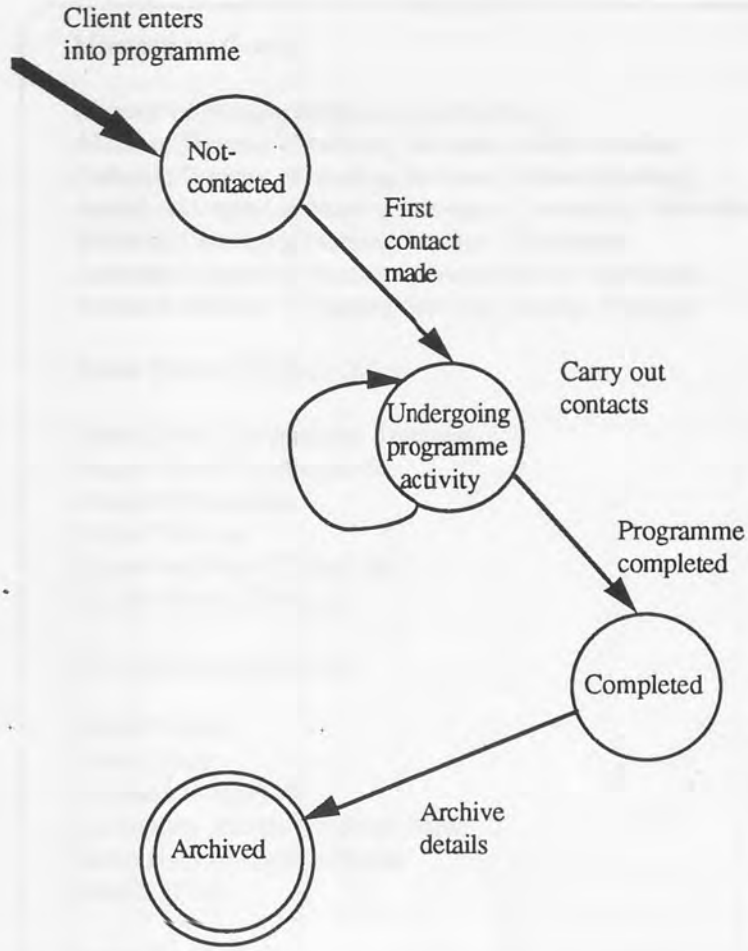


Fig. 6.11. Entity life cycle for 'Person-in-Programme'

The design group met together as a whole and also in smaller teams as appropriate. For the facilitator to gain an initial knowledge of the community health services, and more importantly to become familiar with members of the design group, a period was spent where the facilitator worked with all of the design group members for one day each, to observe the tasks they carried out. For example, for the six nursing staff grades, time was spent travelling around in the community, visiting clients in their own homes. This time proved to be well spent and helped form good relationships between the facilitator and design group members.

There were 110 separate reports defined by the users (see Figure 6.13). The number of reports required to provide all the Körner minimum data set information for the community and para-medical services collectively was 15. By merging all of these reports with those requested by the users, a total of only 38 reports needed to be produced which satisfied all the Körner requirements and 85% of the original user requests. The other 15% of reports were categorised by the users as 'quite nice to have' but not essential for their work.

A second area where the information system requirements of the users and Körner

Management Grades

Director of Nursing Services (Community)
Assistant Director of Nursing Services (Health Visiting)
Assistant Director of Nursing Services (School Nursing)
Assistant Director of Nursing Services (Community Midwifery)
Assistant Director of Nursing Services (Psychiatry)
Assistant Director of Nursing Services (Mental Handicap)
Assistant Director of Nursing Services (District Nursing)

Senior Clinical Medical Officer

District Head Occupational Therapist
District Head Physiotherapist
District Chiropodist
Senior Dietician
Senior Audiology Technician
District Speech Therapist

Non-Management Grades

Health Visitor
School Nurse
Community Midwife
Community Mental Handicap Nurse
Community Psychiatric Nurse
District Nurse

Senior Psychologist
Dietician
Audiology Technician
Speech Therapist

The Facilitator

Fig. 6.12. Composition of the design group

proved less costly than expected concerns data collection. The standardisation of reports also enabled a common approach to data collection and only two types of input documents were designed for the new system. A simplification of the data recording process leads to improvements in the quality and accuracy of the information recorded. Minor modifications to the standard forms were made to meet the individual requirements of each service. Further, as Stages 1 and 3 showed, health visitors, school nurses, community midwives and other professionals working in the community health services, already collect data relating to their work. As this formed the basis of the data requirements for the information system specified, no extra data collection was necessary.

We were fortunate to receive the full co-operation from user groups. These included

management grades, such as the director of nursing services for the community, the six assistant directors of nursing services, the senior clinical medical officers, and district heads and non-management grades such as health visitors, school nurses, community midwives, district nurses and dieticians. These formed design groups meeting regularly to decide on overall strategy as well as detailed designs.

STAFF GROUP	NUMBER OF REPORTS ASKED FOR	NUMBER OF REPORTS INCORPORATED	% INCORPORATED
HEALTH VISITORS	12	12	100
SCHOOL NURSES	12	12	100
DISTRICT NURSES	7	6	85
COMMUNITY MIDWIVES	6	6	100
PSYCHIATRIC NURSES	9	6	66
MENTAL HANDICAP NURSES	8	8	100
CLINICAL MEDICAL OFFICERS	10	6	60
OCCUPATIONAL THERAPISTS	6	6	100
PHYSIOTHERAPISTS	5	5	100
PSYCHOLOGISTS	10	6	60
DIETICIANS	7	5	71
AUDIOLOGY TECHNICIANS	6	6	100
CHIROPODISTS	6	5	83
SPEECH THERAPISTS	6	4	66
TOTAL	110	93	AVERAGE 85%

Fig. 6.13. Consolidating Reports

Users were also happy to try new tools and techniques such as portable hand-held data recorders as an alternative to the redesigned forms. The portable data recorders, no bigger than hand calculators, enable direct data entry into the computer system (see Catchpole, Avison & Peart, 1987 for a discussion of this trial). This experiment was not imposed on staff - the facilitators made the staff groups aware of the various ways of collecting data and the staff groups chose to try both the recorders and the redesigned forms in the prototype.

The prototypes were designed using an application generator which generated the programs from a specification of the data (and validation procedures), creating the database (Stage 7), and screen and report formats. It also created a data dictionary and maintenance software. The design groups were able to view screen and report layouts as they were 'painted' on the screen and suggested changes as an iterative process. A series of menus were also designed by the user groups to provide access through the system in accordance with the recommendations made at Stage 5. A password security system was also implemented.

Two staff groups, chosen by the design group, carried out a pilot study. One nursing group (school nurses who carry out 'group session' type activities) and one para-medical group (chiropodists as they carry out 'face-to-face contact' type activities) were chosen for the pilot study because they provided a contrast to test different parts of the system. A telephone 'help line' service operated during this period to assist users. A series of review meetings were held to evaluate progress throughout the period and consider any problems that had arisen.

Figure 6.14 shows a structured English specification relating to a report used in the health information system giving details of those people who have been referred by various consultants. As can be seen, it has a logical structure that is similar to that of a computer program, though it is easier to follow than conventional programming languages. Unfortunately no applications generator was available at the time on the hardware used to interpret structured English and produce code directly from it. Again, this is an example of real-world constraints, in this case the given hardware, producing a less than ideal environment for the systems analysts using the methodology.

A number of forms were completed as input documents to the applications generator (effectively a statement of requirements in a very formalised way) and this was processed by the applications generator package to produce the required programs (producing code in the standard computer programming language, Cobol).

In order to design the applications, we already had the following information from the earlier analysis and design stages, although some of these had to be fleshed out in more detail as a result of the prototyping stage:

- *The tasks that had to be done* (from the function chart); for example, school nurses' timetabling and daily diary sheets, chiropodists' work allocation, and the detailed functions within each.
- *The data that had to be collected during each transaction* (from the entity/event model); for example, the doctor's name and address is collected during school-child referral and the details of each patient are collected when registering patients.
- *The type of input method* (from the socio-technical design stage); for example, data recording on hand-held data recorders or on paper forms (this varied according to the user group).
- *The sort of computer system that would be used* (from the socio-technical design stage); for example a minicomputer system capable of processing batches of data coming from the health workers (including direct input from the data recorders) and with on-line enquiry facilities.
- *The nature of the dialogues that would have to take place and the style that would be appropriate to each* (from the human-computer interaction); for example, menu mode for application selection and forms mode for data entry and for retrieval of

STRUCTURED ENGLISH SPECIFICATION System: CHIS

Name

CONTACTS-LOCATION REPORT

Description

LOGIC OUTLINE

```
produce report-headings
  REPEAT {for each staff group}
    access staff group record
    print staff group
    REPEAT {for each staff team within group}
      access staff team record
      print staff team
      REPEAT {for each staff member within team}
        access staff member record
        REPEAT {access all contacts for that record}
          access contact record
          cumulate by locations specified
        UNTIL all contacts accessed
      UNTIL all staff members accessed
      REPEAT {for each location identified}
        print report-line
      UNTIL all locations printed
      print staff team totals
    UNTIL all staff teams accessed
  print staff group totals
  UNTIL all staff groups accessed
```

Keywords: IF....THEN....ELSE....SO....
REPEAT....UNTIL
CASE.....OF....[conds:stmts]....OTHERWISE....ENDCASE

Fig. 6.14. Structured English specification

individual items.

Most systems now also give the user opportunities to make enquiries directly using a conveniently located computer terminal or a microcomputer with compact disk or hard disk containing the information. A query in the community health prototype might look like:

PRINT ALL VISITS WHERE DATE=31.05.90 AND CHIROPODIST=JONES

This would produce a list of visits that this particular chiropodist had to make on that day. A language like this is very suitable for people who regularly use a computer as part of their job. Such a language is not too difficult to learn, and usually a few half-day training sessions are sufficient. Provided that users know what information is available in the database, they can obtain it in a flexible manner.

Where an information retrieval system is to be used by occasional users it needs to be much easier to operate. The most common way of offering this ease of access is through a menu. In the community health case study both dialogues and menus were used, depending on the user.

Figure 6.15 shows part of one menu structure. On the top level the user is asked whether the reports required relate to the Chiropody Service or the School Nursing Service (the two pilot systems). The user has keyed in '1' to bring down the second level menu for the Chiropody Service. This menu lists six types of report, and again the user has keyed in '1' to request a report on face to face contacts. The next level menu suggests alternative ways in which the report can be displayed. The user could key in '1' to ask for the report to be given in sequence of location. That particular report (shown as Figure 6.16) will then be displayed on the screen. If the user had keyed in '3', then the report shown as Figure 6.17 would be displayed and if the user had keyed in '2', then the report shown as Figure 6.18 would be provided. If, on the other hand, the user had keyed in '3' in the second level menu, control would lead straight to the display of the patient register (Figure 6.19) as there is no lower level menu for that.

In the community health prototype, the information retrieval required was of two kinds:

- Regular reporting to health visitors, school nurses, and so on.
- One-off queries as they arise, for example, progress checks on an individual patient.

Many of the reports could be determined in advance from known or predictable requirements, but there would undoubtedly be new ones needed to cope with new situations. The Government bodies involved, such as the DHSS, tend to require reports of a kind demanded by the Government Minister responsible. These are sometimes

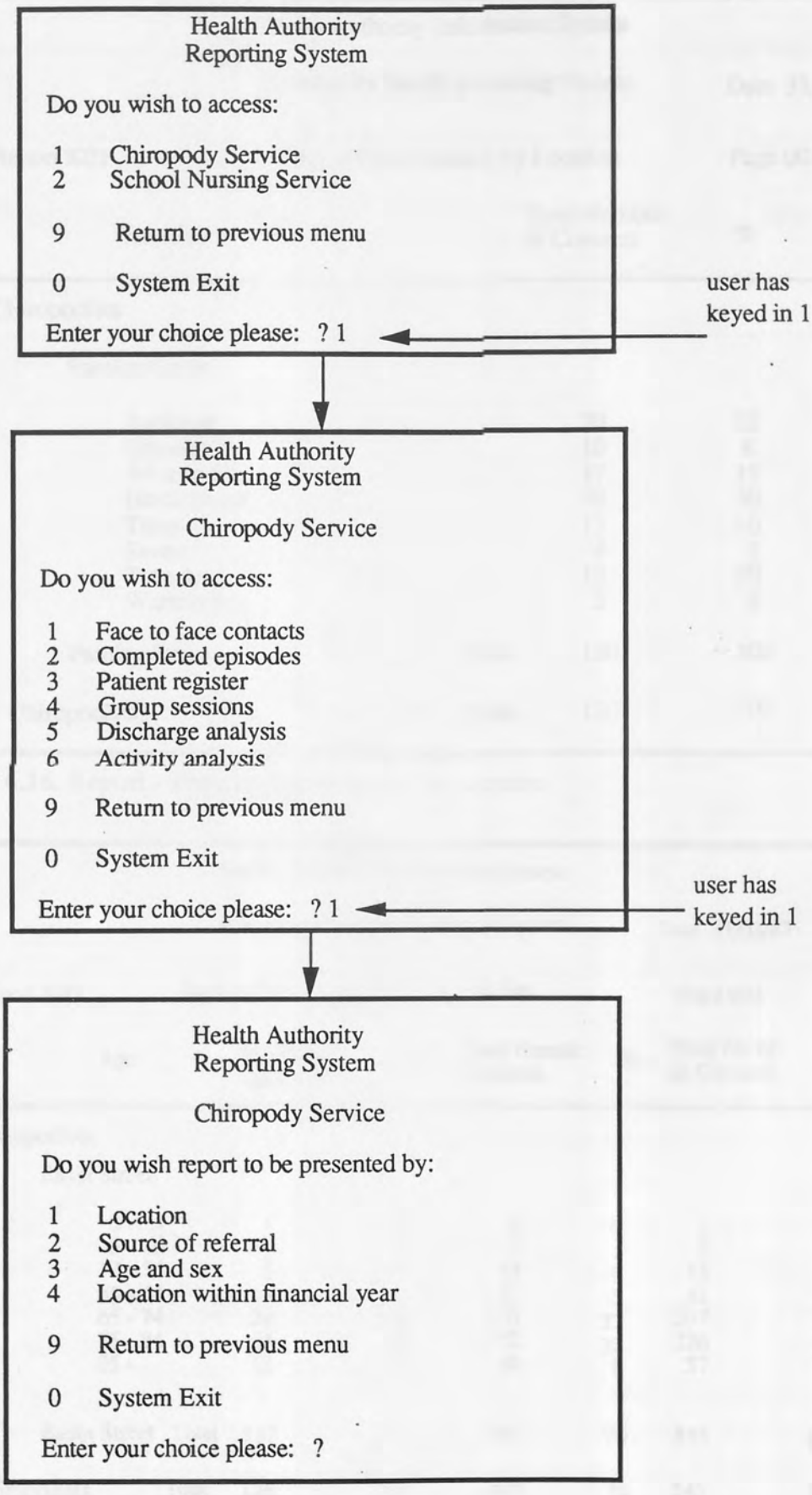


Fig. 6.15. Menu structure for reporting system

Health Authority Information System

Community Health Reporting System Date 31/03/87

Report K01 Face to Face Contacts by Location Page 001

Location		Total Number of Contacts	%
Chiropodists			
Farnley Street			
Rackman		29	22
Graneham		10	8
Whitworth		17	13
Hurlingham		39	30
Tulsa		13	10
Seven		4	3
Totterham		13	10
Warmington		5	4
Farnley Street	Total	130	100
Chiropodists	Total	130	100

Fig. 6.16. Report - Face to face contacts by location

Health Authority Information System

Community Health Reporting System Date 31/03/87

Report K03 Face to Face Contacts by Age and Sex Page 001

Age	Total Male Contacts	%	Total Female Contacts	%	Total No of all Contacts	%
Chiropodists						
Basin Street						
0 - 4	1	0	0	0	1	0
5 - 16	1	0	7	1	8	1
17 - 54	1	0	14	3	15	3
55 - 64	4	1	27	5	31	6
65 - 74	58	11	149	27	207	38
75 - 84	54	10	172	32	226	42
85 -	13	2	44	8	57	10
Basin Street Total	132	24	413	76	545	100
Chiropodists Total	132	24	413	76	545	100

Fig. 6.17. Report - Face to face contacts by age and sex

Health Authority Information System

Community Health Reporting System Date 31/03/87

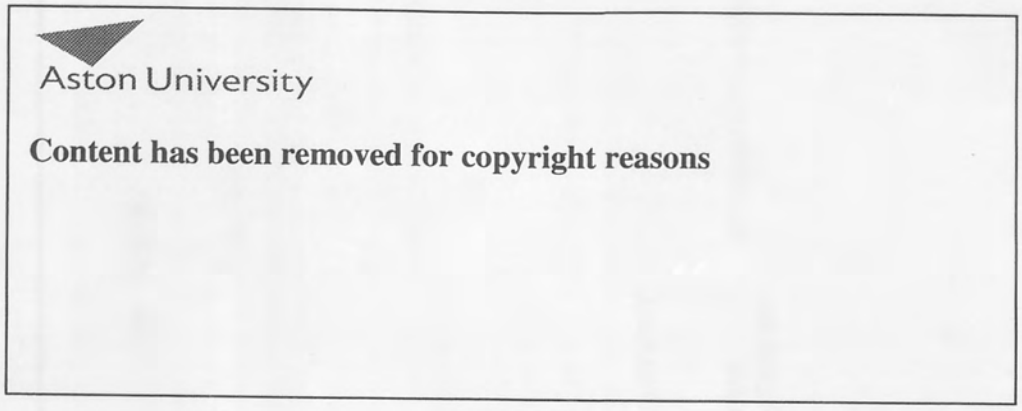
Report K02 Face to Face Contacts by Source of Referral Page 001

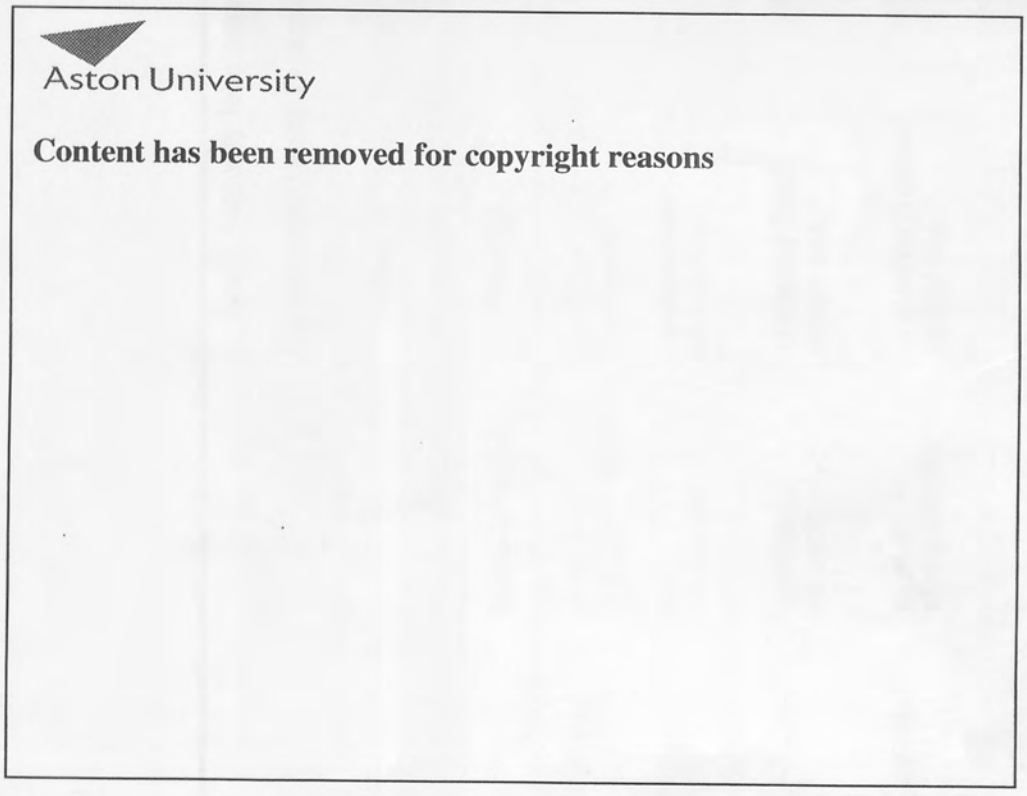
Source of Referral		Total Number of Contacts	%
Chiropodists			
Farnley Street			
Other		72	55
Self Referral or Relative		54	42
G.P		4	3
Farnley Street	Total	130	100
Basin Street			
Self Referral or Relative		451	84
G.P		33	6
Other		34	6
Community nurses etc		24	4
Clinical medical officer		1	0
Consultant		2	0
Basin Street	Total	545	100
Chiropodists	Total	675	100

Fig. 6.18. Report - Face to face contacts by source of referral

unpredictable because they depend on the particular political climate of the time. Usually they can be supplied from the existing database by a process of selecting and sorting. This is not always the case. For example, a request from one particular source required the health authority to give information on the ethnic background of patients which was, as a matter of principle, not kept in the system. So as to fulfil the request, a voluntary questionnaire was completed by patients which we hoped would satisfy requirements.

Not all possible reports were defined although some attempt was made to predict some of the entities and attributes that might be important in reports. This was carried out by studying the politics of health and the sorts of questions that Ministers, health authorities, and employers would be likely to ask us. Future analysis (Section 3.4) was used in this exercise.





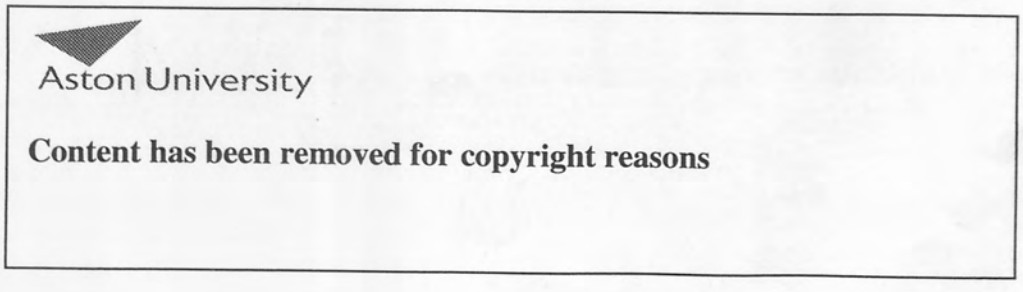


Fig. 6.19. Report - Alphabetic patient register

For the one-off enquiries, one attempt was made to predict the sorts of questions that were going to be asked. These were all collected and the questions were used to test the information model. Some of the questions could be answered in reports and others by setting up particular enquiries.

The database was set up using an applications generator package which facilitated the entry of database descriptions for the entities and attributes and the setting up of the relationships. This solution avoided many of the technicalities normally associated with setting up a database. The system did impose some restrictions on database design, for example, a maximum number of data items per file and a maximum record length, but none of these restrictions proved a problem in the case.

Many of the insertions, amendments and deletions will be carried out as part of some application. For example, the patient registration application will delete the corresponding referral record and create a new patient record. Similarly the receipt of visit data regarding a recent visit by a health visitor to his or her client will cause an amendment to the patient record so that it includes the additional information about the progress of the patient. The amendment may be achieved technically by wiping out the old record and creating a new one, or by writing the changed record back into the same place. This will depend on the particular technical decisions that will have been taken and this is in the domain of the technical analyst.

The raw data for inserting, amending or deleting records in the database was either input directly from the data recorders or keyed in from completed forms. The hand-held data recorders are small, battery-powered devices which can be programmed to collect data and store it in memory, and later be 'electronically drained' directly into the main computer.

Controls are particularly important in health service applications. Here are some of the checks that were applied:

- *Is the item of the specified data type?* For example, a patient number must be numeric.
- *Is the item within specified boundaries?* For example, largest and smallest acceptable value for numeric items (frequently called a range check), such as the month being greater than zero but less than 13.
- *Is there a limit on the actual values that may be used?* For example, if there are four items on a menu, only the numbers 1, 2, 3, or 4 are acceptable.
- *Can the item have a self checking mechanism?* For example, a check digit.
- *Can the item be cross-checked?* For example, the patient referral date must always be less than the patient date of death, individual amounts must always add up to the batch total given, and entries must be input in the correct sequence.
- *Is the data complete, have any items been missed out?*

- *Can the operator be asked to check the item?* For example, when entering a patient number, the record with that patient name could be displayed to see whether this was the correct patient.

Most users felt obliged to 'play with the system for a while', but none made much progress until one of the technical team was assigned to each group for systems testing. This person acted as the facilitator. This served two purposes:

- It provided training that was directly related to the job in hand. Most computer user training tends to be about how to operate a system and not how to perform the job that you want it to do.
- It meant that if a problem did emerge, it could be sorted out.

The facilitator carried out the training of the operator, whose tasks included controlling data entry, report extraction and distribution, and the handling of security procedures. The opportunity was also taken to help users get over their initial difficulties with an unfamiliar technology. This was frequently carried out on a small group basis. Following training and use (for approximately four months) came user evaluation. For example, the use of the portable data recorders was evaluated alongside an equivalent paper method for data collection. Training was given and feedback on the two methods was provided by an informal interview, group discussion and a questionnaire. The recorder was found to be easy to use and slightly faster, and on the whole was preferred.

The pilot system was assessed from the point of view of effectiveness, that is, the proportion by which the system meets organisational objectives and goals, and efficiency, that is, a measure of the 'mechanical' aspects of the system such as the accuracy, timeliness and speed of access to information. The training and design process was also reviewed, essentially a review of the participative and prototyping approach to information systems development that was chosen. In general, these reviews were designed to ascertain how Multiview worked in action. Opinions were solicited about the system from the senior managers as well as user groups and individual users. The methods used to assess the system were questionnaires with follow-up interviews and group discussions (group feedback analysis, Section 1.2.1).

Where possible, comparison of the prototype was made relative to the previous equivalent manual system which was replaced by the prototype, and both of these comparisons were related to theoretical objectives, goals and needs which were identified by the design group during application of the methodology. Costs and benefits associated with the old and new systems were identified, and then qualitative and quantitative items were evaluated covering such issues as services to clients, value added features, changes in the decision making process, and so on.

The questionnaires had four sections: effectiveness (objectives fulfilled, problems

addressed, needs addressed, costs and achieved benefits, impact on work and clients, and impact on decision-making and control); efficiency of the system (attributes of the information produced, aspects of using the information system, value placed on information generated, data collection methods, working practices of staff, and the personal aspirations of staff); the design process (in terms of perceived 'success' of the implementation, contribution made by 'participation' procedures, and personal benefits and gains of staff); and finally the training process.

The respondents considered that the system has helped people to perform their work tasks more effectively, drawing attention to what is actually being done, and has resulted in staff deciding to change their working practices. The system has improved the handling of case loads, improving timetabling and utilisation of clinical time. More up-to-date, accurate and easily accessible information will help in both short and long term service planning. Respondents also suggested a number of enhancements which could be added to the system such as an ad hoc search option and a patient register giving details of other professions and agencies working with the client. On the negative side, respondents did point out that, at least in the short term, data collection was slower. On working practices, comments such as "it was satisfying to see what work had been done at the end of the month", "I have been made more aware of what I have been doing" and "I have realised what a large amount of time is spent travelling and walking to homes and schools" are revealing.

Four of the fourteen tables that were derived from responses which lend themselves to statistical analysis have been included, though objectivity is not claimed in the assessment procedures used. Figure 6.20 displays the comments of school nurses regarding the degree to which the old and new systems met their information objectives (expressed as percentages). The old system fared badly partly because some of the objectives did not apply. However, it is readily apparent that the data collected for the old system led to few information gains. Figure 6.21 shows the comparative attributes of the two systems as perceived by all users of the prototype system. The new system seems to fare better on all attributes. Figure 6.22 looks at the value of the information provided using a scale of extremes for each value type. Figure 6.23 looks at the various views of the users concerning the use of the new system. This is the most revealing because it is critical of the new system. It highlights in particular the time necessary to use the new system, though on follow-up interviews this was seen as a 'once-and-for-all' investment of time.

As for the method of implementation, one school nurse commented: "...as it is the community health professionals who have to use the system, we are in the best place to help design it; we know what information is important and relevant for recording ....." and another: "I have been made aware of the difficulties in the system design

process" and another: "I have improved my knowledge of computer systems, and found this experience both interesting and beneficial". Many of the participants argued that they had enjoyed being involved, and the expansion of the prototype to all the community professionals and inclusion of future enhancements could be implemented successfully with the continued participation of those involved. As one school nurse recorded, "the system has been a success because we are all still using it after the pilot (study) has been completed".

INFORMATION OBJECTIVE	OLD SYSTEM SCORE (%)	NEW SYSTEM SCORE (%)
TO MEET THE KÖRNER OBJECTIVES	0	80
TO AVOID THE STORAGE OF DUPLICATE INFORMATION	0	90
TO PROVIDE ON-LINE ACCESS TO BASIC PATIENT DETAILS AND RECENT CONTACTS	30	90
TO SAVE TIME IN ACCESSING INFORMATION	10	100
TO PROVIDE AN UP-TO-DATE PATIENT REGISTER	0	100
TO DECREASE TIME SPENT BY STAFF ON FORM FILLING	10	90
TO REMOVE THE NEED TO MANUALLY AGGREGATE STATISTICS	0	100
TO PROVIDE MORE MEANINGFUL STATISTICS TO STAFF AND MANAGEMENT	20	100
TO PROVIDE BETTER INFORMATION FOR STRATEGIC MANPOWER AND SERVICE PLANNING; EDUCATION; EPIDEMIOLOGY AND DEMOGRAPHY	10	80
TO PROVIDE A SCHEDULING SYSTEM FOR WORKLOAD	20	50
TO PROVIDE A BETTER CONTROLLED STOCK SYSTEM FROM WORKLOAD	30	50

Fig. 6.20. Comparing old and new systems - information objectives

A great deal was learnt from this case. It was the first large application of Multiview using a more refined version of the approach and where the newer techniques and tools described in Chapter 5 were available. Not all these were deemed appropriate for the particular situation, although the newly specified data flow diagrams, entity life cycles and structured English specifications were used and proved to be valuable. Use was also made of an applications generator package, though some programs that were generated needed to be adapted by conventional means, and other programs were

ATTRIBUTE	OLD SYSTEM SCORE (%)	NEW SYSTEM SCORE (%)
ACCURACY OF INFORMATION	40	84
TIMELINESS OF INFORMATION	27	63
APPROPRIATENESS TO YOUR NEEDS	39	74
RELIABILITY OF INFORMATION	43	87
COMPLETENESS OF INFORMATION	36	85
SPEED OF ACCESS TO INFORMATION	43	67

Fig. 6.21. Comparing old and new systems - attributes

written in Cobol from scratch. This was partly due to the rather rudimentary package available on the particular minicomputers used (the system had to be run on equipment already in place). Procedures for evaluation and improved documentation also helped to make the system easier to maintain and the comments proved helpful when reviewing the prototype. This prototype was used in the design of the Comcare system which was adopted by Darlington Health Authority amongst many others.

Another experience gained from the case was the ability of the Multiview framework itself to be adapted to suit the requirements of the situation. Although the

VIEWS ON THE VALUE OF INFORMATION -2 -1 0 +1 +2	AVERAGE
TRIVIAL ____*____*____*X___ SOPHISTICATED	+1.19
USELESS ____*____*____*__X__ USEFUL	+1.50
UNIMPORTANT ____*____*____*__X__ IMPORTANT	+1.50
BORING ____*____*____*X___ INTERESTING	+1.13
VALUELESS ____*____*____*_X___ VALUABLE	+1.44
MEANINGLESS ____*____*____*_X___ MEANINGFUL	+1.38
UNCLEAR ____*____*____*_X___ CLEAR	+1.25

Fig.6.22. Assessing the value of the new system

VIEWS ON USING THE NEW SYSTEM -2 -1 0 +1 +2		AVERAGE
DIFFICULT TO LEARN	____*____*____*_X__ EASY TO LEARN	+1.38
DIFFICULT TO USE	____*____*____*_X__ EASY TO USE	+1.25
TROUBLESOME	____*___X*____*____ FEW PROBLEMS	- 0.13
TIME CONSUMING	____*_X__*____*____ TIME SAVING	- 0.81
IMPOSING	____*___X*____*____ UNIMPOSING	- 0.19
		+1.06

Fig.6.23. User views of the new system

differences between Figures 6.1 and 3.2 are exaggerated by the diagramming technique, there are differences in the analysis phase, through the separation of Körner and internal process analysis, and in the explicit separation of function design from data design. Other phases ran true to Multiview, such as the design of the human-computer interface, and in particular the roles of the users and facilitators and the contribution of prototyping. The willingness of users to participate, and their enthusiasm to try out new tools and techniques, despite their busy schedules, also supported the Multiview approach.

Chapter 7
The Academic Department Case

This case concerns the development of an information system for the Computer Science Department of Aston University. Further details of the application can be found in Avison (1990a). The action learning aspects are explored in Avison (1989). The project has the long term aim of fulfilling the information requirements of staff and students of the department. It should also integrate with other systems of the university. There is a university steering group deciding on the university's strategy for management information systems over a period of ten years. The author is also a member of that group. The departmental information system (DIS) should integrate with this system, but the whole application will take many years to develop fully.

The department has grown considerably over the last few years, and it is of the size where communications become more difficult and the sheer weight of administrative work makes control a particular problem. But it is essential to meet target dates, such as those stated in conference 'calls for papers' or research council 'calls for proposals', and to meet examination schedules. It is also necessary to control activities in the department, such as the selection, purchase, maintenance and use of equipment.

As the early draft of the rich picture shows (Figure 7.1), there are a number of areas that warrant investigation in the problem situation. The emphasis in this diagram has been placed on the courses offered by the department. Figure 7.2 gives an alternative view of the department and emphasises the role of the head of department and senior tutor. The various intermediate rich pictures were later merged to form one which was generally acceptable. Rich pictures were also drawn for a number of areas of concern in the department, and these areas have been identified as:

- *Admissions*: The admissions process must handle enquiries, visits, student and staff presentations, ensure an adequate supply of booklets, and so on. A DIS should provide useful information such as the correlation between student visits and admissions, and statistics on the qualifications of applicants.
- *Lecturer/course/student*: This subsystem needs to provide up-to-date class lists, syllabuses, reading lists and tutor lists, draw up and provide printed timetables and monitor student assessment of courses. The fact that this information is held for the community will help to ensure that the latest versions are always readily available, and there is no confusion about which is the latest version.
- *Examinations*: Keeping track of progress in following the examination schedule

through the processes of examination paper design, writing questions, typing and checking, sending to external examiners, and holding boards of examiners meetings, is complex, and failure to keep to timetables causes many problems. This subsystem is likely to inter-relate with the electronic notice-board and departmental diary subsystems.

- *Notice board*: An electronic notice board and diary can be used to hold details of department meetings, such as seminars, committee and staff meetings, and give dates related to calls for papers and research proposals. This would help to ease the process of arranging meetings when people are free.

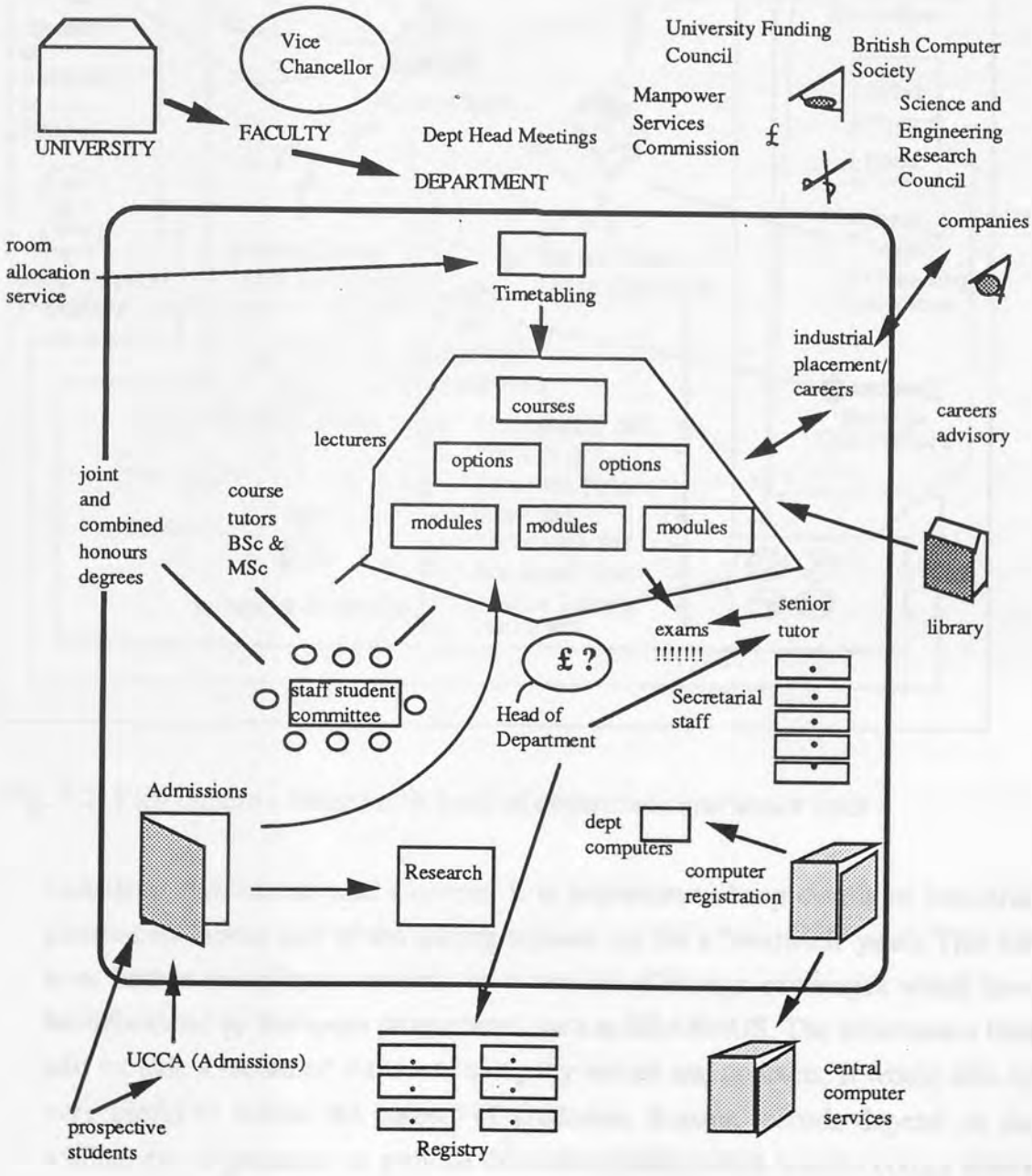


Fig. 7.1. Rich picture of academic department

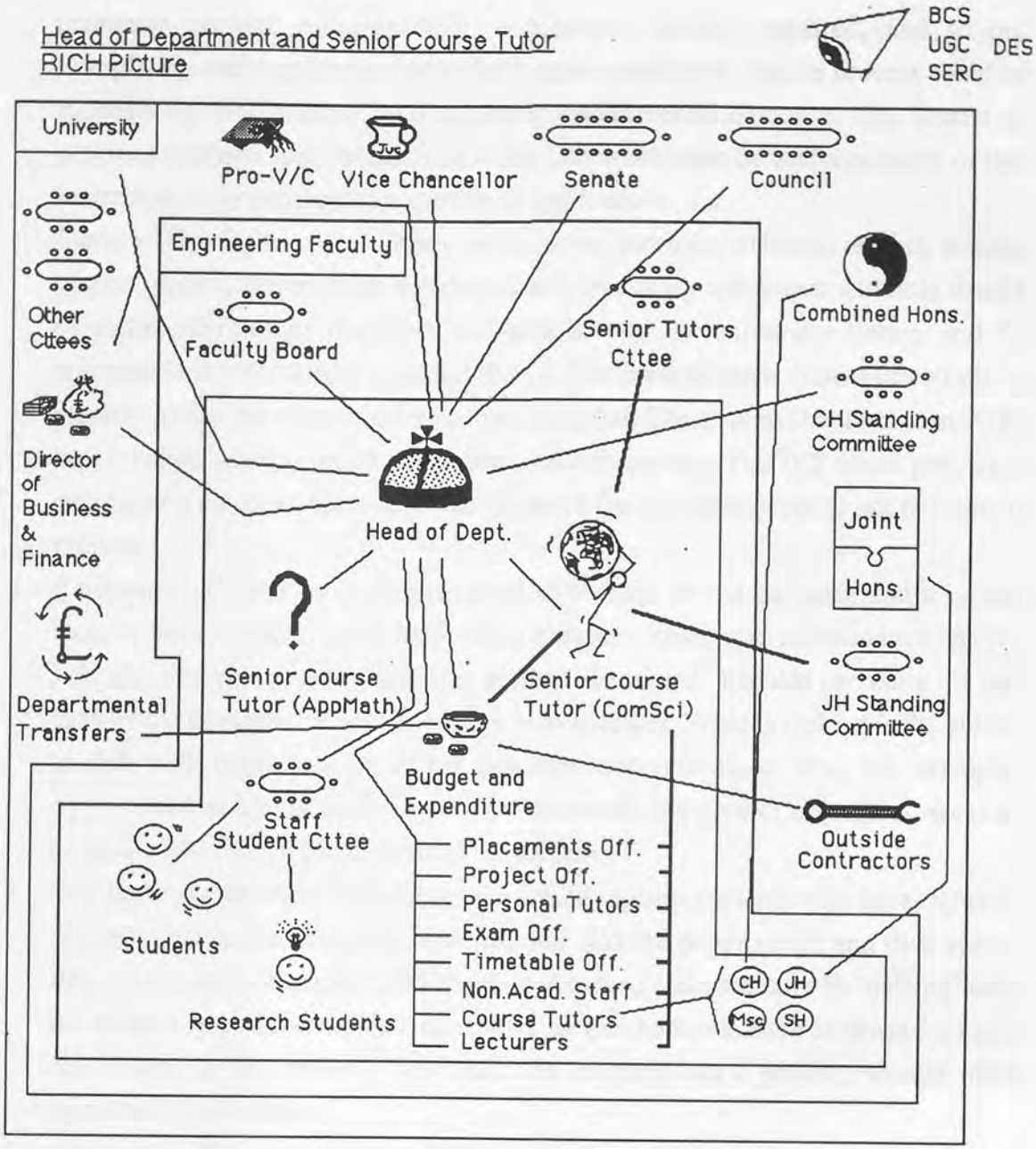


Fig. 7.2. Rich picture - concerns of head of department and senior tutor

- *Industrial Placements and Careers*: It is important to keep details of industrial placements (about half of the undergraduates opt for a 'sandwich' year). This has been further complicated recently by a number of foreign exchanges which have been financed by European programmes such as ERASMUS. The information held will include a record of visits and company names and contacts. It would also be very useful to follow the careers of graduates. Present records depend on the willingness of graduates to provide this information (which usually comes with a request for a reference) and it would be helpful to formalise this process.
- *Research*: Many calls for research proposals require detailed information about the

proposed project, research staff, equipment, finance required, and so on. Completing each application individually takes much time, but the process could be speeded up considerably with a general departmental database. The details of previous research applications held in the DIS would also be useful as much of this material may be common to a number of applications.

- *Library*: The departmental library holds books, journals, technical reports, student project reports, dissertations and theses, and the library subsystem will hold details of stock, borrowings recorded, and will link to the university library and its computerised information system (GEAC). The present paper system often fails to identify easily the whereabouts of some material. The system should also manage an exchange scheme which exists between universities. The DIS could provide a satisfactory solution, for a full-time librarian for the department is not feasible at present.
- *Equipment*: Details of equipment need to be held on the database and this will include their 'owners', place held, serial numbers, costs, and maintenance record. It is true that these details, like many others discussed, are held presently, in this case by the computer officer given this responsibility. What is lacking is the ability to link with other aspects of the departmental system, so that, for example, equipment loaned can be tied in with coursework and project timetables, so as to make the best overall use of available equipment.
- *Computer registration*: This subsystem would register students who have rights of access to particular computer systems, and allocate paper usage and disk space. Registration and de-registration could be achieved automatically by 'liaising' with the student records on the database. Only special requests, such as those for extra disk space or for use of a new machine to carry out a project, should need significant intervention.
- *Accounting*: This subsystem could detail the financial incomings and outgoings of the department, keeping track of funds for conferences and travel, equipment, stationery, fees for seminars, and so on.
- *Decision support system*: This system would facilitate access to particular individual subsystems to deal with ad hoc queries and also facilitate (to appropriate users) the provision of information which might necessitate source data coming from a number of subsystems.

The overall system is to produce standard reports for regular output, provide statistical analyses, and handle ad-hoc queries.

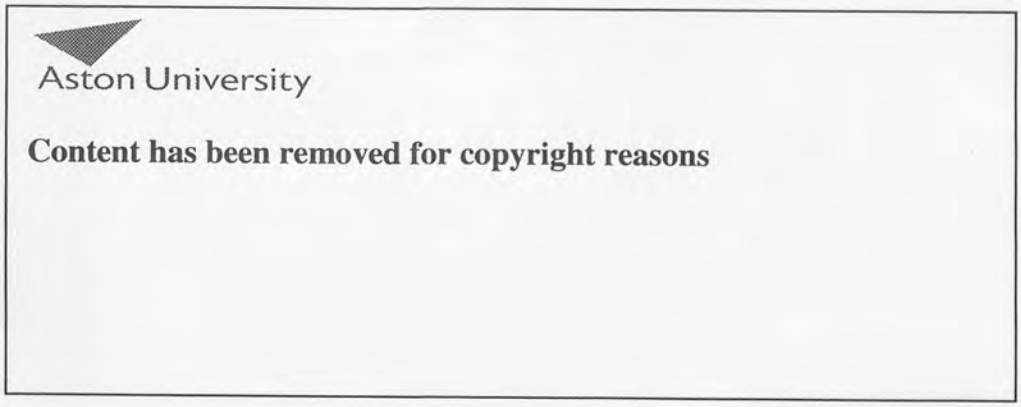
The project has mainly been carried out by students. Each group of students has tended either to concentrate on one of the above areas (either starting from scratch or carrying on from where another group finished) or to help other groups in that year by

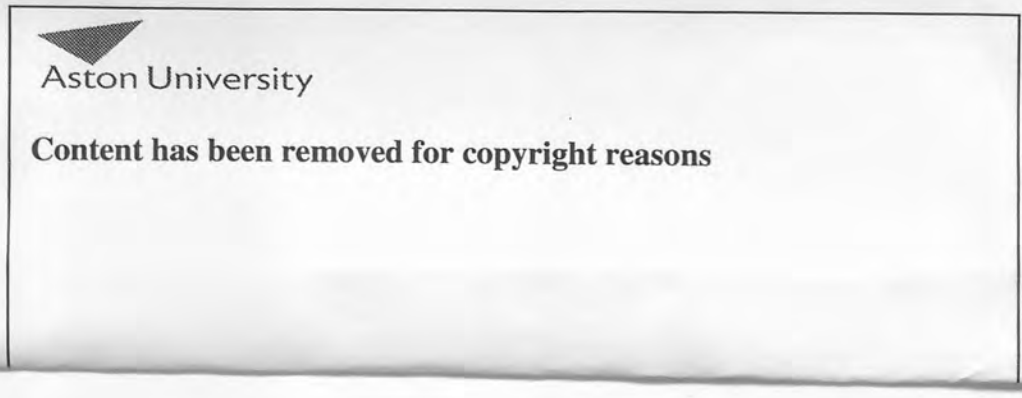
updating the overall models (such as the entity model as new entities were discovered) and providing standards for menu layouts and other screen designs.

The entity model, shown as Figure 7.3, shows the scale of the overall project. The number of entity types approaches one hundred. Each group working on a particular area used a subset of this diagram. Figure 7.4 shows the entity model for the area 'students and courses'.

Different groups of students have been working on prototypes and the project has been progressing over a period of four years. Many of the diagrams presented in this chapter have been taken from student reports over this period. A computer officer has already been appointed full-time whose role is to convert these prototypes into an integrated operational system.

The details of the entities found on Figure 7.4 are shown below. In retrospect, the attribute names should have been more helpful. We were limited to eight characters on one of the database management systems used.





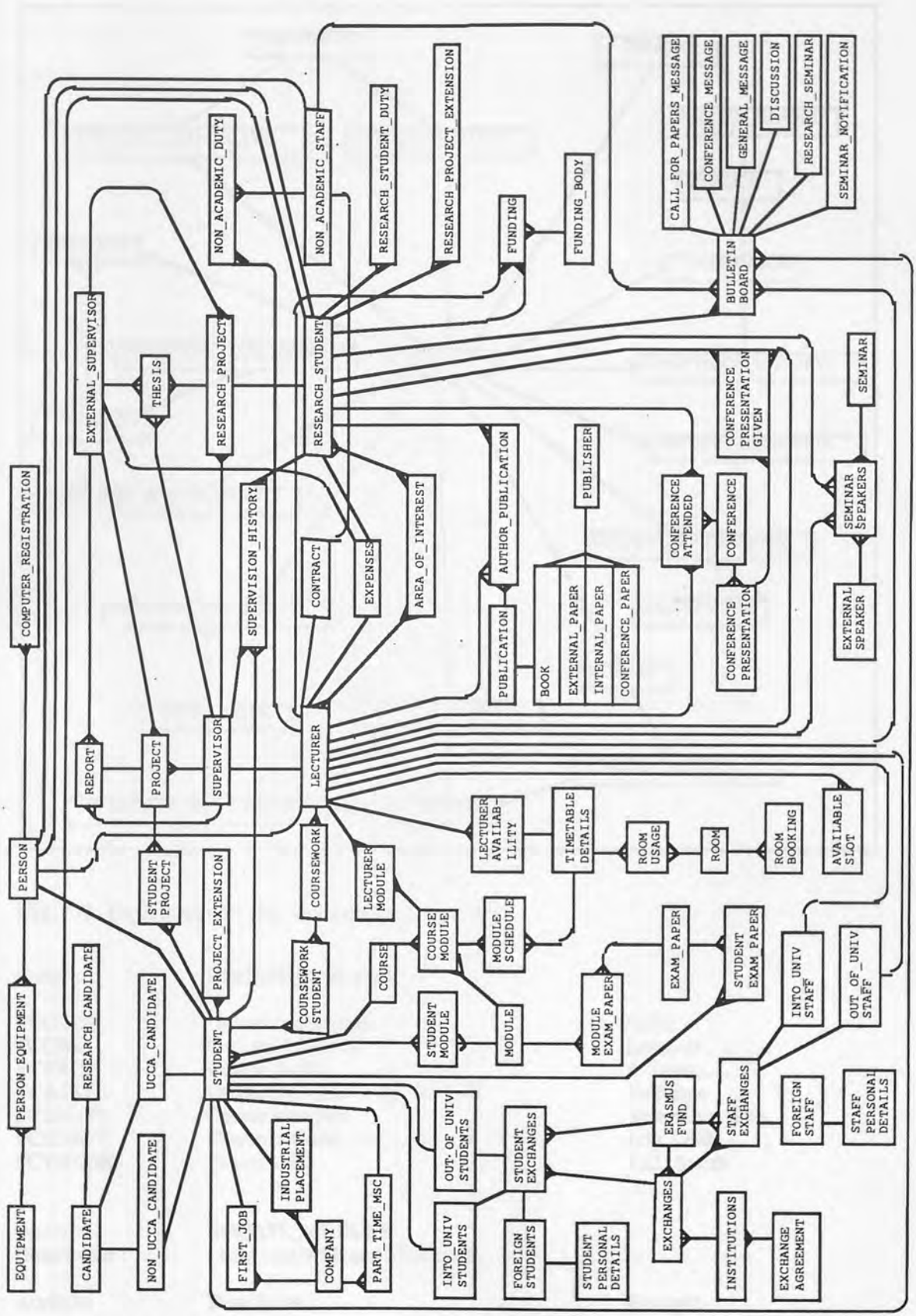


Fig. 7.3. Entity model of the academic department

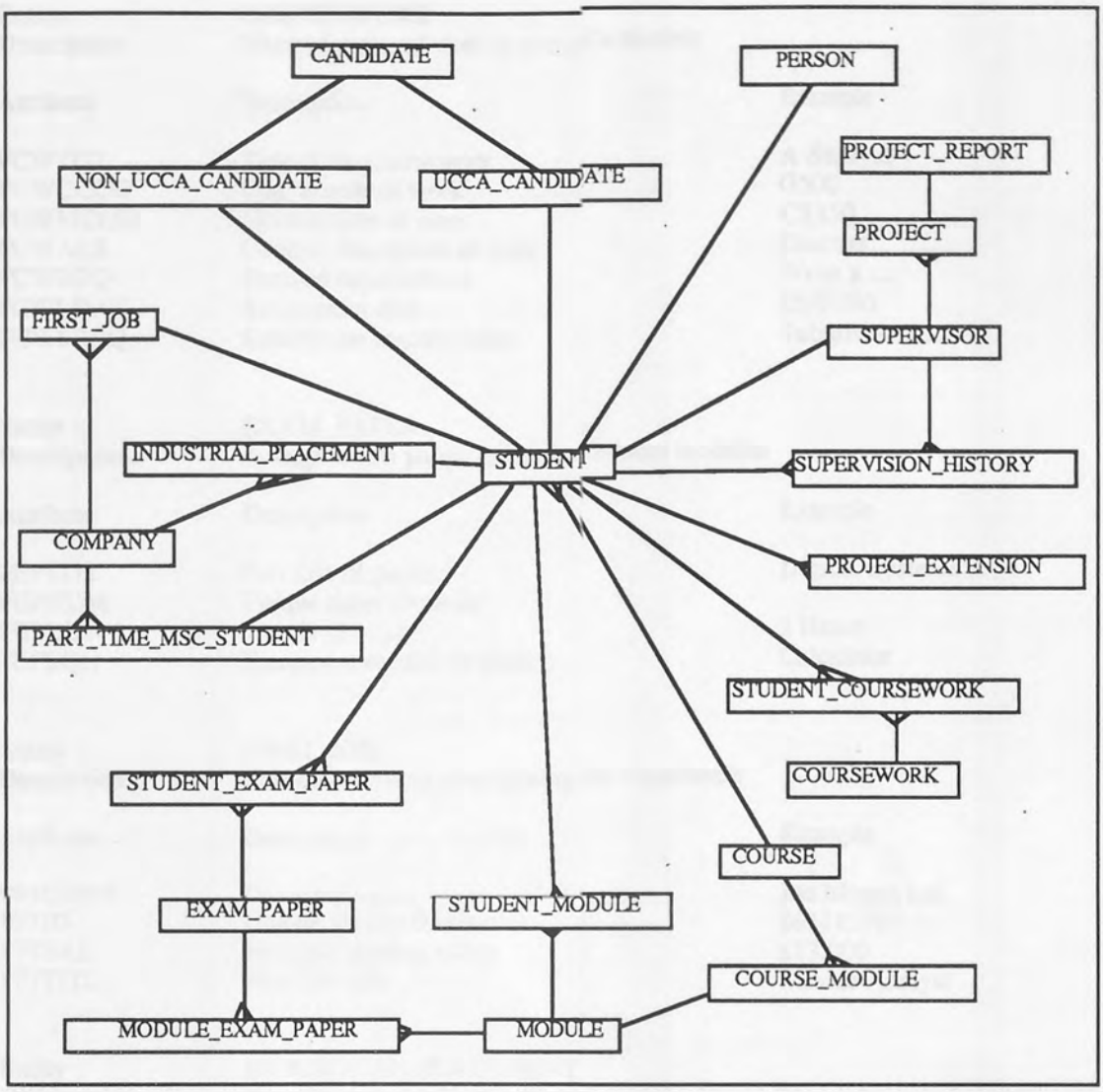
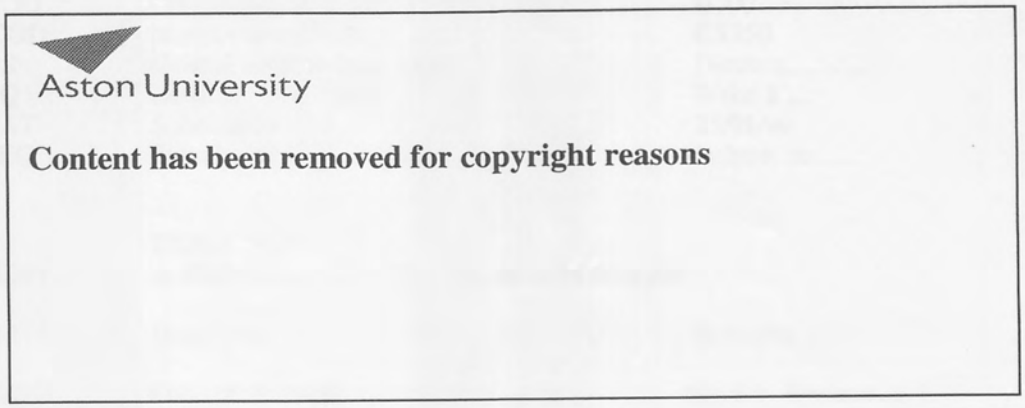
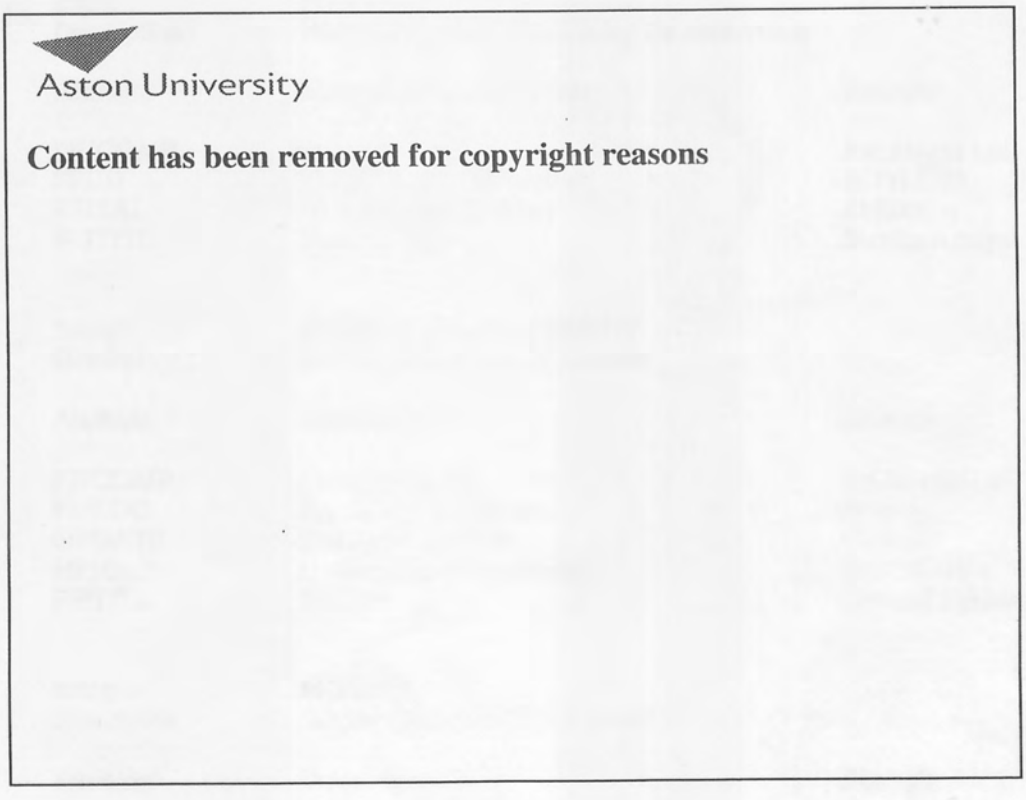


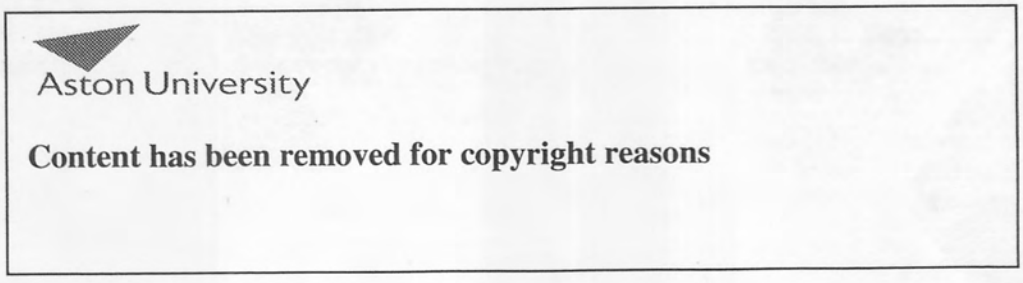
Fig.7.4. Entity model for students and courses

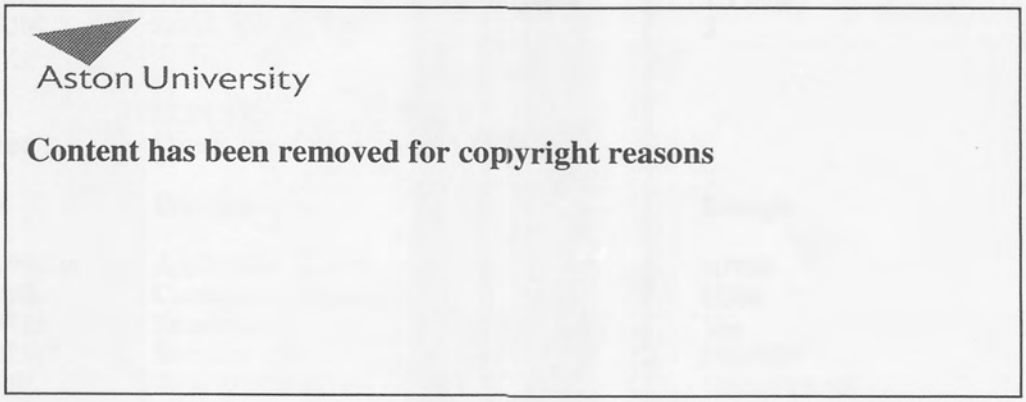
Aston University

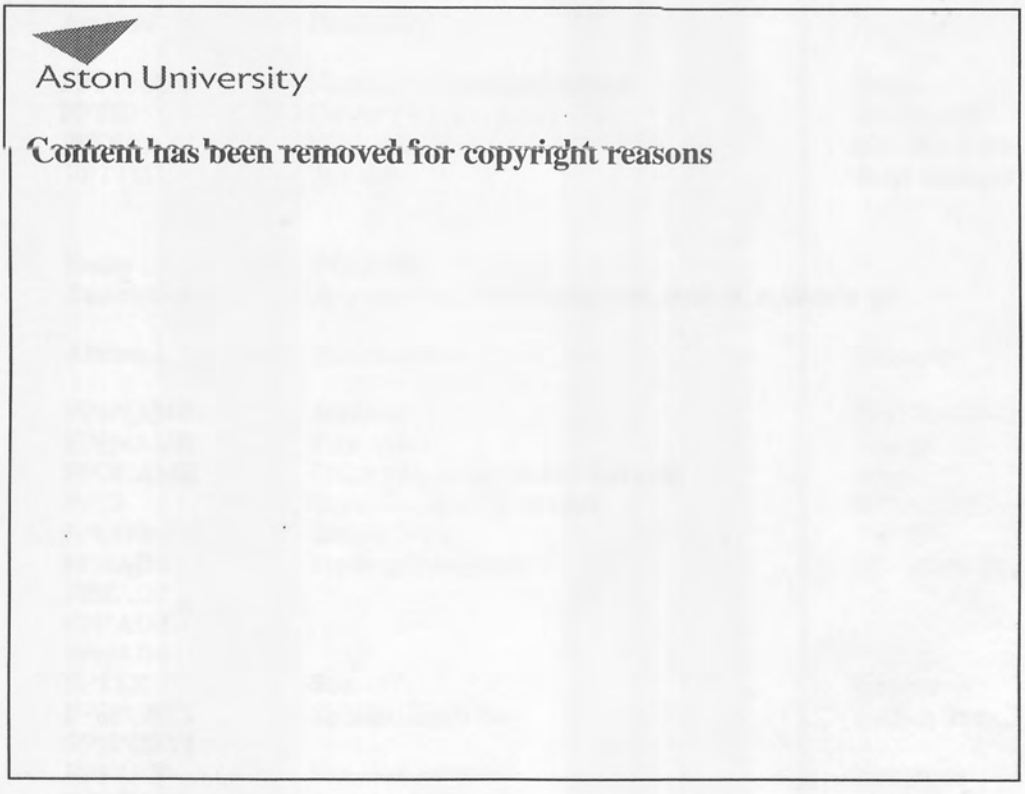
Content has been removed for copyright reasons

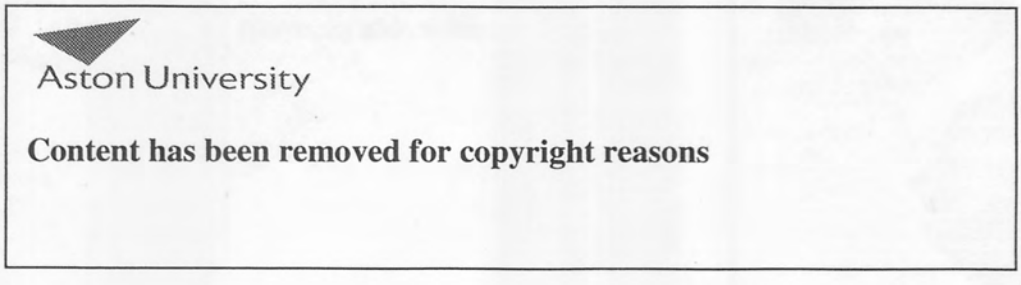

Aston University
Content has been removed for copyright reasons

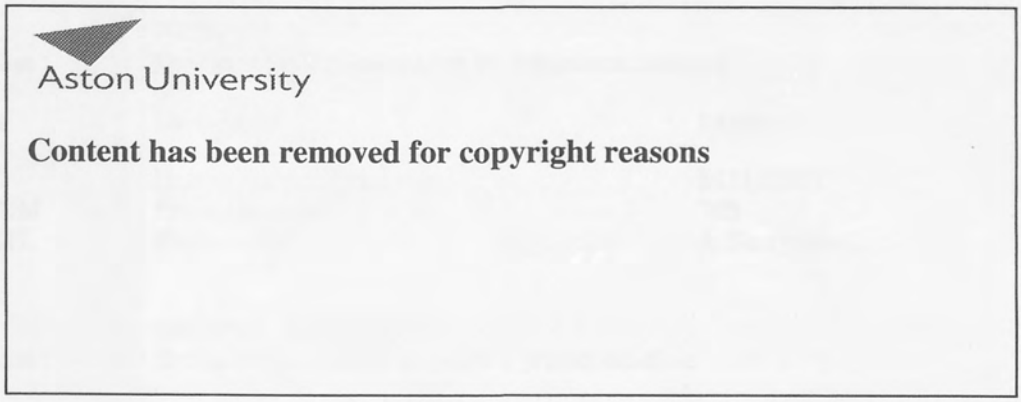

Aston University
Content has been removed for copyright reasons

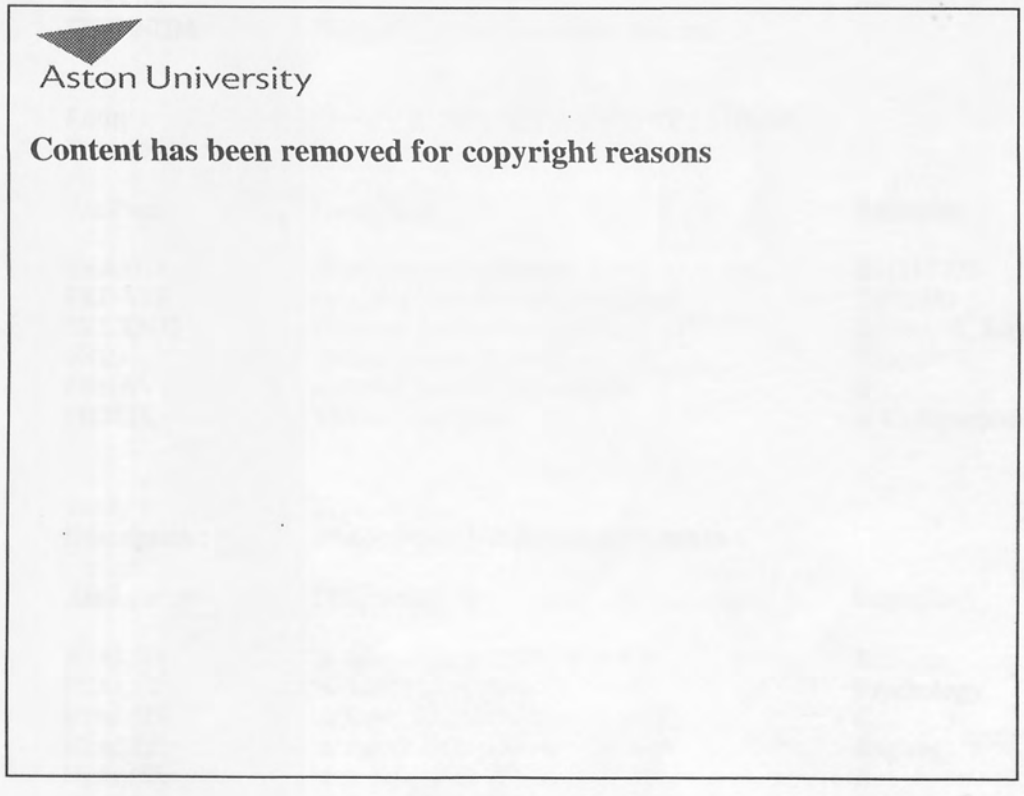

Aston University
Content has been removed for copyright reasons

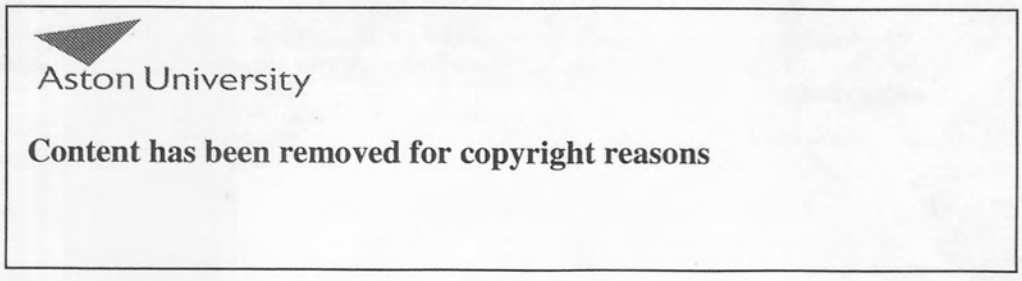

Aston University
Content has been removed for copyright reasons

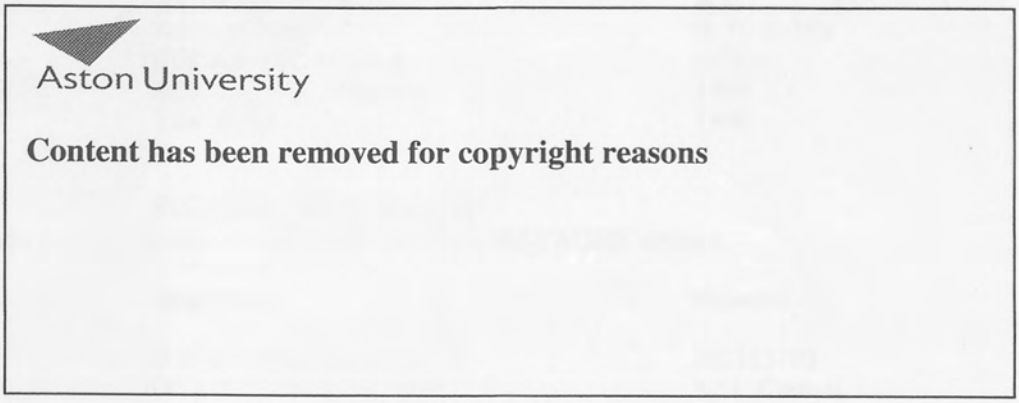

Aston University
Content has been removed for copyright reasons

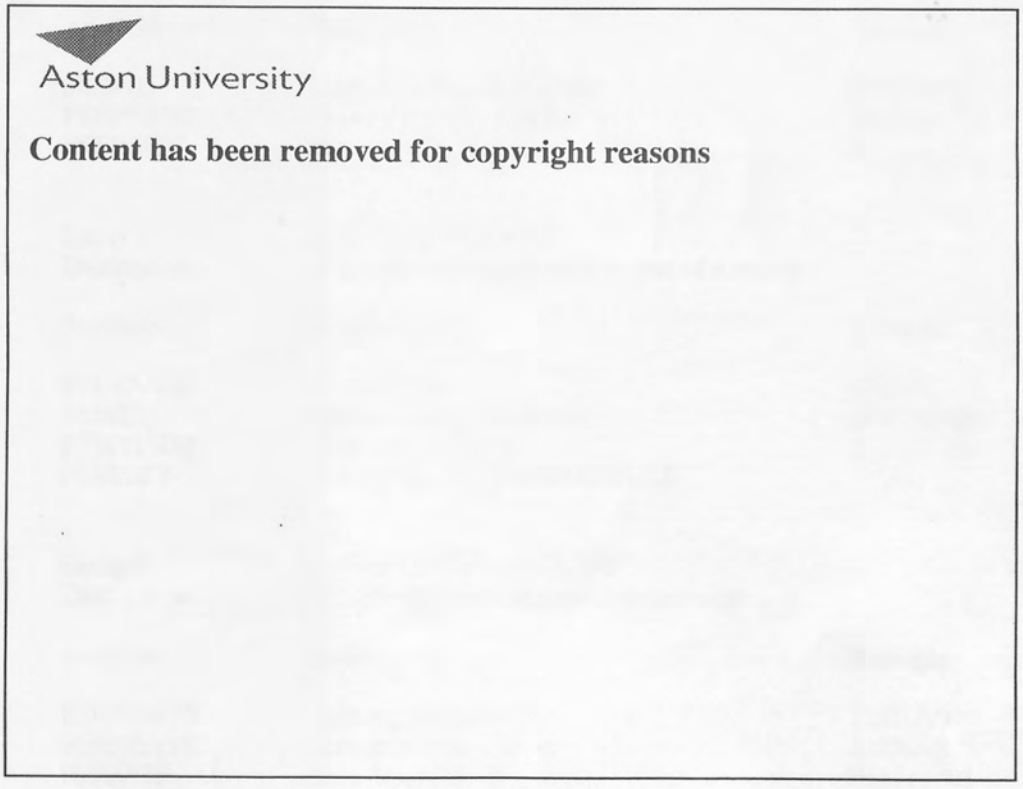

Aston University
Content has been removed for copyright reasons

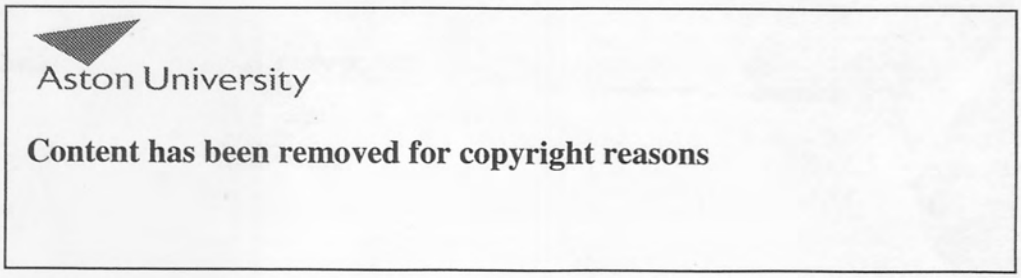

Aston University
Content has been removed for copyright reasons


Aston University
Content has been removed for copyright reasons


Aston University
Content has been removed for copyright reasons


Aston University
Content has been removed for copyright reasons


Aston University
Content has been removed for copyright reasons


Aston University
Content has been removed for copyright reasons

In developing the information system, the co-operation of colleagues was and still is required, and the prospective carrot of an information system to help administrative work, although neither guaranteed nor short-term, proved tempting to academic and non-academic staff alike. Colleagues agreed to help both the academic staff leading the project and the students requiring advice when investigating the application area. Such co-operation was agreed informally, through discussions, and formally, in a departmental staff meeting.

This promise of help has been fulfilled. One colleague even 'played' a non-helpful role - something that was not agreed beforehand, but happened 'naturally' - and this proved useful to students who could experience and react to this situation. This colleague had developed his own system which worked and he did not want the application to be duplicated in the DIS. This may not be an unsatisfactory situation, unless he leaves the department.

In general, those colleagues teaching information systems and other teachers of computer science, administrative staff, and technicians have proved willing to be interviewed, participate in the prototype development, use the prototypes, comment on the design, attend formal presentations, and help in the assessment process (both of the prototypes themselves and of the students' achievements and performance).

The students take the course as an option on the final year BSc Computing Science and a joint course in Management and Computer Science. They have already had courses in information systems and databases. Warnings of hard work and many difficulties ahead reduce their number so that about half the students take the option. Nevertheless, the course has gained popularity over the years. Groups meet frequently to ensure that their efforts integrate successfully. Students on the course tend to be highly motivated.

For the students, the course is very different from their other courses which consist of lectures, tutorials, laboratory work and written examinations. In this course, the students work on a real problem, in groups, and are assessed using the project report, a demonstration of their prototype and a verbal presentation. Students on this 'action learning' course have gained experience in:

- Interviewing techniques.
- Group work and inter-group work.
- Project control.
- Presenting seminars.
- Training users.
- Report writing and other documentation techniques.
- Understanding roles, such as that of the database administrator.
- Using and evaluating contrasting methodologies, techniques and tools used in information systems work.
- Producing part of a working information system.

Figures 7.5 to 7.9 show the set of structure charts for the processing of UCCA applications. These are applicants for undergraduate courses who apply through the standard channels. Figures 7.10 to 7.12 show the data flow diagrams that were constructed representing this processing.

Groups were asked to produce reports contrasting the use of different techniques and automatic aids in given situations. As well as using different application packages, they used more general tools, such as application generators and documentation tools (including those that help draw data flow diagrams) and also tools designed to support particular information systems development methodologies. A few groups have followed different methodologies, but most chose Multiview as the basis of their approach.

Students are also able to look at some of the wider implications of information systems work, such as the effect of the Law (for example, the Data Protection Act) and the ways people react (for example, to requests for co-operation, to different interviewing methods, and to change in general). Students made recommendations for the departmental information system concerning the roles of the database administrator (DBA) and security levels for the DBA, academic staff, non-academic staff, research students and students taking formal courses.

General consensus between student groups working on the project is required for much of the planning and development stages. Agreement on the establishment of the goals of the overall system and common standards, such as naming conventions and screen designs, is essential. However, groups of students worked reasonably independently when modelling 'their' part of the department, mapping this onto a database and developing the prototypes.

The prototype has been designed for different types of user:

- Database administrator.
- Head of Department.
- Senior tutor.

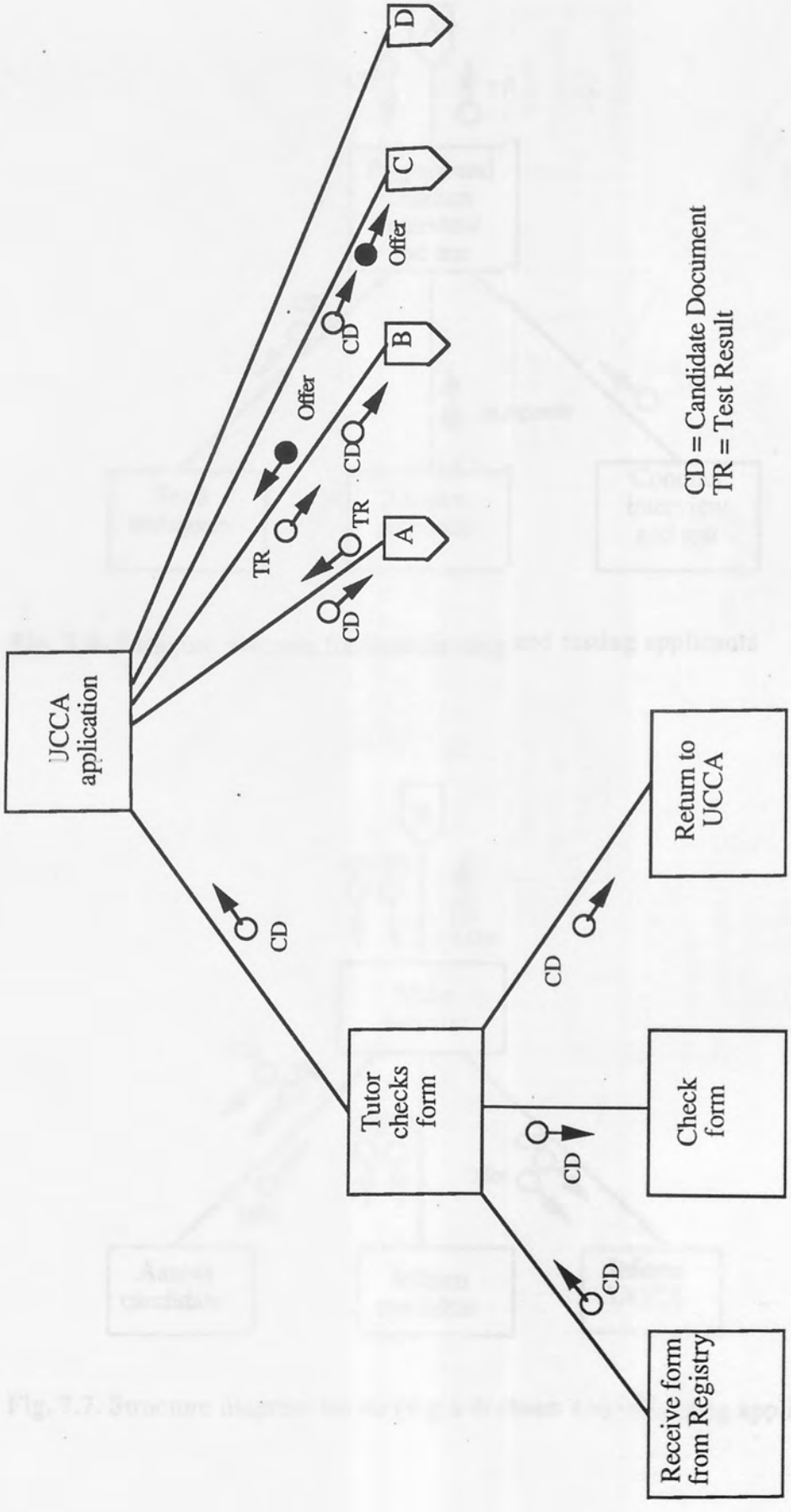


Fig.7.5. Top level structure chart for UCCA application

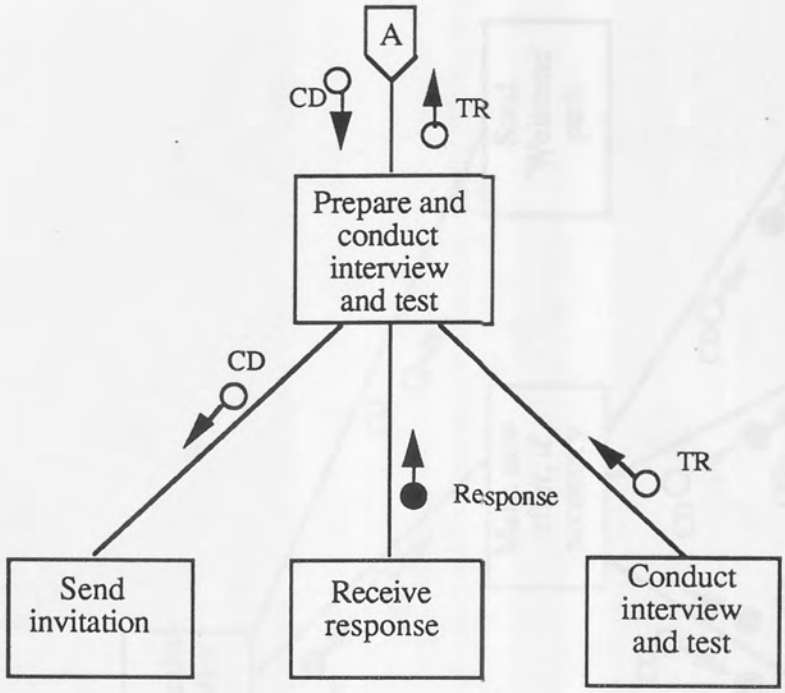


Fig. 7.6. Structure diagram for interviewing and testing applicants

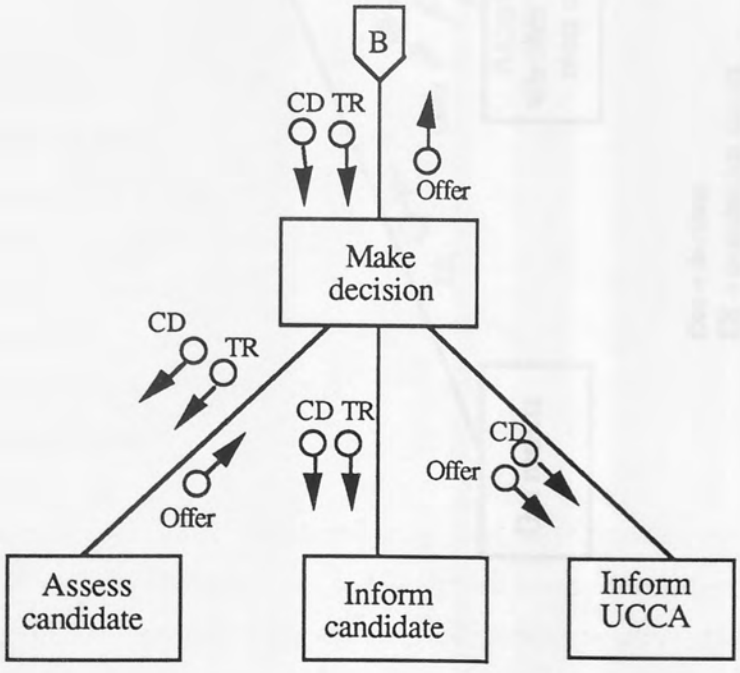


Fig. 7.7. Structure diagram for making a decision and informing applicants

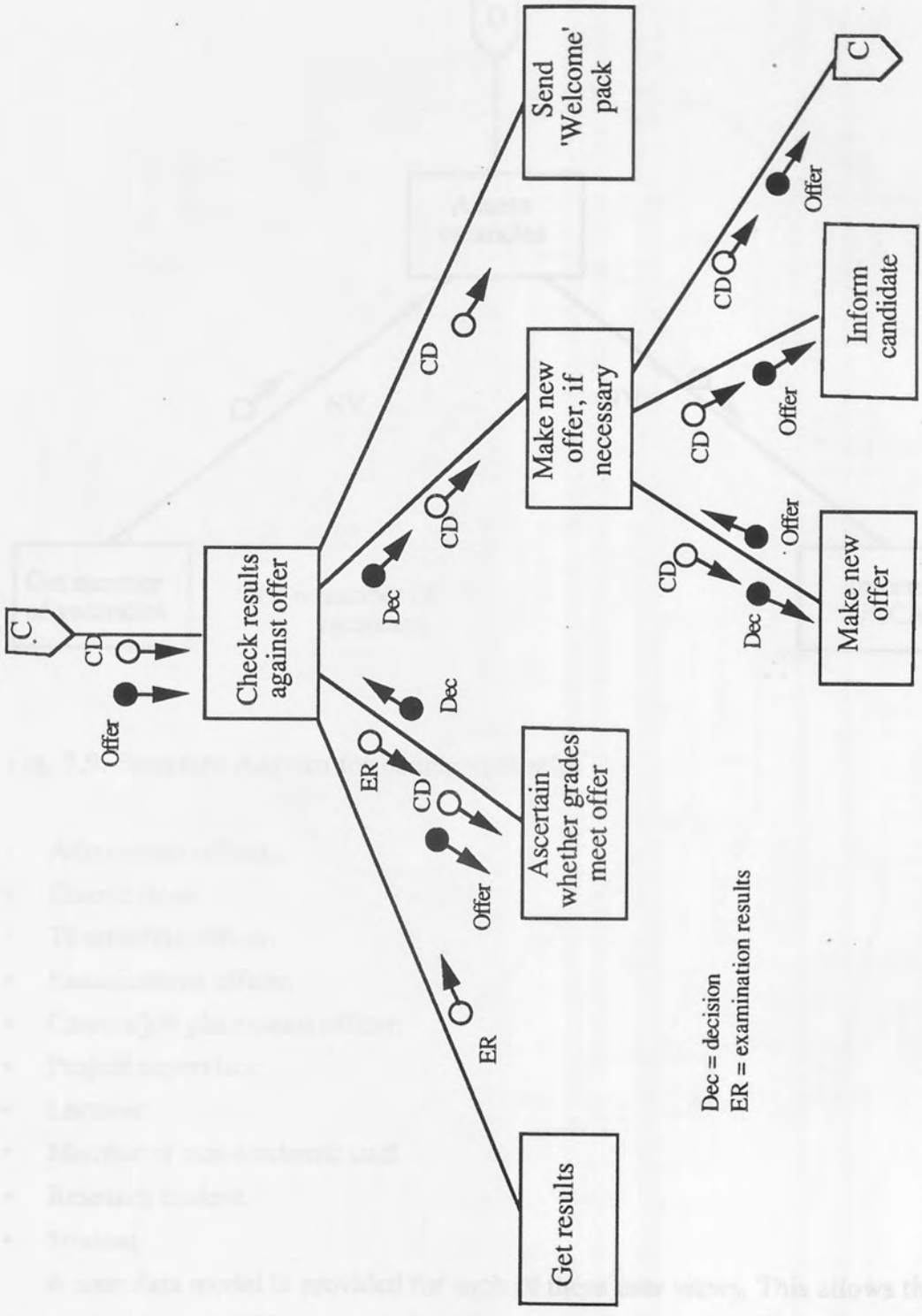


Fig.7.8. Structure diagram for examination results processing

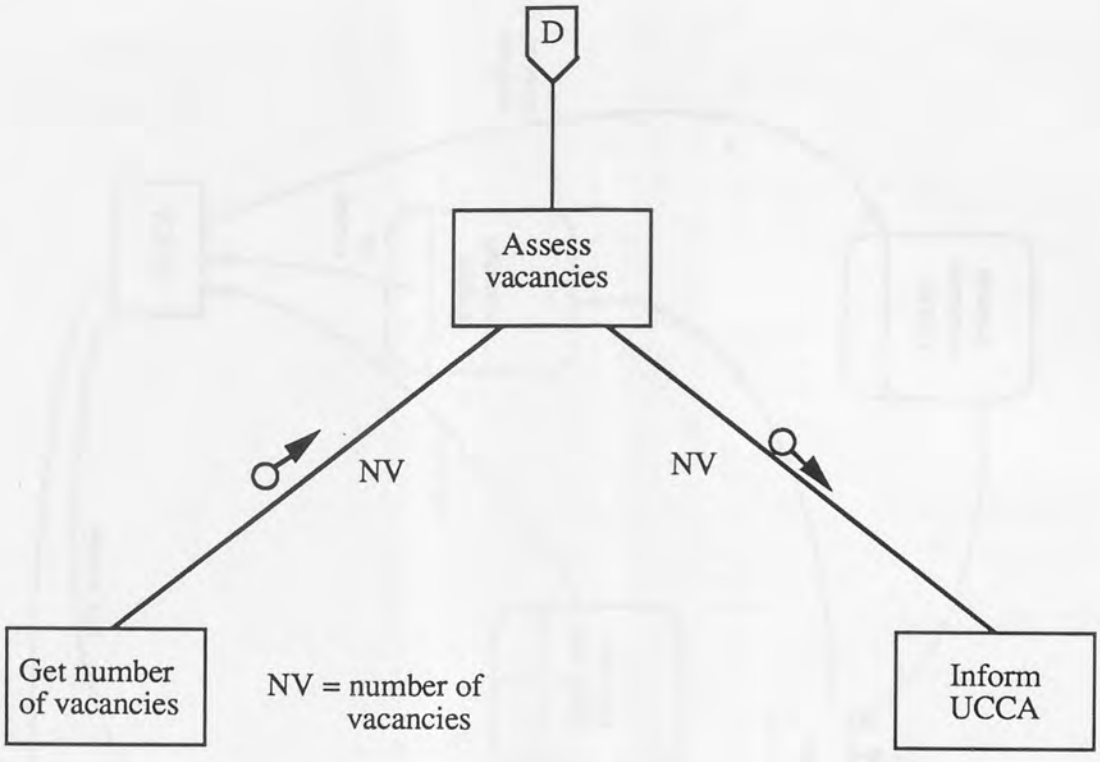


Fig. 7.9. Structure diagram for course vacancies

- Admissions officer.
- Course tutor.
- Timetabling officer.
- Examinations officer.
- Careers/job placements officer.
- Project supervisor.
- Lecturer.
- Member of non-academic staff.
- Research student.
- Student.

A user data model is provided for each of these user views. This allows the same data to be seen by different users in different ways; simplifies the user's perception; provides a certain amount of logical data independence when restructuring the database; and provides automatic security for personal data, as some data will not be available in a particular view. With the exception of the database administrator's view, which is the complete database, all these views are subsets of the data held in the DIS database.

An example user view is shown as Figure 7.13. It shows the data model for the admissions officers. There are three sub-views, those for the admissions officers for undergraduate courses, postgraduate courses and research students. The user model

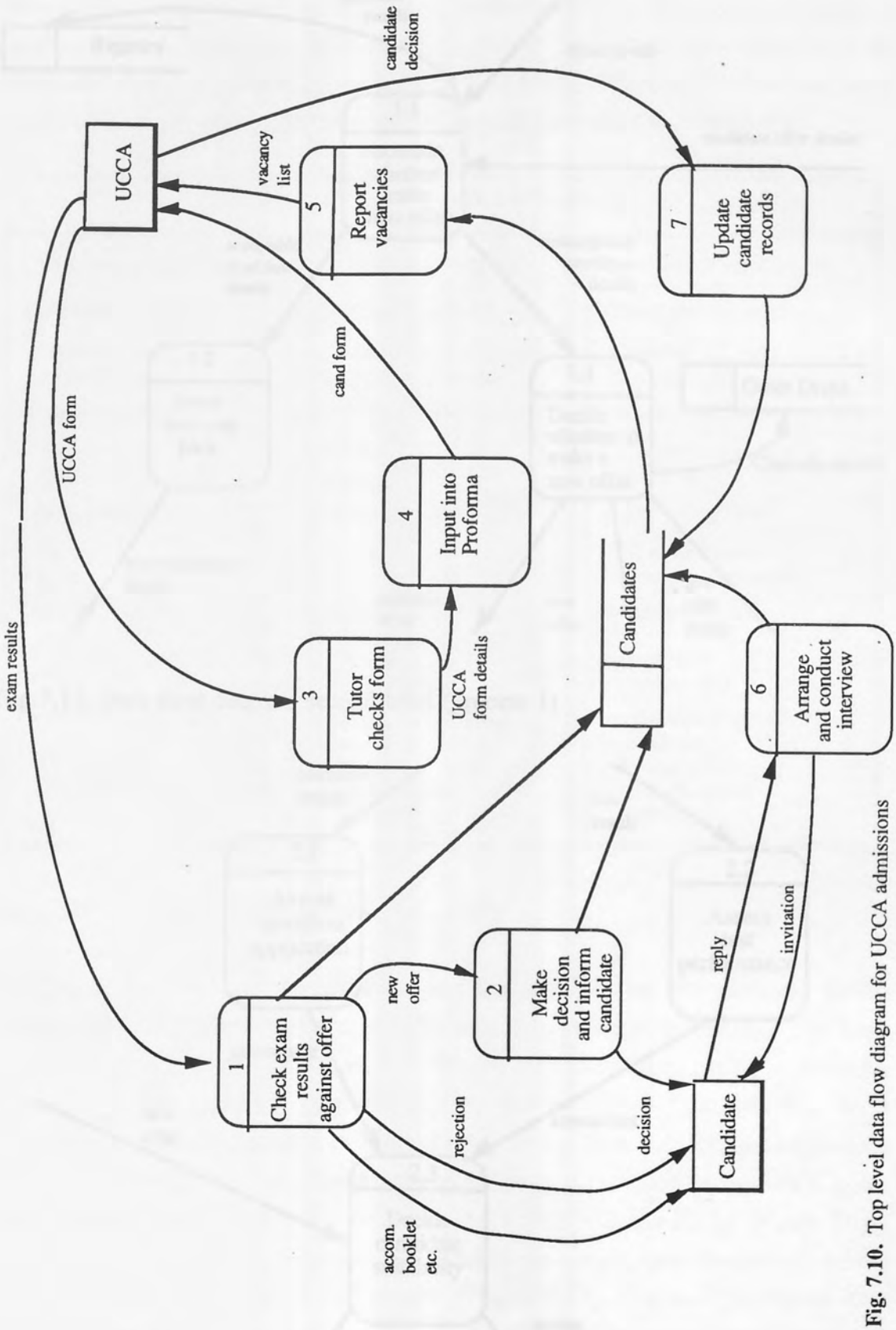


Fig. 7.10. Top level data flow diagram for UCCA admissions

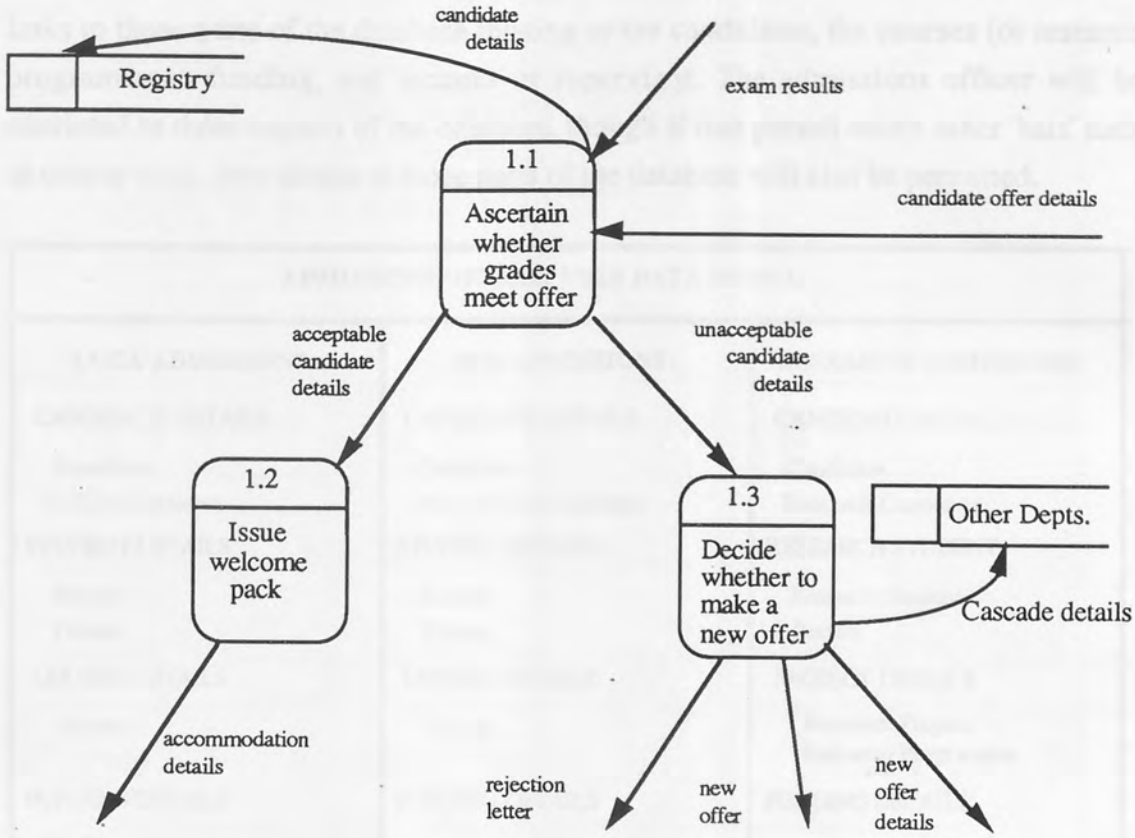


Fig.7.11. Data flow diagram second level (process 1)

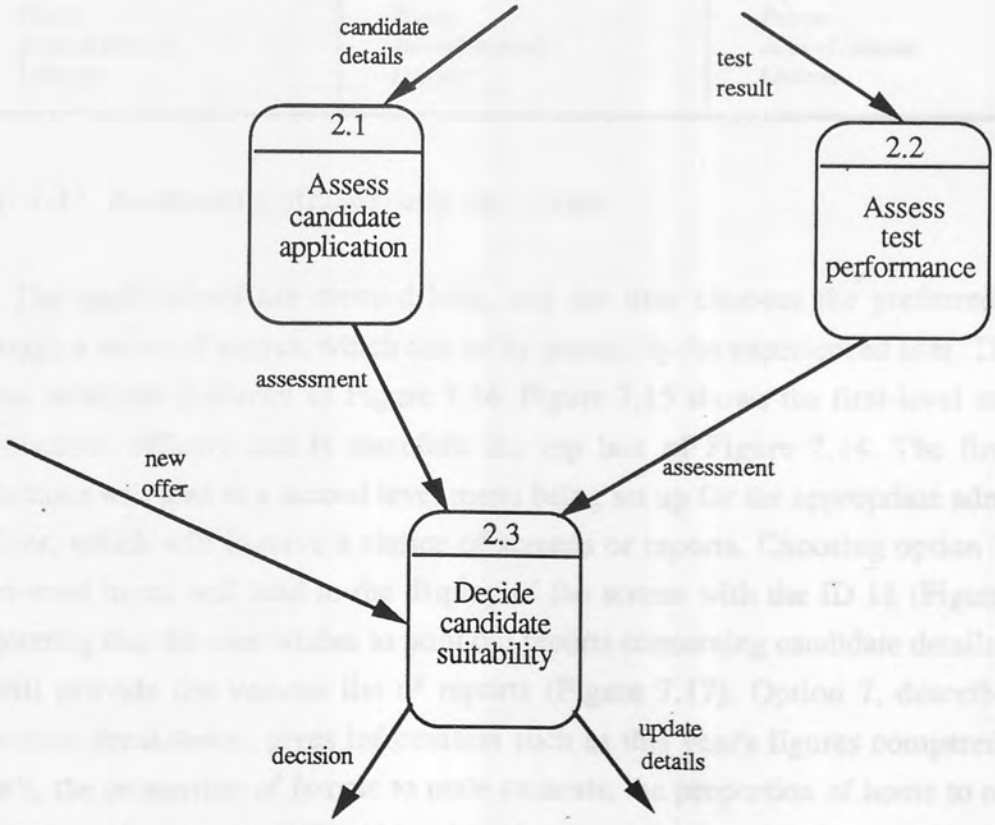


Fig.7.12. Data flow diagram second level (process 2)

links to those parts of the database relating to the candidates, the courses (or research programmes), funding, and lecturer or supervisor. The admissions officer will be restricted to those aspects of the database, though if that person wears other 'hats' such as course tutor, then access to those parts of the database will also be permitted.

ADMISSIONS OFFICER USER DATA MODEL		
UCCA ADMISSIONS	**MSc ADMISSIONS**	**RESEARCH ADMISSIONS**
CANDIDATE DETAILS	CANDIDATE DETAILS	CANDIDATE DETAILS
Candidate	Candidate	Candidate
UCCA-Candidate	Non-UCCA-Candidate	Research-Candidate
STUDENT DETAILS	STUDENT DETAILS	RESEARCH STUDENT
Student	Student	Research Student
Person	Person	Person
COURSE DETAILS	COURSE DETAILS	PROJECT DETAILS
Course	Course	Research Project
		Research Programme
FUNDING DETAILS	FUNDING DETAILS	FUNDING DETAILS
Funding	Funding	Funding
Funding body	Funding body	Funding body
SUPERVISOR DETAILS	SUPERVISOR DETAILS	SUPERVISOR DETAILS
Supervisor	Supervisor	Supervisor
Person	Person	Person
Area-of-interest	Area-of-interest	Area-of-interest
Lecturer	Lecturer	Lecturer

Fig. 7.13. Admissions officers' user data model

The applications are menu-driven, and the user chooses the preferred option through a series of menus, which can be by-passed by the experienced user. The basic menu structure is shown as Figure 7.14. Figure 7.15 shows the first-level menu for admissions officers and is therefore the top box of Figure 7.14. The first three selections will lead to a second level menu being set up for the appropriate admissions officer, which will involve a choice of screens or reports. Choosing option 1 in the first-level menu will lead to the display of the screen with the ID 18 (Figure 7.16). Assuming that the user wishes to print out reports concerning candidate details, option 2 will provide the various list of reports (Figure 7.17). Option 7, described as a 'statistics breakdown', gives information such as this year's figures compared to last year's, the proportion of female to male students, the proportion of home to overseas applicants, the average GCE 'A' level grade, the percentage of offers against the total number of applicants, and the percentage of offers which are accepted. Another option

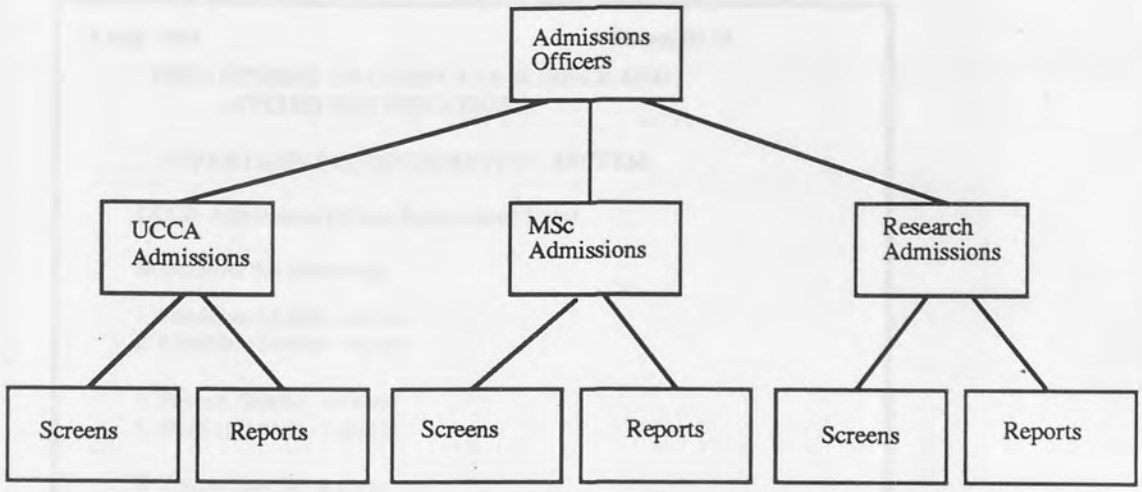


Fig. 7.14. Menu structure for admissions officers

is Standard Table Two (option 9). This is presented as Figure 7.18. The option name and report name are rather unhelpful, but it is the choice of the present admissions tutor and can, in any case, be changed easily.

15 July 1989 Screen ID 17

DEPARTMENT OF COMPUTER SCIENCE AND APPLIED MATHEMATICS

DEPARTMENTAL INFORMATION SYSTEM

Admissions Officer Applications Menu

Select from the following:

1 UCCA Admissions Officer: Admission Details

2 MSc Admissions Officer: Admission Details

3 Research Admissions Officer: Admission Details

H Help

R Return to previous menu

Q Quit

Enter your selection _

Fig. 7.15. The first-level menu for admissions officers

15 July 1989 Screen ID 18

DEPARTMENT OF COMPUTER SCIENCE AND APPLIED MATHEMATICS

DEPARTMENTAL INFORMATION SYSTEM

UCCA Admissions Officer Applications Menu

Select from the following:

1 Candidate Details - screens
2 Candidate Details - reports

3 Student Details - screens
4 Student Details - reports

5 Course Details - screens
6 Course Details - reports

7 Supervisor Details - screens
8 Supervisor Details - reports

H Help

R Return to previous menu

Q Quit

Enter your selection _

Fig. 7.16. A second-level menu for admissions officers

15 July 1989 Screen ID21

DEPARTMENT OF COMPUTER SCIENCE AND APPLIED MATHEMATICS

DEPARTMENTAL INFORMATION SYSTEM

UCCA Admissions Officer: Candidate Reports

Select from the following:

1 Overseas candidates list
2 EEC candidates list
3 Disabled candidates list
4 Acknowledgement letters
5 Offer letters
6 Address labels
7 Statistics breakdown
8 Standard table one
9 Standard table two

H Help

R Return to previous menu

Q Quit

Enter your selection _

Fig. 7.17. A third-level menu for admissions officers

STANDARD TABLE TWO

WEEKLY BREAKDOWN OF UCCA ADMISSIONS PROCEDURE

Week	Ends	Rec	Unc	Con	Rej	Pen	PAc	FAc	Enr
1	07/01/90	1		1				1	
2	14/01/90	1		1				1	
3	21/01/90	3	1	1	1		1	1	1
4	28/01/90	3	1	1	1		1	1	1
5	04/02/90	3	1	1	1		1	1	1
6	11/02/90	3	1	1	1		1	1	1
7	18/02/90	3	1	1	1		1	1	1
8	25/03/90	4	2	1	1		1	2	1
9	04/03/90	4	2	1	1		1	2	1
10	11/03/90	4	2	1	1		1	2	1
11	18/03/90	4	2	1	1		1	2	1
12	25/03/90	7	2	2	1	2	1	2	1

Rec = Applications received
Unc = Number of Unconditional Offers
Con = Number of Conditional Offers
Rej = Number of Rejections
Pen = Number of Pending Offers
PAc = Number of Provisional Acceptances
FAc = Number of Firm Acceptances
Enr = Number of Candidates Enrolled

(These figures are cumulative.)

Fig. 7.18. Standard table two

The soft copy form for inputting and updating basic student information is shown as Figure 7.19. It shows the screen for adding, updating and deleting data relating to students (option 3 of the menu in Figure 7.16). The menu for the actions in this case is placed at the bottom of the screen. The user has to key-in the first letter of the options: thus A for Add, and so on. This screen is designed for experienced users. They will be used to completing these forms and do not need detailed prompting (though there is an 'help' option which shows the user what to do if required).

The data dictionary is used to describe every component of the database; locate the data areas; verify that requested modifications to the descriptions are permitted; co-ordinate restructuring tasks that must be performed on the database; control all access to the database; catalogue back-ups; and keep track of users and programs that access the database.

Security in the DIS is important, particularly as the database will contain personal information, and the system has a number of security features. Only registered users can create or use a file. Users are assigned an ID and password and will be registered

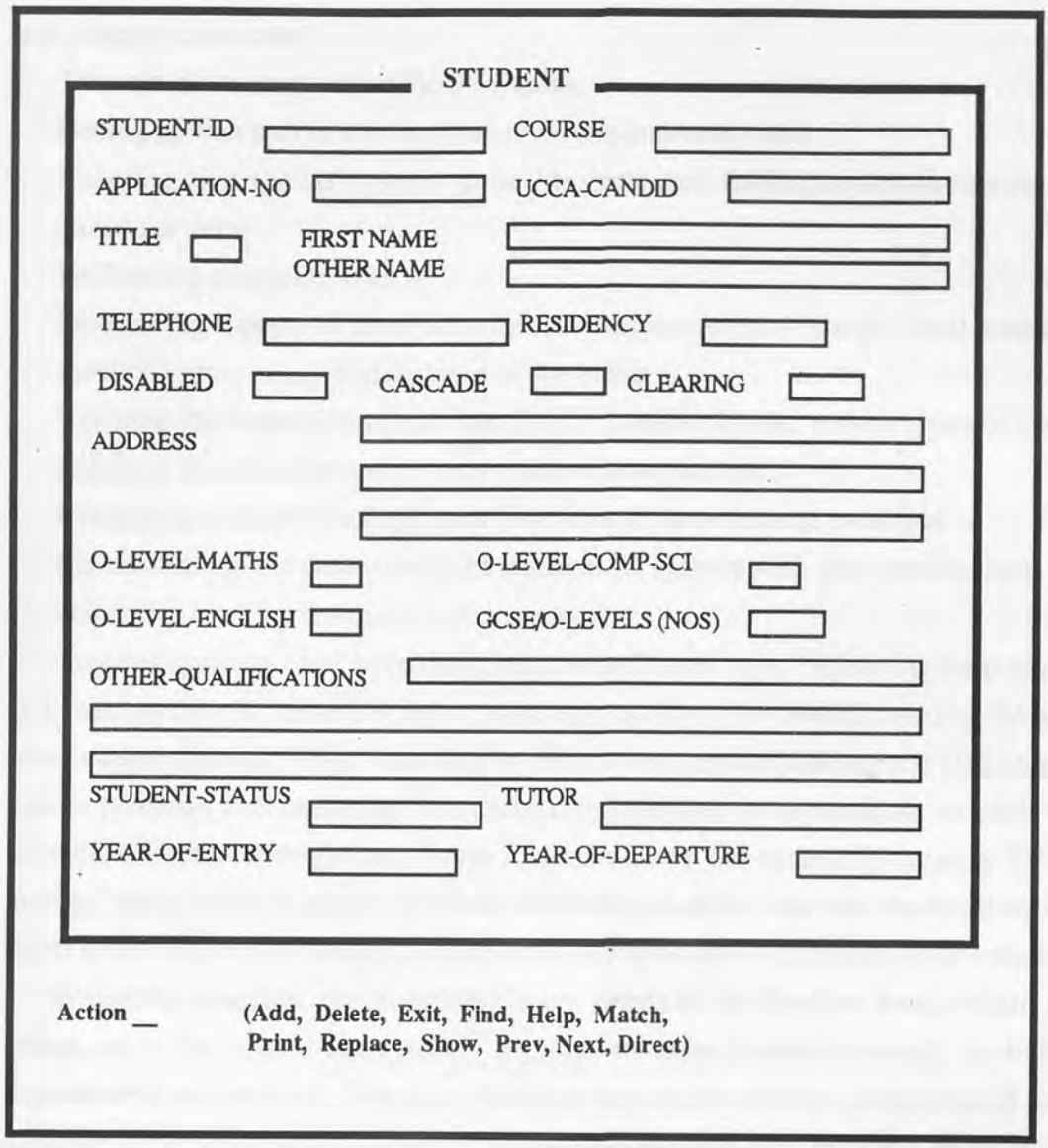


Fig. 7.19. VDU form for adding, updating or changing data relating to students

with certain 'signon' privileges (for example, to read, create, change or delete a file) and a priority that defines the way in which they can use the system. At the user data model level, privileges that can be granted to an individual user include exclusive use of a file; the right to 'unhold' a record held by another user; rights to hold, delete, modify or add a record; and rights to modify the value of an individual data element. Privacy directly relates user access permission with the access constraints assigned to record types, view types and data elements. Privacy codes are assigned to record types and elements, and view types and elements. Tests compare these codes with user read and write codes, which are assigned by the database administrator to a particular user when he or she is authorised to use a model. When a database is shared by a number of users, it is also essential to maintain the integrity of the data. Again, it is not appropriate here to discuss this in detail, but the system provides a wide range of validity checks

and integrity constraints.

The role of the computer officer includes:

- Setting up that part of the database to be implemented next.
- Ensuring that the subsystem to be implemented fulfils the requirements of the particular users.
- Performing a training role.
- Fine-tuning aspects of the system already implemented to ensure that it continues to meet changing usage and demand of the users.
- Ensuring the human-computer interface is suitable for the various types of users.
- Ensuring that security and privacy requirements are met.
- Evaluating software packages and hardware as they become available.
- Co-ordinating the departmental information system with the development of the university management information system.

Tutorial systems have been designed for each user type, improving help facilities, and making the menus and screens more appropriate - in general making the system more usable (Eason, 1984). Usability in this context means making the DIS as easy to use as possible and ensuring that each user interface is appropriate to each of the thirteen different user groups. Users can now enter the system by typing 'DIS' and thereby going straight to the top menu (assuming that the user has the required access rights), whereas early versions required the user to follow a complex set of commands.

Wherever possible, the interface design needs to be flexible, even within a user group, as a naive user can quickly migrate to casual expert through to being an experienced professional. The extra facilities required to enable inexperienced users to navigate their way round a system will be time-wasting and irritating to professional users.

Although the prototype, which has been most recently developed on a large computer using a large and complex software package, has shown that it is possible to set up a DIS using a database system which will fulfil many of the information needs of staff and students and ease the administrative load on staff, there are many question marks in the prototype relating to ease of use, ease of learning, flexibility and speed of developing applications. We are constantly looking at other possible directions, such as the provision of an icon interface on a microcomputer which is linked to the mainframe. Here all accesses relating to data will be intercepted by the interface software:

- Receiving data requests from the user.
- Processing those requests.
- Transferring the requests to the mainframe system, executing the request and transferring the results back to the user.
- Displaying the results in a suitable screen format.

The most likely direction is to implement the database on the mainframe and also link this with the microcomputer version of the database management system which will be implemented on the departmental network of microcomputers. This has a number of advantages:

- The use of the familiar icon interface, which the users prefer.
- The availability of a number of tools which will speed up the development of some applications, in particular reports and screens.
- Connections with the departmental network and the University local area network.
- Easy transfer of information to other microcomputer packages for viewing in a number of ways, including graphically, and for data manipulation.

Figure 7.20 shows a screen used as part of a prototype system for student feedback, using Hypercard on the Macintosh. The use of buttons (for the various options) and icons (for help and return), gives a screen which seems a considerable improvement on the conventional menus of Figures 7.15 and 7.16. The input document shown as Figure 7.21 for recording students' general comments, also seems an improvement on the type of form illustrated as Figure 7.19. The up and down arrows for each window allow the user to extend the amount of text held beyond the actual window size. Thus the user could 'scroll' up (using the up arrow) to go to the top of the message or downwards using the down arrow. We have also been experimenting with multi-media systems, and the students' photographs are scanned and are included for display in each student's computer record (Figure 7.22).

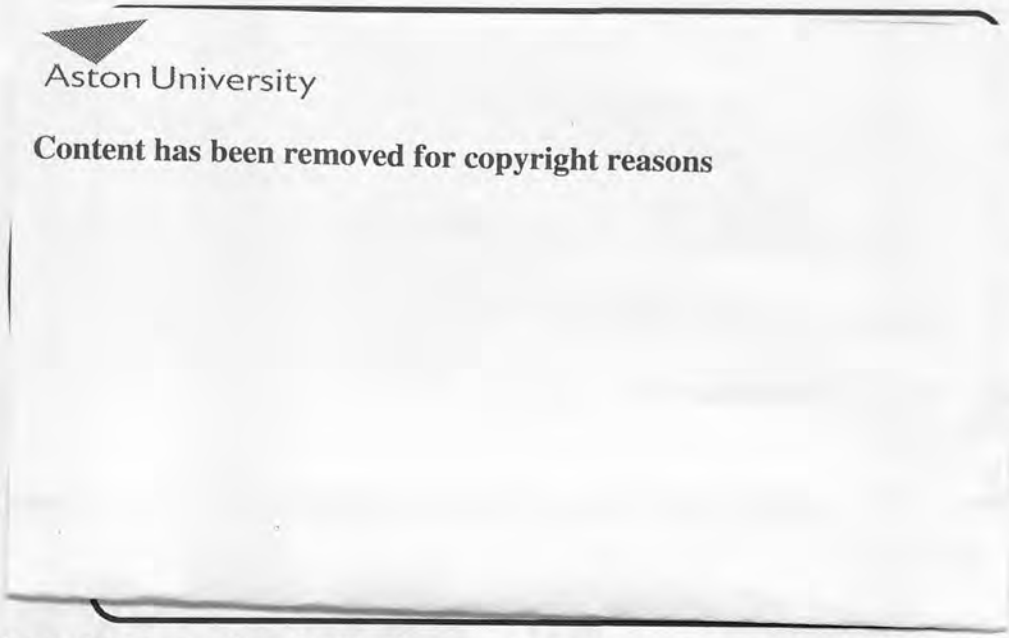
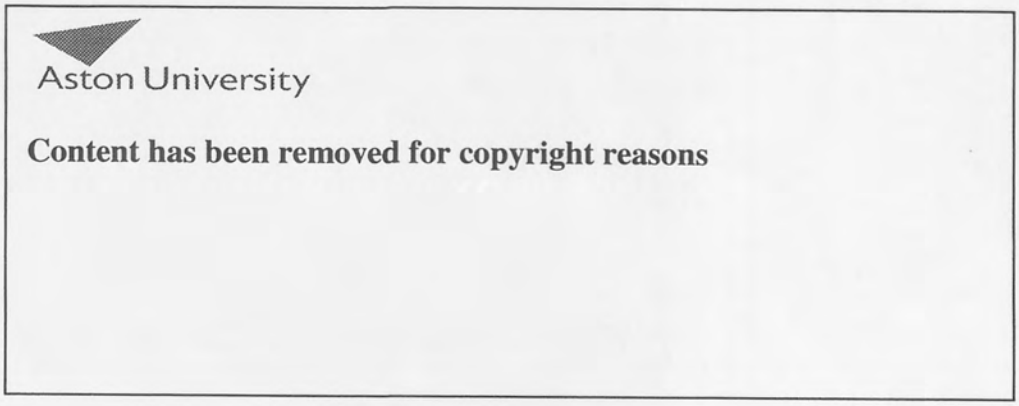


Fig. 7.20. The student feedback prototype - analysing student questionnaires

Although some aspects of the work have not been detailed here in view of space considerations, the case has illustrated many aspects of the Multiview approach in a



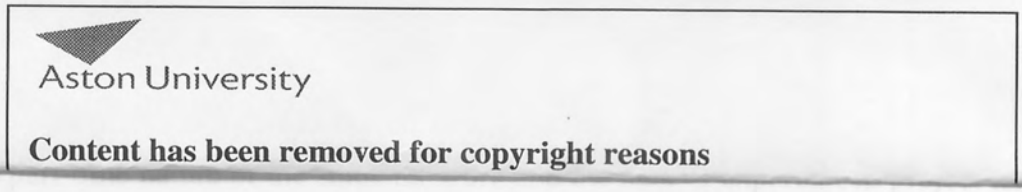


Fig. 7.21. Student feedback prototype - recording students' general comments

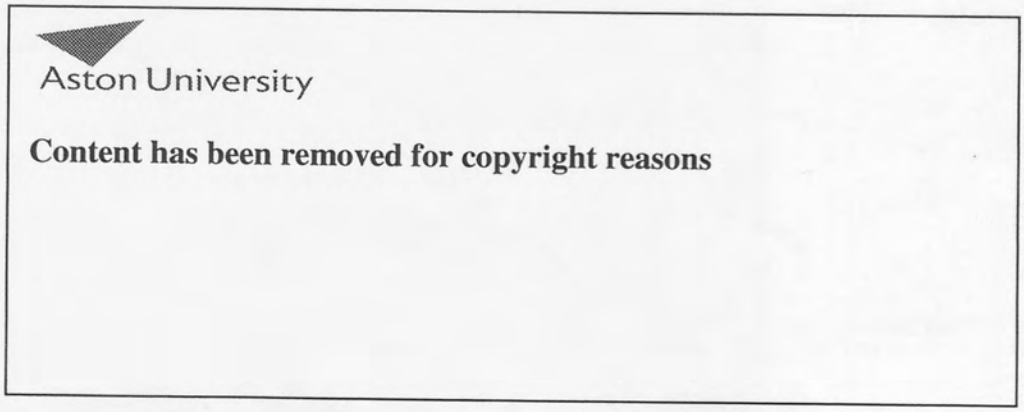


Fig. 7.22. Student record with photograph

fairly large application which is presently only in its early stages of development (despite some four years' work already invested). There are eleven major subsystems that have been identified. Participation by the users has been successful, although there are some difficulties. The analysts are students who could be referred to as 'naive analysts' and they have a difficult role to play in view of the fact that the users are academic and non-academic members of staff. This has caused some difficulties, for example where one academic - from the students' point of view - refused to cooperate, they were unable to exert any pressure and did not know how to handle the situation.

The scale of the project has also resulted in many difficulties. It was a very difficult to coordinate the groups and facilitate communications between them in any one year, and very little possibility of doing so between years. The project control techniques described in Section 5.3.5 help, but do not address this aspect fully. It is very important that staff who have contributed towards the development of the prototypes are rewarded with a working system that helps them in their tasks. The appointment of the computer officer should be crucial in this respect.

Another problem area has been the conflict between the goal of giving students many learning experiences and that of producing working prototypes to standard. For example, student groups would purposefully choose different prototypes from the 'tool box' and choose different database management systems. This gives them useful learning experiences, but slows down the progress of the overall information system. The part-time nature of the students' work and the lack of experienced project leaders also slowed down progress. Indeed, we have been overtaken by progress in tool design so that in each year one group has normally been concerned with transferring prototypes from one database management system to another or one computer system to another. Again, the appointment of the computer officer should help progress.

Multiview has proved a useful approach even in these circumstances, but no approach is a panacea, there will always be human, political and other factors which transcend the description of any information systems development methodology.

Chapter 8
Conclusions

In the particular research described in this thesis, the area of application has been the development of information systems. There is confusion over choice of approaches that might be appropriate for information systems development. In an attempt to remedy this, after a theoretical exploration and classification of approaches which established a number of 'themes' (discussed in Chapter 2), an intellectual framework, called Multiview (discussed in Chapter 3), was constructed. The question then was which research method was appropriate to learn about the theory, the methodology and the development of information systems?

The need for this learning suggests that the practical implications of the three aspects should be explored. To do this, the methodology was used as an exploratory tool to search out patterns in the practical world using action research as the main research method. An example of one such effort was presented as a case in Chapter 4. The problem context was an organisation using systems analysts and users to develop an information system using the Multiview approach. There are iterations in the research - it is a cyclical process - and as the results from the case discussed in Chapter 4 were analysed, Multiview was adapted (Chapter 5) to take account of the practical experience of using it. The modified methodology was itself used as an exploratory tool (two further cases were presented in Chapters 6 and 7). The cases as a whole provide considerable learning material. In this final chapter we draw conclusions on the research work including reflections on the experience of developing Multiview and using it in action. Further work to continue this research is then suggested.

8.1 LESSONS GAINED FROM THE ACTION RESEARCH

8.1.1 A blended, contingency approach to information systems development can work in practice

Multiview is a blended approach to information systems development. It has incorporated aspects of the themes explored in Chapter 2: systems approach, planning approach, participation, prototyping, structured approach and data analysis. It was argued in Section 2.9 that it was unreasonable for an organisation to rely on any methodology falling into any one of these themes for all information systems development. Furthermore, it was also argued that one rigid approach to information

systems development can never be a full answer, because the tools and techniques appropriate for one set of circumstances may not be appropriate for others; the 'fuzziness' of some applications require an attack on a number of fronts; and as an information systems project develops, it takes on very different perspectives or 'views'. Multiview is an explorative structure, a loose framework with which to define and develop information systems, a contingent approach, in that alternative techniques and tools are available which may be chosen according to the dictates and requirements of each situation. However, Multiview does have a framework guiding the user and therefore avoiding a completely loose 'tool kit' approach such as that proposed by Benyon & Skidmore (1987). One of the purposes of this research was to find out if the Multiview approach could work in practice.

The three cases which form part of the action research were used to evidence that it could work. In each case a prototype was developed which has proved useful. The DLU prototype, once modified to the users' satisfaction, 'became' the working system. The case revealed that in a situation where progress did not conform to an 'ideal' pattern, Multiview was helpful because of its flexibility. The prototype was also used as the basis from which a second system was developed for an office technology course.

The community health prototype was used as a prototype for the widely-adopted Comcare system. The systems view provided by Multiview, encouraged the development of a prototype which fulfilled the Körner requirements and also those of the users in community health without much extra cost. Although many users were not favourably disposed towards computing at the beginning because of previous experiences, the prototype was liked by the stakeholders who were willing to use the system in the long term.

The academic department case was the largest of the three, and eleven major subsystems have been identified. The potential users have been cooperative, though it is particularly important to implement at least one working subsystem which will be of benefit to staff. For this reason a full-time computer officer has been employed recently to enable the smooth implementation of the admissions and lecturer/course/student subsystems. Control of this large project has been poor, partly as a result of using students as part-time analysts, but the project control methods described in Section 5.3.5 have not been adequate. Nevertheless, the prototypes would seem to be a good basis for the operational systems.

8.1.2 An information systems development methodology is complex and difficult to learn

The wide range and large number of techniques and tools that need to be described in

an information systems development methodology make such a description long and complex and therefore difficult to learn and to master. Multiview is, perhaps, even more difficult because alternative techniques and tools are included. Further, it does not follow the usual rigid step-by-step description with deliverables, well defined, at each step, because of its contingent philosophy.

In all the cases, the actors using Multiview for the first time took some time to understand the methodology and therefore to use it in the application. The original exposition of Multiview was not thorough and this caused problems for the actors when attempting to use it in the distance learning case. The descriptions of some techniques, for example, were inadequate and therefore they were difficult to use.

Further, as the academic department case showed, although the methodology had been described fully - at least to the lecturer's satisfaction - in a classroom situation, it was evident that the participants only fully understood it following use of the methodology in an action learning situation, in other words using the methodology 'for real'. Without using the methodology in some context, the students were not understanding the methodology. The action learning method of teaching information systems is advocated amongst other methods. Experiences of teaching and learning Multiview can be found in Wood-Harper & Flynn (1983), Antill & Wood-Harper (1984), Wood-Harper (1985 and 1989), and Avison (1989 and 1990a).

8.1.3 The conventional descriptions of information systems methodologies are inappropriate

This methodology, as evidenced by the field work, does not, in practice, exhibit the step-by-step, top-down nature of conventional models and none of the cases have exactly followed the methodology as espoused in Chapter 3 and in Avison & Wood-Harper (1990). The users of the methodology will almost certainly find that they will carry out a series of iterations which are not shown in the model. Further, in the real-world cases undertaken, some phases of the methodology were omitted and others were carried out in a different sequence from that expected. For example, in the distance learning unit case, some of the decisions relating to the technical subsystem were made very early in the project. In the academic case, on the other hand, elements were added to the methodology by the students from their other courses as they thought appropriate. Aspects of the methodology were interpreted and selected depending on the context. For example, in the distance learning and the academic department cases, the social options of the socio-technical phase were either omitted or not fully explored. In the health authority case, the human activity analysis was only partially completed and the whole framework adapted because of the need for Körner as well as process analysis. In this case, in particular, the framework was adapted to suit the

circumstances of the problem situation. The appropriateness of the use of some of the techniques also varied. For example, the comparative value of rich pictures, conceptual models, data flow diagrams and entity models varied greatly between the cases and even within cases, as preferences for various representational forms varies from individual to individual.

8.1.4 The political dimension is important

The manipulation of power, that is, the political dimension, is important in real-world situations. This transcends the rationale of any methodology. Most of the cases show decisions being made which were influenced by considerations beyond those that are implied by the Multiview methodology. For example, in the distance learning case, the choice of equipment was strongly influenced by the necessity, at that time, to purchase British products in order to obtain financial support for the project. Some hardware had already been purchased. It is difficult for analysts to argue that such purchases should not have been made, indeed, post-rationalisation was expected from them. In the academic case, the university steering committee's decisions have impacted on the departmental system and vice versa in a positive way. However, during the lifetime of the project, a UGC/UFC initiative to provide information systems for all universities has changed the direction of the university steering committee. This has caused major problems and the departmental system requires further major adaption to be in line with the university system. The relative power of the naive analysts and academic staff also proved difficult. In one case the decision was made to omit one application from the departmental information system. This was an undesirable consequence of the imbalance of power.

8.1.5 Responsible participation is contingent

A high level of responsible participation, where appropriate, is a positive ingredient of successful information systems development. For example, in the distance learning unit case, the active participation of managers enhanced the usefulness of the methodology, and in the district health authority case, the active participation of the various health workers proved very helpful. Indeed in the community health case, the pilot groups adopted the hand-held data recorders readily and the trial would have been less successful had such a decision been imposed on the groups, particularly as many users were not favourably disposed towards computing at the beginning because of previous experiences.

Users in the community health case did require the presence of the facilitator to encourage them to use the system. This was more of a 'confidence boosting' role than one of adviser. The role of the facilitators proved crucial in all the case studies and the

role is at least as much on the people side of systems than on the technical side.

In the community health case, the facilitator described the three levels of participation (Section 2.5) with users and management and the representative level was chosen because, although there was enthusiasm for full consensus participation, a continual high quality level of service had to be maintained, and this approach would have been impractical. In the academic case, non-academic users in particular have been willing to invest much time in developing the prototype, whereas most academic staff have preferred to be consulted and comment on the proposals rather than be involved further. Other stakeholders such as the university library and administrative staff have been consulted and have commented on the proposals. Students (other than those taking the course) have, on the whole, not been willing to participate. They see the system as not being relevant to them but to future students. One key user, a member of academic staff, refused to cooperate fully. One possible interpretation of this behaviour is that the particular user wanted to develop his own system. The department information system therefore 'invaded' his territory. This contradicts the arguments of 'pure' Multiview, presented in Chapter 3, in which it is assumed that it is always possible to use responsible participation in information systems development.

8.1.6 In certain situations the methodology gives insufficient guidance

The methodology did not give sufficient guidance for all aspects in all the cases. For example, it was difficult to identify the interest groups in the distance learning unit case (though the rich picture contains references to many of them). The students in the academic case did not know how to deal with the situation where one lecturer refused to cooperate with them. Further, the students, effectively naive systems analysts, needed more help than the users in the other cases who were, in fact, less well trained. This was probably due to the students not being used to a situation in which they are expected to use such a high level of initiative (other courses are of a lecture/examination type). In using some of the techniques, particularly in a complex situation, further guidance was needed. The students developing the academic department information system found that the descriptions of some techniques given were inadequate, and they had to delve further in the reference manuals. In some instances they exposed weaknesses in the lecture material given previously and the text used. The work espoused in Chapter 5, forming part of this research, attempts to fill some of these gaps.

8.1.7 The methodology is interpreted by users and analysts

The users and analysts affect the perception of the situation and they interpret the methodology. The variety of possible interpretations reflect the variety of backgrounds

and experiences of the analysts. The academic background of students, for instance, illustrates this as some were studying for a business degree and others for a computer science degree. The emphasis placed on various techniques, for example those user oriented and those computer oriented, was biased by their background. Project teams having a mix of students on both courses sometimes suffered disputes, partly as a result of these differences. This may offset some of the advantages of mixed teams argued by supporters of the systems approach (Section 2.3) which emphasise the positive aspects, such as the benefits of multiple perspectives. In the world of professional analysts and users, where cultures, education, background, and so on, differ, the interpretation of the methodology and the way it is used is also likely to differ.

8.1.8 The technical dimension is also important

In the exposition of Multiview, the social and human dimension has been stressed. This is neglected in many information systems development methodologies. But the research has also exposed difficulties over technical decisions. Sometimes the techniques were not well applied and in all three cases there were major problems with the hardware and software used.

In the DLU case, there were difficulties interfacing the database, computer and optical mark reader. According to the suppliers, this was a trivial matter, but software had to be written and the testing of the interface delayed the implementation of the prototype. In the health authority case, there were major problems relating to the use and the limitations of the applications generator adopted. Programs were difficult to adapt to the specific needs of the application because the code generated by the package was difficult to follow. Some tailor-made programs had to be written. No applications generator was available which would produce code from structured English specifications for the particular computer systems used in the health authority.

In the academic case, many weeks were lost in the development of the departmental information system over the problems of using database management systems, two of which were dispensed with. One, for example, worked well until the number of files became very large, at which point it failed to work. Such limitations are only revealed after using the system 'in anger', that is with significant real data and processing. Another database system was poorly documented and difficult to use and the speed of developing application prototypes was reduced as a consequence.

8.2 CONCLUSIONS FROM THIS EXPERIENCE

The following broad conclusions are implied from the lessons above and the literature.

8.2.1 Multiview is in a continuing state of development

This conclusion is not an attack on Multiview in isolation - all information systems development methodologies have limitations. Information Systems is a comparatively new discipline, the diversity of approaches is caused to some extent by the background and cultures of their authors and none is all-inclusive. The methodologies address a moving target in that the technology, along with tools and techniques supporting it, develop relentlessly. Thus Multiview, as defined in this thesis, is part of a process of improving information systems practice.

In a previous paper (Avison & Wood-Harper, 1986) it is argued that Multiview is an exploration in the development of information systems rather than a methodology, because the latter term suggests a formal, fixed and inflexible approach. There are many changes described in this thesis to the original exposition of Multiview described in Wood-Harper, Antill & Avison (1985).

The writers of that text have had differing experiences when using Multiview since that time. Although there is always a 'family resemblance' in each version of the methodology (for example, the basic structure of the 1985 and 1990 texts are the same), the 'agreed methodology in 1985' has been modified in Avison & Wood-Harper (1990) on the basis of these experiences, the literature and hardware and software developments since that time. Our experiences have been different, because no situation is the same as any other.

The methodology will develop further in the future. New tools are likely to be incorporated, for example, though it is expected that the basic five phase framework, which provides the coherence in the approach, will be maintained. Some additions included in this thesis are not included in the 1990 text. These include decision trees, decision tables and data dictionaries. The omission of data dictionaries is particularly regrettable. These are likely to be included in the next version of Multiview, along with other changes.

8.2.2 Developing an information system is contingent

Developing an information system is contingent on the methodology, the situation and the information systems development team. In some of the cases not all of the stages of Multiview were used because of the situation. It is possible to envisage cases where it is deemed inappropriate to develop a computer-based information system. The framework was adapted in the health authority case. In others, the different analysts (for example, students in the academic department case) interpreted the 'same situation' differently. In any situation where an information system might be appropriate, there are factors such as culture, language and education which have to be taken into consideration. Sometimes the political and social climate is such that participation is

difficult to achieve. In other situations particular tools and techniques are not appropriate to the problem situation. Some tools suggested in Multiview were not used in any of the cases described in the thesis. The systems analyst has to choose from a 'tool box' those tools and techniques appropriate for each situation (Benyon & Skidmore, 1987), but within the framework of an approach such as Multiview (Avison, Fitzgerald & Wood-Harper, 1988). Without such a framework, the information systems are likely to be idiosyncratic and difficult to maintain, and therefore of variable value. The decisions relating to choosing which tools, when and how to use them, is likely to be difficult without some basic structure.

8.2.3 Problems of a contingency approach

Although developing an information system is contingent, there are a number of difficulties associated with adopting a contingency approach. These include factors relating to choice and factors relating to use:

- The first concerns the choice of approach to adopt (that is whether to follow the view of Davis (1982) or that of Iivari (1987) described in Section 1.1.3) and then, respectively, which range of methodologies or which aspects of the framework to include.

  The complexity of information systems methodologies would seem to go against Davis' view that information systems departments ought to have available (and analysts familiar with) a number of methodologies, any of which might be chosen for a given situation. It is too ambitious, perhaps, to expect systems analysts to know many methodologies, with different philosophies, as well as different techniques and tools, covering different parts of the life cycle.

  In Multiview we have constructed a methodology framework following Iivari (rather than proposing a number of methodologies to be used as appropriate following Davis). We have made choices over the techniques and tools to be included, and are aware of the need to re-evaluate these as new ones become available or are discussed in the literature. Although there are difficulties with the Iivari approach, the experience gained from the cases would suggest that Multiview was a useful implementation of a contingency approach to information systems development.
- The second concerns difficulties associated with the levels of expertise and the breadth of knowledge required from systems analysts in order to use these approaches. In a conventional information systems development methodology analysts are expected to follow a particular structure for all cases. This structure is well-defined and the techniques and tools within them are also well-defined.

  The analyst following an approach such as Multiview needs to know which

technique and tool from a range might be appropriate for a particular situation at a particular point of time. This requires wide experience in the practice of systems analysis. The analyst also needs to know a broader range of techniques and tools from which to choose. In both cases, the analyst is helped by the Multiview framework which will guide the analyst in this choice. As stated earlier, this is preferred to a totally free choice. Nevertheless, considerable skill and experience is required to make these choices.

- The third concerns difficulties relating to the consistency of standards in organisations that adopt a contingency approach. Because of the nature of contingency approaches, information systems will be developed using different techniques and tools and in different ways, dependent on the particular situation. This does mean a loss of common standards to some extent and this is one of the practical benefits that the use of an information systems development methodology is supposed to provide.

  This has been recognised by the authors of Multiview and in the latest version, documentation and other standards are recommended. Further, the techniques (such as data flow diagrams and entity life histories) are now more in line with the best practice of information systems. It is necessary to continue developing Multiview in order to keep it in line with information systems knowledge and to continue using Multiview in different problem situations.

8.2.4 Advantages and problems of action research in information systems

Action research allows the researcher great potential to utilise the ideas of users and change concepts and methods as the work develops. Researchers and subjects cooperate in solving a real-life problem. It is particularly useful that action research allows work to take place in its natural setting and it gives the researcher an insight into real-life practical areas. Feedback from the practical application of techniques can be used to refine and improve those techniques and their description. Although the results of action research are of a qualitative nature, they do offer a degree of external validity because the theory developed can be interpreted and refined by others in other real-world situations.

Action research has proved helpful to the development of Multiview. The weaknesses in the descriptions of some of the techniques, such as data flow diagrams and entity life histories, were revealed when using them in the first case. The practicality of using techniques and tools contingent on a particular problem situation as suggested by Multiview, can only be revealed by its use in different situations. Major omissions in the exposition of Multiview, such as the inadequacy of documentation

procedures, project control and the lack of advice when selecting application packages, were revealed in the cases. Some assumptions in Multiview, such as the users' enthusiasm for participation and the low relative importance of the technical dimension have also been questioned through this experience. The part played by researchers as active players in the problem situation (not merely 'impartial' observers), along with users and analysts, has meant that both researchers and practitioners have together influenced the practice (by implementing change in the problem situation) and the theory (by changing the Multiview approach).

However there are disadvantages of action research. The lack of impartiality of the researcher has led to its rejection by a number of researchers and academic departments. The lack of scientific discipline in such research makes it difficult for the work to be assessed for the award of research degrees and for publication in academic journals. A particular difficulty that universities have is persuading research funding bodies that this type of research is as valid and as useful as conventional methods of scientific research. Further, although the researcher's intent is to conduct research while effecting change, the approach is sometimes branded with the description 'consultancy' and not research. The open-endedness of such research and the consequent flexibility necessary in writing a research proposal also provide additional difficulties. Further, as seen in Figure 1.3, a major consequence of the choice of the action research method is that the research is context-bound as opposed to context-free. It is difficult to determine the cause of a particular effect, which could be due to environment (including its subjects), researcher, or methodology. This has been explicit in this research, but it can mean that action research produces narrow learning in its context because each situation is unique and cannot be repeated. This is a major criticism of action research - it does not produce generalisable learning.

However, in this research there is an attempt to reconcile the narrow learning from action research with the need for generalisable research (Warmington, 1980). It is hoped that the generalised findings from this research (discussed in this chapter) will result in other researchers and practitioners applying the Multiview theory in other problem situations and will take into consideration the learning that emerges in the complex process of developing an information system.

8.3 FURTHER RESEARCH

The major requirement is to gain further knowledge of Multiview in action and to improve the practice as well as the theoretical base of Multiview. Not all the techniques and tools discussed in Chapter 5 have been used in a Multiview action research project, and it would be useful to have experience in using them. Some techniques and tools

have not proved adequate, such as the project control techniques suggested in Section 5.3.5. As stated in Section 8.2.1, information systems methodologies need to incorporate new and improved techniques and tools as they become available. Multiview is no exception to this requirement, and as Multiview incorporates further techniques and tools, these need to be used in different problem situations. The methodology is in a continuing state of development and it will develop with practical use.

One important aspect of contingency which has only been partly addressed in this research is the identification of the various elements of contingency and their importance. In general terms, it is suggested that the information systems development process is dependent on the methodology, the problem situation and the team of analysts and users developing the system. Contingency factors relating to the team of analysts and users, for example, will include their training, education, culture, experience, and so on. It would be useful to investigate further this aspect of the research and to suggest when and why these factors are important and how they influence the success of the information system. Further, the aspect of contingency related to how an analyst chooses from the 'tool box' in any particular problem situation would be a major contribution to the field. This could include, for example, some indication as to under what circumstances a particular tool or technique would be advised or ill-advised.

One particular area of possible development concerns the incorporation of multi-media systems and hyperdata into Multiview. The experience with conventional database systems has shown that they are not appropriate to all applications. Conventional databases assume that all information can be classified and retrieved in relational form. This limits the use of the database to a specific range of types of application and users. Some information may not suit this tabular representation, it may be better represented graphically or in image form. Information might be in the form of hand-written and annotated application forms or photographs. Further, some users may find tables an inconvenient and difficult-to-understand way of representing their information. In the long term, we need to develop multi-media databases, with their user interface much more flexible. Hyperdata and multi-media systems in general are likely to be worth investigating.

A new direction would be to develop a methodology workbench of tools which have been designed especially for or tuned to follow the Multiview methodology, so that there is a Multiview methodology workbench. This would be in addition to the incorporation of general tools supporting information systems methodologies which has already been suggested.

As well as incorporating newer techniques and tools is the possibility of changing

the framework of Multiview. It was suggested in Section 8.2.1 that versions of Multiview always show a family resemblance. Further action research might reveal the necessity of changing the basic structure. One major change to the first stage (the analysis of the human activity system) would be to incorporate a full strategic planning phase. One way of achieving this would be to use aspects of the IBM methodology Business Systems Planning (Section 2.4).

Another way of adapting the framework is to incorporate evaluation throughout the development of a system. Evaluation of an information system occurs most frequently as the early stage for cost-justification purposes and following implementation (Horton, 1990). Evaluation in Multiview is also presently confined to a justification stage (albeit taking a much broader view than that found in most methodologies) and post-implementation (again taking a broad view). However, it might be possible to include evaluation throughout the methodology and integrated in it. This would be desirable because that taking place at the feasibility study stage is prone to inaccuracy because it occurs so early in the life of a project (the design decisions have not been made and the information available is poor). At a post-implementation stage there may still be teething problems and unfamiliarity with the system or, if left later, problems which might have been resolved are left for some time during the operation of the system. It might therefore be desirable to have frequent monitoring of the system. One possible form that this could take is a benefits realisation programme (Parker, 1988 and Avison, Catchpole & Horton, 1989).

There are dangers in any type of evaluation and any method should be cross-referenced using results from another method. User reactions described in the health authority case may not represent the situation accurately (Silverman, 1989). User reactions may be determined by the most recent experience with the system. Further, what may look good from the point of view of the user, may not be satisfactory from the analyst's view (or the problem owner). Time is a major factor in evaluation. A follow-up discussion with users in the health authority case carried out in June 1990 showed that the enthusiasm placed on the prototype in 1987 was no longer felt for the operational system. The reasons for this change would be worth investigating.

Finally, one of the major problems, discussed in Section 8.2.3, of using a contingency approach is the expertise required to know which of the techniques and tools available is appropriate for a particular situation at a particular point of time. A prototype expert system could be conceived which holds the expertise of systems analysts to enable advice to be given relating to the parameters applying to that particular situation. It is envisaged that the prototype would expose some of the difficulties of such an attempt.

References

Ackoff, R. L. (1971) Towards a system of system concepts, *Management Science*, 17.

Andersen, E., Kensing, F., Lundin, J., Mathiassen, L., Munk-Madsen, A., Rasbech, M. & Sørgaard, P. (1990) *Professional Systems Development,* Prentice Hall, Hemel Hempstead.

Antill, L. & Wood-Harper, A.T (1984) The Who's doing What for Whom Design Stage, *SOFT*, 2, 2.

Avison, D. E. (1981) Techniques of Data Analysis, *Computer Bulletin*, Series II, 29, September, 1981.

Avison, D. E. (1985) *Information Systems Development: A Data Base Approach*, Blackwell Scientific Publications, Oxford.

Avison, D. E. & Wood-Harper, A. T. (1986) Multiview - An Exploration in Information Systems Development, *Australian Computer Journal*, 18, 4.

Avison, D. E. & Catchpole, C. P. (1987) Information Systems for the Community Health Services, *Medical Informatics*, 13, 2.

Avison, D. E. & Fitzgerald, G. (1988a) *Information Systems Development: Methodologies, Techniques and Tools*, Blackwell Scientific Publications, Oxford.

Avison, D. E. & Fitzgerald, G. (1988b) Information Systems Development: Current Themes and Future Directions, *Information and Software Technology*, 30, 8.

Avison, D. E., Fitzgerald, G. & Wood-Harper, A. T. (1988) Information Systems Development: A Tool-kit is not Enough, *Computer Journal*, 31, 4.

Avison, D. E., Catchpole, C. P & Horton, J. (1989) *Achieving Benefits from Health Information Systems*, MEDINFO89, Sixth World Conference on Medical Informatics, Singapore.

Avison, D. E. (1989) Action Learning for Information Systems Teaching, *International Journal of Information Management*, 9,1.

Avison, D. E. & Wood-Harper, A. T. (1990) *Multiview: An Exploration in Information Systems Development*, Blackwell Scientific Publications, Oxford.

Avison, D. E. (1990a) A Departmental Information System, *International Journal of Information Management*, 10, 2.

Avison, D. E. (1990b) *Mastering Business Microcomputing*, 2nd ed, Macmillan, Basingstoke.

Avison, D. E. & Fitzgerald, G. (1991) Information Systems Practice, Education and Research, *Journal of Information Systems*, 1, 1.

Avison, D. E. & Wilson, D. (1991) Controls for Effective Prototyping, *Journal of Management Systems*, SA-58.

Beer, S. (1985) *Diagnosing the System for Organizations*, Wiley, Maidenhead.

Bemelmans, T. M. A. (Ed.) (1984) *Beyond Productivity: Information Systems Development for Organizational Effectiveness*, North Holland, Amsterdam.

Benbasat, I., Goldstein, D., & Mead, M. (1987) The Case Research Strategy in Studies of Information Systems, *MIS Quarterly*, February, 1987.

Benyon, D. & Skidmore, S. (1987) Towards a tool-kit for the systems analyst, *Computer Journal*, 30, 1.

Bertalanffy, L. von (1968) *General Systems Theory*, Braziller, New York.

Bjorner, D. & Jones, C. (1982) *Formal Specifications and Software Development*, Prentice Hall, Englewood Cliffs, New Jersey.

Bjørn-Andersen, N. (1984) *Challenge to Certainty*, in: Bemelmans (1984).

Bjørn-Andersen, N. & Davis, G. B. (1988) *Proceedings of IFIP WG 8.2 Conference on Information Systems Assessment*, North Holland, Amsterdam.

Blackler, F. & Brown, P. (1988) *Theory and Practice in Evaluation: The Case of the New Information Technologies*, In: Bjørn-Andersen & Davis (1988).

Blumenthal, S. (1969) *Management Information Systems: A Framework for Planning and Development*, Prentice Hall, Englewood Cliffs.

Boar, B. H. (1984) *Application Prototyping: A Requirements Definition Strategy for the 80's*, Wiley, New York.

Boehm, B. W., Gray, T. E. & Seewaldt, T. (1984) Prototyping versus Specifying: A Multiproject Experiment, *IEEE Transactions on Software Engineering*, SE-10(3).

Boland, R. J. (1985) *Phenomenology: A Preferred Approach to Research in Information Systems*, In: Mumford et al (1985).

Bubenko J. A, Jr. (1986) *Information System Methodologies - A Research View*, in: Olle et al (1986).

Buckingham, R. A., Hirschheim, R. A., Land, F. F. & Tully, C. J. (1987) *Information Systems Education: Recommendations and Implementation*, CUP, Cambridge.

Budde, R., Kuhlenkamp, K., Mathiassen. L. & Zullighoven, H. (Eds) (1984) *Approaches to Prototyping*, Springer-Verlag, Berlin.

Bullen, C. V. & Rockart, J. F. (1984) *A Primer on Critical Success Factors*, CISR Working Paper 69, Sloan Management School, MIT, June 1984.

Burns, R. N. & Dennis, A. R. (1985) Selecting the Appropriate Application Development Methodology, *Data Base*, 17, 1.

Burnstine, D. C. (1986) *BIAIT: An Emerging Management Engineering Discipline*, BIAIT International, Petersburg, New York.

Caine, S. H. & Kent, E. G. (1975) PDL - A Tool for Software Design, *National Computer Conference Proceedings*, 44.

Capper, L. (1985) *The Use of Analysis and Design Methodologies in the Working Environment: An Experimental Approach*, In: Bemelmans (1985).

Carlson, W. M. (1979) Business Information Analysis and Integration Technique (BIAIT), The New Horizon, *Data Base*, 10, 4.

Catchpole, C. P. (1985) Survey of the Applications of Computers in the Community Health Services, *Public Health*, 99, 6.

Catchpole, C. P. (1987) *Information Systems Design for the Community Health Services*, PhD Thesis, Aston University, Birmingham.

Catchpole, C. P., Avison, D. E. & Peart, S. (1987) A Tale of Two Systems, *Nursing Times*, 83, 34.

Checkland, P. B. & Griffin, R. (1970) Management Information Systems: A Systems View, *Journal of Systems Engineering*, 1, 2.

Checkland, P. B. (1975) The Development of Systems Thinking by Systems Practice: A Methodology from an Action Research Program, In: Trappl, R. & Hawika, F. de P. (Eds), *Progress in Cybernetics and Systems Research*, Vol II, Hemisphere, Washington.

Checkland, P. B. (1981) *Systems Thinking, Systems Practice*, John Wiley, Chichester.

Checkland, P. B. (1984) *Rethinking a Systems Approach*, In: Tomlinson & Kiss (1984).

Checkland, P. B. (1985) From Optimisation to Learning: A Development of Systems Thinking for the 1990's, *Journal of the Operational Research Society*, 36, 9.

Churchman, C. W. (1971) *The Design of Inquiring Systems*, Basic Books, New York.

Churchman, C. W. (1979) *The Systems Approach and its Enemies*, Basic Books, New York.

Codd, E. F. (1974) *Seven Steps to Rendezvous with the Casual User*, In: Klimbie & Kofferman (1974).

Collongues, A., Hugues. J. & Laroche, B. (1989), *Merise: 1. Méthode de Conception*, Dunod, Paris.

Cougar, J. D. & Knapp, R.W. (1974) *Systems Analysis Techniques*, Wiley, New York.

Cougar, J. D., Colter, M. A. & Knapp, R. W. (1982) *Advanced Systems Development/Feasibility Techniques*, Wiley, New York.

Curtice R. M. & Dieckmann, E. M. (1981) A Survey of Data Dictionaries, *Datamation*, March.

Crowe, T. & Avison D. E. (1980) *Management Information from Data Bases*, Macmillan, Basingstoke.

Daniels, A. & Yeates, D. A. (1971) *Basic Training in Systems Analysis*, 2nd ed., Pitman, London.

Date, C. J. (1984) *A Guide to DB2*, Addison Wesley, Reading.

Date, C. J. (1986) *An Introduction to Database Systems*, 4th ed, Addison Wesley, Cambridge.

Davenport, R. A. (1978) Data Analysis for Data Base Design, *Australian Computer Journal*, 10.

Davis, G. B. (1982) Strategies for Information Requirements Determination, *IBM Systems Journal*, 21, 2, pp 4-30.

Davis, G. B. & Olsen, M. H. (1985) *Management Information Systems: Conceptual Foundations, Structure and Development*, 2nd ed, McGraw-Hill, New York.

Dearnley, P. A. & Mayhew, P. J. (1983) In Favour of System Prototypes and their Integration into the Systems Development Cycle, *Computer Journal*, 26, 1.

DeMarco, T. (1979) *Structured Analysis and System Specification*, Prentice Hall, Englewood Cliffs.

Dickson, G. S., Senn, J. & Chervany, N. L. (1977) Research in Management Information Systems: The Minnesota Experiments, *Management Science*, 23, 9.

Douglas, J. D. (1976) *Investigative Social Research*, Sage, Beverly Hills, California.

Downs, E., Clare, P. & Coe, I. (1988) *Structured Systems Analysis and Design Method: Application and Context*, Prentice Hall, Hemel Hempstead.

Eason, K. D. (1984) Towards the Experimental Study of Usability, *Behaviour and Information Technology*, 2, 2.

Ein-Dor, P. & Segev. E. (1981) *A Paradigm for Management Information Systems*, Prager, New York.

Episkopou, D. M. & Wood-Harper, A. T. (1986) Towards a Framework to Choose Appropriate IS Approaches, *Computer Journal*, 29, 3.

Espejo, R. & Harnden, R. (1989) *The Viable Systems Model*, Wiley, New York.

Etzerodt, P & Madsen, K. H. (1988) *Information Systems Assessment as a Learning Process*, In: Bjørn-Andersen & Davis (1988).

Galliers, R. D. (1985) *In Search of a Paradigm for Information Systems Research*, In: Mumford et al (1985).

Galliers, R. (1986) *Information Systems Planning*, Internal paper, London School of Economics.

Galliers, R. (1987) *Information Analysis: Selected Readings*, Addison Wesley, Sydney.

Galliers, R. D. & Land, F. F. (1987) Choosing Appropriate Information Systems Research Methodologies, *Communications of the ACM*, 30, 11.

Gane, C. P. & Sarson, T. (1979) *Structured Systems Analysis: Tools and Techniques*, Prentice Hall, Englewood Cliffs.

Ginzberg, M. J. & Zmud, R. W. (1988) *Evolving Criteria in Information Systems Assessment*, In: Bjørn-Andersen & Davis (1988).

Gradwell, D. J. L. (1983) *ICL's DDS, DD Update*, British Computer Society, London, April.

Grindley, C. B. B. (1966) *SYSTEMATICS - A Nonprogramming Language for Designing and Specifying Commercial Systems for Computers*, In: Cougar & Knapp (1974).

Grindley, C. B. B. (1968) *The Use of Decision Tables within Systematics*, In: Cougar & Knapp (1974).

Grindley, C. B. B. (Ed.) (1987) *Information Technology Review 1987/88*, Price Waterhouse, London.

Hamilton, J. & Chervany, N. (1981) Evaluating Information Systems Effectiveness Part 1: Comparing Evaluation Approaches, *MIS Quarterly*, September.

Hawgood, J. & Land, F, F. (1986) *A Multivalent Approach to Information System Assessment*, In: Bjørn-Andersen & Davis (1988).

Heidegger, M. (1962) *Being and Time*, Harper and Row, New York.

Hekmatpour, S. & Ince, D. (1986) *Rapid Software Prototyping*, Open University, Technical Report, 86/4.

Herzberg, F. (1966) *Work and the Nature of Man*, Staple Press, New York.

Hirschheim, R. (1985) *Information Systems Epistemology: An Historical Perspective*, In: Mumford et al (1985).

Horowitz, E. (1985) A Survey of Application Generators, *IEEE Software*, January 1985.

Horton, J. (1990) The Evaluation of Information Systems in the UK National Health Service, Working Paper, Aston University.

Howe, D. R. (1983) *Data Analysis for Data Base Design*, Arnold, London.

Hult, M. & Lennung, S. (1980) Towards a Definition of Action Research: A Note and a Bibliography, *Journal of Management Studies*, 17, 2.

IBM (1971) *The Time Automated Grid System* (TAG), In: Cougar & Knapp (1974).

IBM (1975) *Business Systems Planning*, In: Cougar, Colter & Knapp (1982).

Iivari, J. (1987) *A Methodology for IS Development as an Organisational Change: a Pragmatic Contingency Approach*, In: Klein & Kumar (1987).

Jackson, M. A. (1983) *Systems Development*, Prentice Hall, Englewood Cliffs.

Jackson, M. C. (1987) *New Directions in Management Science*, In: Jackson, M. C. & Keys, P (1987) New Directions in Management Science, Gower, London.

Jeffery, D. R. & Low, G. (1989) Productivity Issues in the Use of Current Back-end CASE Tools, *Third International Workshop on Computer-Aided Software Engineering*, London, July.

Jenkins, A. M. (1985) *Research Methodologies and MIS Research*, In: Mumford et al., (1985).

Kent, W. A. (1978) *Data and Reality*, North Holland, Amsterdam.

Klein, H. K. & Lyytinen, K. (1985) *The Poverty of Scientism in Information Systems*, In: Mumford et al, (1985).

Klein, H. K. & Kumar, K. (1987) *Information Systems Development for Human Progress in Organisations*, North Holland, Amsterdam.

Klimbie, J. W. & Kofferman, K. L. (Eds) (1974) *Proc. of the IFIP Conference on Data Base Management*, North Holland, Amsterdam.

Körner, E. (1982) *First Report to the Secretary of State on Health Service Information*, Körner Steering Group, DHSS/NHS, London.

Körner, E. (1984a) *Fourth Report to the Secretary of State on Health Service Information*, Körner Steering Group, DHSS/NHS, London.

Körner, E. (1984b) *Fifth Report to the Secretary of State on Health Service Information*, Körner Steering Group, DHSS/NHS, London.

Korth, H. F. & Silberschatz, A. (1986) *Database System Concepts*, McGraw-Hill, New York.

Land, F. F. (1982) Adapting to Changing User Requirements, *Information and Management*, 5, In: Galliers (1987).

Land, F. F. (1983) quoted In: Owen, K., *Formal Methods in an Informal World*, IFIP Conference, Geneva (1983) and in Wood-Harper (1989).

Land, F. F. & Hirschheim, R. (1983) Participative Systems Design: Rationale, Tools and Techniques, *Journal of Applied Systems Analysis*, 10.

Land, F. F. (1985) Is an Information Theory Enough?, *Computer Journal*, 28, 3.

Land, F. F. & Somogyi, E. (1986) Software Engineering: The Relationship between a Formal System and its Environment, *Journal of Information Technology*, 1, 1.

Land, F. F. & Kennedy-McGregor, M. (1987) *Information and Information Systems: Concepts and Perspectives*, In: Galliers (1987).

Lantz, K. E. (1985) *The Prototyping Methodology*, Prentice-Hall, Englewood Cliffs.

Law, D.T. (1985) *Prototyping*, NCC, Manchester.

Lobell, R. F. (1983) *Application Program Generators, A State of the Art Survey*, NCC, Manchester.

Longworth, G. (1985) *Designing Systems for Change*, NCC, Manchester.

Lundeberg, M., Goldkuhl, G. & Nilsson, A. (1982), *Information Systems Development - A Systematic Approach*, Prentice Hall, Englewood Cliffs, New Jersey.

Lundeberg, M. (1982) *The ISAC Approach to Specification in Information Systems and its Application to the Organization of an IFIP Working Conference*, In: Olle et al (1982).

McCracken, D. D. & Jackson, M. A. (1982) Life Cycle Concept Considered Harmful, *ACM SIGSOFT, Software Engineering Notes*, 17, 2.

MacDonald, I. G. & Palmer, I. (1982) *System Development in a Shared Data Environment, the D2S2 Methodology*, In: Olle *et al* (1982).

MacDonald, I. G. (1987) *Automating Information Engineering*, unpublished paper, James Martin Associates, London.

McFarlan, F. W. (Ed.) (1984) The Information Systems Research Challenge, Harvard Business School Press, Boston, Mass.

Maddison, R. N. (Ed.) (1983) *Information System Methodologies*, Wiley Heyden, Chichester.

Mansell, G. (1990), Application of a Problem-Solving Methodology to Software Usability, submitted PhD thesis, Aston University.

Markus, L. M. (1984) *Information Systems in Organisations: Bugs and Features,* Pitman, Marshfield.

Martin, J. (1980) *Strategic Data Planning Methodologies*, Savant, Carnforth, Lancashire.

Martin, J. & Finkelstein, C. (1981) *Information Engineering, Vols 1 and 2*, Prentice Hall, Englewood Cliffs, New Jersey.

Martin, J. (1982a) *Application Development without Programmers*, Prentice Hall, Englewood Cliffs.

Martin, J. (1982b) *Fourth Generation Languages, Vol 1*, Prentice Hall, Englewood Cliffs.

Martin, J. (1983) *Fourth Generation Languages, Vol 2*, Prentice Hall, Englewood Cliffs.

Martin. J. & McClure. C. (1984) *Structured Techniques for Computing*, Savant Research Institute, Carnforth.

Mayhew, P. J. & Dearnley, P. A. (1987) An Alternative Prototyping Classification, *Computer Journal*, 30, 6.

Mumford, E., Land. F. F. & Hawgood, J. (1978) A Participative Approach to Computer Systems, *Impact of Science on Society*, 28, 3.

Mumford, E. & Weir, M. (1979) *Computer Systems in Work Design - The ETHICS Method*, Associated Business Press, London.

Mumford, E. (1981) Participative Systems Design: Structure and Method, *Systems, Objectives and Solutions*, 1.

Mumford, E. (1983a) *Designing Human Systems*, Manchester Business School, Manchester.

Mumford, E. (1983b) *Designing Participatively*, Manchester Business School, Manchester.

Mumford, E. (1985a) Defining Systems Requirements to meet Business Needs: A Case Study Example, *Computer Journal*, 28, 2.

Mumford, E. (1985b) *From Bank Teller to Office Worker: The Pursuit of Systems Designed for People in Practice and Research*, The 6th International Conference on Information Systems, December, Indianapolis.

Mumford, E. (1985c) *Socio-Technical Systems Design Evolving Theory and Practice*, IFIP TC9 Conference, August, Århus.

Mumford, E., Hirschheim, R. A., Fitzgerald, G. & Wood-Harper, A. T (Eds) (1985) *Research Methods in Information Systems*, North-Holland, Amsterdam.

Mumford, E. (1986) *Using Computers for Business,* Manchester Business School, Manchester.

Myers, G. J. (1975) *Reliable Software through Composite Design*, Petrocelli/Charter, New York.

Myers, G. J . (1978) *Composite/Structured Design*, Van Nostrand Rinehold, New York.

Naumann, J. D. & Jenkins, A. M. (1982) Prototyping: The New Paradigm for Systems Development, *MIS Quarterly*, 6, 3.

NCC (1986) *SSADM Manual*, National Computing Centre, Manchester.

Nelson, K. (1985) Technical Requirements of a 4-GL, *Data Processing*, November.

Olle, T. W., Sol, H. G. & Verrijn-Stuart, A. A. (Eds) (1982) *Information Systems design Methodologies: A Comparative Review*, North Holland, Amsterdam.

Olle, T. W., Sol, H. G. & Tully, C. J. (Eds) (1983) *Information Systems Design Methodologies: A Feature Analysis*, North Holland, Amsterdam.

Olle, T. W., Sol, H. G. & Verrijn-Stuart, A. A. (Eds) (1986) *Information Systems design Methodologies: Improving the Practice*, North Holland, Amsterdam.

Page-Jones, M. (1980) *The Practical Guide to Structured Systems Design*, Yourdon, New York.

Parker, R. (1988) Realizing Benefits: The Role of Management and the Computer, *Computers and Healthcare*, 9,3.

Parsons, T. & Shils, E. (1951) *Towards a General Theory of Action*, Harvard University Press, Mass.

Pergamon (1987), *Analyst Workbenches: State of the Art Report*, 15,1 Pergamon-Infotech, Maidenhead.

Quang, P. T. & Chartier-Kastler, C. (1991) *Merise in Practice*, Macmillan, Basingstoke (translated by D. E. & M. A. Avison from the French *Merise Appliquée*, Eyrolles, Paris, 1989).

Robinson, H. (1989) *Database Analysis and Design*, 2nd ed., Chartwell-Bratt, Bromley.

Rock-Evans, R. (1981) *Data Analysis*, IPC Press, London.

Rock-Evans, R. (1989) *CASE Analyst Workbenches: A Detailed Product Evaluation*, Ovum Report, London.

Rockart, J. F. (1979) Chief Executives Define Their Own Data Needs, *Harvard Business Review*, March-April.

Shave, M. J. R. (1981) Entities, Functions and Binary Relations: Steps to a Conceptual Schema, *Computer Journal*, 24, 1.

Silverman, D. (1989) Six Rules of Qualitative Research: A Post-Romantic Argument, *Symbolic Interaction*, 12, 2.

Stevens, W., Myers, G. & Constantine, L. (1974) Structured Design, *IBM Systems Journal*, May.

Susman, G. & Evered, R. D. (1978) An Assessment of the Scientific Merits of Action Research, *Administrative Science Quarterly*, 23, December.

Susman, G. (1983) Action Research: A Sociotechnical Systems Perspective, In: Morgan, G. (Ed.) (1983) *Beyond Method: Strategies for Social Research*, Sage, Beverly Hills, California.

Symons, V. & Walsham, G. (1988) The Evaluation of Information Systems: A Critique, *Journal of Applied Systems Analysis*, 15.

Teichroew, D. & Hershey, E. A. (1977) PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems, In: Cougar et al (1982).

Teichroew, D., Hershey, E. A. & Yamamoto, Y. (1979) *The PSL/PSA Approach to Computer-Aided Analysis and Documentation*, In: Cougar et al (1982).

Tozer, E. (Ed.) (1984) *Application Development Tools - A State of the Art Report*, Pergamon-Infotech.

Utterback, J. M. & Abernathy, W. J. (1975) A Dynamic Model of Process and Product Innovation, *Omega*, 3, 6.

Van Horn, R. L. (1973) Empirical Studies of Management Information Systems, *Data Base*, 5, Winter.

Veryard, R. (1984) *Pragmatic Data Analysis*, Blackwell Scientific Publications, Oxford.

Warmington, A. (1980) Action Research: Its Methods and its Implications, *Journal of Applied Systems Analysis*, 7.

Waters, S. J. (1979a) *Systems Specifications*, NCC, Manchester.

Waters, S. J. (1979b) Towards Comprehensive Specifications, *Computer Journal*, 22, 3.

Weick, K. E. (1984) *Theoretical Assumptions and Research Methodology Selection* In: McFarlan (1984).

Weinberg, V. (1978) *Structured Analysis*, Prentice Hall, New Jersey.

Welke, R. J. (1987) *The New Architecture of PSL/PSA*, Ann Arbor, MI:MetaSystems.

Wetherbe, J. C. & Davis, G. B. (1983) *Developing a long-range information architecture*, Proceedings of the National Computer Conference, Anaheim, Calif, May 1983.

Willcocks, L. & Mason, D. (1987) *Computerising Work*, Paridigm, London.

Wilson, B. (1984) *Systems: Concepts, Methodologies and Applications,* Wiley, Chichester.

Winograd, T. & Flores, F. (1986) *Understanding Computers and Cognition: A New Foundation for Design*, Ablex, New Jersey.

Wood-Harper, A. T. & Fitzgerald, G. (1982) A Taxonomy of Current Approaches to Systems Analysis, *Computer Journal*, 25, 1.

Wood-Harper, A. T. & Flynn, D. J. (1983) Action Learning for Teaching Information Systems, *Computer Journal*, 26, 1.

Wood-Harper, A. T. (1985) *Information Research Methods: Using Action Research*, In: Mumford et al (1985).

Wood-Harper, A. T., Antill, L. & Avison, D. E. (1985) *Information Systems Definition: the Multiview Approach,* Blackwell Scientific Publications, Oxford.

Wood-Harper, A. T. (1989) *Comparison of Information Systems Definition Methodologies: Action Research Multiview Perspective*, PhD Thesis, University of East Anglia, Norwich.

Wood-Harper, A. T. & Avison, D. E. (1991) *An Enquiry into Information Systems: A Research Perspective,* Blackwell Scientific Publications, Oxford.

Yadav, S. B. (1983) Determining an Organisations Information Requirements: A State of the Art Survey, *Data Base*, 14, 3.

Yourdon, E. (1989) *Modern Structured Analysis*, Prentice Hall, Englewood Cliffs.

Yourdon, E. & Constantine, L. L. (1978) *Structured Design*, 2nd. ed., Yourdon, New York.

Zani, W. M. (1970) Blueprint for MIS, *Harvard Business Review*, Nov-Dec.

Zloof, M. M. (1978) Design Aspects of the Query-by-Example Data Base Management Language, In: Shneiderman, B (Ed.) *Improving Database Usability and Responsiveness*, Academic Press, New York.