THE UNIVERSITY OF ASTON IN BIRMINGHAM

USER INTERFACE DESIGN SUPPORT FOR THE DEVELOPMENT OF KNOWLEDGE-BASED SYSTEMS

CLIVE KINGSLEY BRIGHT

Submitted for the Degree
of
Doctor of Philosophy

MAY 1989

THE UNIVERSITY OF ASTON IN BIRMINGHAM

USER INTERFACE DESIGN SUPPORT IN THE DEVELOPMENT OF KNOWLEDGE-BASED SYSTEMS

Clive Kingsley Bright

Submitted for the Degree of Doctor of Philosophy
1989

The present scarcity of operational knowledge-based systems (KBS) has been attributed, in part, to an inadequate consideration shown to user interface design during development. From a human factors perspective the problem has stemmed from an overall lack of user-centred design principles. Consequently the integration of human factors principles and techniques is seen as a necessary and important precursor to ensuring the implementation of KBS which are useful to, and usable by, the end-users for whom they are intended.

Focussing upon KBS work taking place within commercial and industrial environments, this research set out to assess both the extent to which human factors support was presently being utilised within development, and the future path for human factors integration. The assessment consisted of interviews conducted with a number of commercial and industrial organisations involved in KBS development; and a set of three detailed case studies of individual KBS projects. Two of the studies were carried out within a collaborative Alvey project, involving the Interdisciplinary Higher Degrees Scheme (IHD) at the University of Aston in Birmingham, BIS Applied Systems Ltd (BIS), and the British Steel Corporation. This project, which had provided the initial basis and funding for the research, was concerned with the application of KBS to the design of commercial data processing (DP) systems. The third study stemmed from involvement on a KBS project being carried out by the Technology Division of the Trustees Savings Bank Group plc.

The preliminary research highlighted poor human factors integration. In particular, there was a lack of early consideration of end-user requirements definition and user-centred evaluation. Instead concentration was given to the construction of the knowledge base and prototype evaluation with the expert(s). In response to this identified problem, a set of methods was developed that was aimed at encouraging developers to consider user interface requirements early on in a project. These methods were then applied in the two further projects, and their uptake within the overall development process was monitored.

Experience from the two studies demonstrated that early consideration of user interface requirements was both feasible, and instructive for guiding future development work. In particular, it was shown a user interface prototype could be used as a basis for capturing requirements at the functional (task) level, and at the interface dialogue level. Extrapolating from this experience, a KBS life-cycle model is proposed which incorporates user interface design (and within that, user evaluation) as a largely parallel, rather than subsequent, activity to knowledge base construction. Further to this, there is a discussion of several key elements which can be seen as inhibiting the integration of human factors within KBS development. These elements stem from characteristics of present KBS development practice; from constraints within the commercial and industrial development environments; and from the state of existing human factors support.

Keywords: Knowledge-based systems Expert systems User interface design Human factors

to my Family

Acknowledgements

There are several people to whom thanks are due for making this research an interesting, stimulating, and very valuable experience. Firstly, sincere thanks go to the various 'collaborators' who have been involved along the way; to the Alvey team members at BIS, David Hannaford, Janet Clifford, Christopher Harris-Jones and Raj Arya; to Tony Dignan from the Alvey Directorate, and to the group at British Steel. Particular thanks also go to Jim Kielty at the TSB whose friendly and supportive cooperation enabled me to gain further experience of 'real-life' KBS work.

At the 'home base' of Aston, I am extremely grateful to my fellow IHD students who have provided an interesting environment in which to exchange ideas, and indeed to share common problems which cut across specific research topics. In particular, thanks go to Jon Bader, a fellow colleague on the Alvey project, and to Andrew Carruthers, James Pang, Al Rodger, and Hardial Sagoo. From the Ergonomics Group of the Applied Psychology Unit, with which I was also fortunate to be affiliated, thanks go to Chris Baber and Neville Stanton; with very special thanks to Mick Carey, who provided a constant source of useful information, and with whom I had many fruitful discussions. And lastly from that group, Jane Astley who as the much needed 'contemporary', was always willing to share ideas and experience.

Above all my thanks go to my supervisors. To my associate supervisor, Dr Alastair Cochran, Director of the IHD Scheme, whose perceptiveness and considerable experience of PhD supervision, are a regular calming influence upon the histrionics of many a research student. Lastly, but of course not least of all, many thanks to my main supervisor, Dr Rob Stammers of the Ergonomics Unit, whose keenness to engender a sense of 'team-spirit' within the Ergonomics Group was much appreciated, and whose guidance of this research was invaluable.

LIST OF CONTENTS

LIST OF FIGURES

CHAPTER EIGHT

CHAPTER NINE

LIST OF TABLES

CHAPTER TWO

CHAPTER THREE

CHAPTER FIVE

CHAPTER SEVEN

CHAPTER EIGHT

Chapter One

Introduction

Outline

The research reported in this thesis addresses the subject of user interface design in knowledge-based systems (KBS) development. In this chapter the research is put in the context of both the Alvey information technology initiative (under which the work is funded), and within the Intellipse project, in which the research was based. Intellipse was a collaborative project concerned with the development of KBS to support the design of commercial data processing systems. Involvement in the development of various KBS in Intellipse was a primary source of information for the research. The collaborators in this project were BIS Applied Systems, the British Steel Corporation and Aston University, where this work was based. Following brief descriptions of the role of these collaborators, the chapter sets out the high-level objectives of the research and the plan of work which was subsequently formulated to address these objectives.

1.1. The Alvey Programme

In 1983, following the report of the Alvey Committee, the Government announced an information technology programme. This research and development initiative was in response to an earlier announcement by the Japanese of their own "Fifth Generation Computer Programme". The Alvey programme was divided into four main technological streams, representing those areas considered important in the development of information technology (Alvey Annual Report 1984). The four streams were Intelligent Knowledge-Based Systems (IKBS), the Man-Machine Interface (MMI), Software Engineering, and Very Large Scale Integration (VLSI). Collaborative projects, involving both academic and industrial partners, were set up to carry out work in these areas. The four streams represented a broad classification; in reality many of the Alvey projects involved a certain degree of overlap between areas.

1.2. The Intellipse project

The research that is described in this thesis is based on work which was driven from within an Alvey-funded project. This project was called *Intellipse*. Although formally based within the software engineering stream the Intellipse project covered, in some depth, aspects from the IKBS and MMI areas. Figure 1, overleaf, shows the links between the three streams in the context of this project.

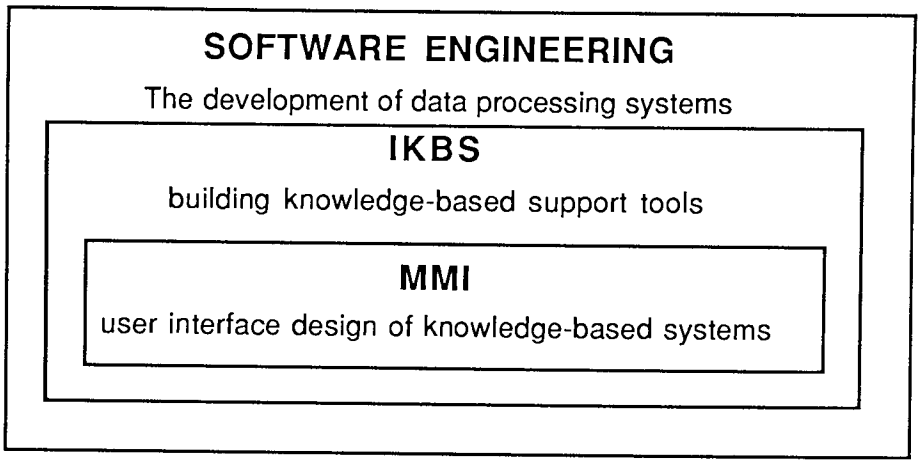


Figure 1.1. Areas addressed within the Intellipse project

Data processing (DP) system development, a field of software engineering, was the overall domain which was addressed within Intellipse through the construction of knowledge-based tools. The aim of the project, as described in the original project proposal, was to:-

> "...develop a series of integrated knowledge-based systems (IKBS) which can be used to support the design phase of a data processing systems development." (BIS 1984; p 2)

A further stated objective of the project was to "produce a framework of techniques and tools which could be incorporated in the development of an IPSE". An IPSE, an acronym for 'Integrated Project Support Environment" is a set of integrated software tools which support the management and design of large software development projects. The project name *Intellipse*, was a concatenation, indicating a future long-term aim of building an *'Intell* igent *IPSE*".

The knowledge-based, or expert system, tools were to be based upon a structured system design methodology developed, and provided through consultancy, by BIS Applied Systems (BIS) one of the collaborating organisations. The construction of these knowledge-based tools provided the focal point for the third dimension of the Intellipse project. This was the MMI dimension, concerned with the user interface design of knowledge-based systems. It is this latter area which forms the research project described in this thesis.

1.3. BIS Applied Systems

The *Intellipse* project was made up of a tripartite collaboration involving BIS Applied Systems Limited, Aston University and the British Steel Corporation (BSC). BIS is a consultancy company, providing a range of services in the area of information systems. Importantly for the work in *Intellipse*, these services include a set of structured methods for information system development. These methods are part of an overall set of Development Standards known as *MODUS*. It was the experience and knowledge of these structured methods which was to be the basis for the work in *Intellipse*. Although BIS provide structured methods for all the phases in system development, from feasibility to implementation and testing, this project focussed on one phase of development - the system design phase. Four consultants, based in the Birmingham office of the company, were involved in the project, with a Principal Consultant from BIS acting as the overall project manager.

1.4. British Steel Corporation

The role of BSC in the project was that of 'end-user'. BSC represents an organisation with a very large data processing division and so were seen as ideally representative of potential users of the KBS which would emerge from *Intellipse*. Members of the Computer Systems Development Research Centre, based at Port Talbot in South Wales were assigned to the project. These members were involved in systems analysis and design, and database management activities within BSC. These are all activities which were pertinent to the proposed work in *Intellipse*.

1.5. Aston University

Aston University, which, like BIS is also based in Birmingham, acted as the academic collaborator in the project. Two full-time research officers, of which the author was one, were based in a unit known as IHD (Interdisciplinary Higher Degrees). Initiated in 1968, IHD had considerable experience of collaborative research with industry (Cochran 1981) and this was particularly desirable given one of the primary aim's of the Alvey initiative - that of stimulating industrial and academic collaboration. Furthermore the Scheme's emphasis upon interdisciplinary research was well suited to the Intellipse project, which as discussed previously, impinged upon several areas within information technology.

An interdisciplinary approach to research has also been highlighted as being an important element in the context of human factors. The following extract is from a report which was published towards the end of the Alvey initiative, it comes from the Director of the Human Interface Club, a club set up to monitor activities within the MMI stream. The

extract identifies three key problem areas seen as factors obstructing the 'technology transfer' of Human Interface knowledge and the awareness of concomitant issues:-

> "Firstly, there is the continued reticence of much of the academic community to recognise product or service design as a subject that is as worthy of study as, for example, experimental design or the study of language and its use....Secondly, there is the continued misunderstanding of the nature of interdisciplinary research and the relative contributions of material and social scientists within this type of team effort. Thirdly, there appears a reticence to believe it worthwhile to involve the actual user in the design process - the cost appears to outweigh the advantages." (Clarke and Tainsh 1987; section 3.1)

The issues identified in the above extract are extremely pertinent to the work which forms the basis for this thesis. The essence of this work, with its interdisciplinary roots, addresses, to some degree, the issues considered important in the above extract.

Before describing the objectives of this research in more depth, it is worth describing briefly, some of the objectives of the MMI initiative within Alvey. The description gives some indication of the background issues in the early stages of the initiative.

1.6. MMI and Alvey

In comparison with the other streams in Alvey MMI was slow to get off the ground. After the first year 60% of the funds originally allocated to MMI had been committed (that is projects had been approved and funding agreed) compared with figures of 75-80% for VLSI and almost total fund commitment in the IKBS stream. In his Director's Report (Alvey Annual Report 1985) Oakley pointed to one possible explanation for the slower onset of MMI projects :-

> "This field (MMI) is the least developed of the programme, if only because the subjects are most immature" (p 11)

The immaturity of the MMI field causes something of a dilemma given that a central aim of the overall MMI strategy was to :-

> "... increase the awareness of the importance of good user interface design in the design of products." (Alvey Annual Report 1984; p 16)

An understanding of MMI principles in design requires a thorough understanding of both the human user (something which is both complex and highly variable in nature) and the machine technology (i.e. the computer system - in terms of hardware and software). Barrow (1985) raises some of these issues, going on to point out the need for multi-disciplinary research :-

> "The design of an interface to maximise effectiveness of this communication (*man-machine interaction*) requires a knowledge of the characteristics of the human being which are pertinent to his use of computer-based systems, and a knowledge of the technology of the system. The area is therefore multi-disciplinary, requiring the expertise of, for instance, applied psychologists, computer scientists and hardware engineers working closely together." (p 35)

Following its relatively slow beginning, interest in MMI showed a significant increase. By 1986 funds for MMI projects had been fully committed. The increased level of interest was also shown, in part, by the number of requests for information on MMI topics via the Alvey MMI mailshot. Indeed the number of requests for this particular mailshot exceeded that for any of the other respective streams (Alvey Annual Report 1985).

1.7. Objectives of MMI research in Intellipse

It was proposed that the MMI related work would have two main objectives:-

1. To give guidance on MMI issues which arose within *Intellipse*.
2. To identify, more clearly, the human factors issues in KBS development.

The first objective was specific to the *Intellipse* project. The intention was to provide distinct guidance on the design of the user interface of the various KBS modules which were to make up Intellipse. The second objective, serving a more general purpose, would utilise the experience from work on *Intellipse*. This experience would form the basis for understanding the relationship, in a pragmatic context, between human factors and KBS development. Figure 2 represents an overview of the context of the work.

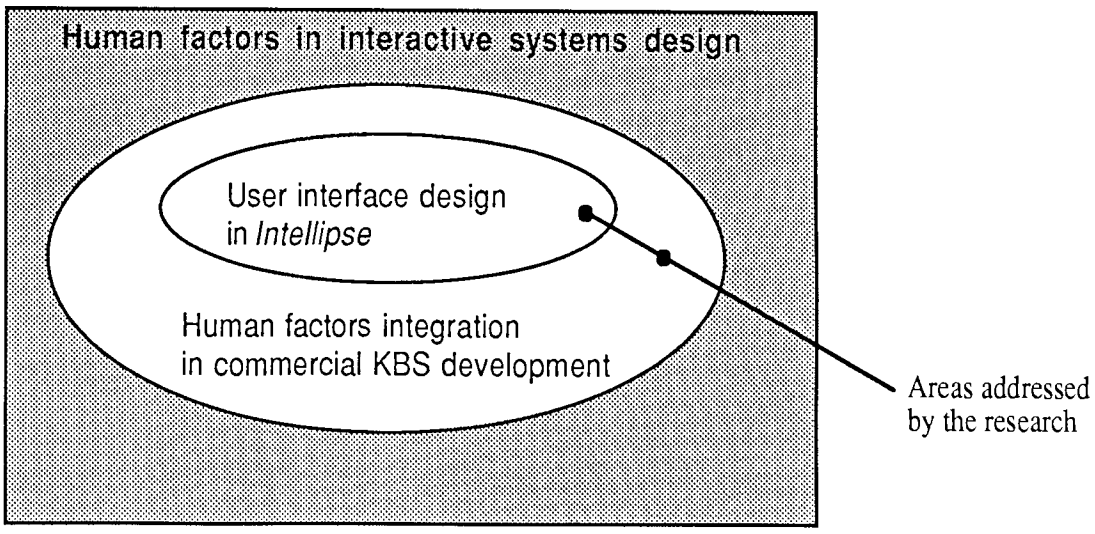


Figure 1.2. The research context

1.8. User interface design in Intellipse

At the outset the author's role and function at the specific project level were only described in high-level terms. Consequently, the first stage in his work was to better define the problem situation, and hence to state the aims and objectives in more depth. Whilst the initial situation which existed concerning the extent of a research problem definition is not uncommon in the experience of IHD collaborative projects, it might also be seen, with hindsight, as being a direct indicator of the level of integration which existed in the two areas of MMI and KBS at that time. This would be in keeping with the early experience within Alvey concerning the slow uptake of MMI projects within the initiative as a whole (as mentioned earlier in section 1.7.).

Intellipse had begun in October 1985, and had been running for 9 months, when the author joined the project in June 1986. On first joining the project he became involved in the work to further progress a prototype system, called *Advisor* which was then under development. This system was an advice giving system covering the design phase of DP system development. The author's involvement with *Advisor* acted as an early case study of a KBS development, and was used to provide the basis for the direction of the subsequent research. As well as raising specific issues of its own, involvement in the work on *Advisor* was instrumental in highlighting some of the problematic issues identified in the existing human factors literature. Together the two sources of information gave both a background to some of the general problems of integrating human factors into interactive systems design in general, and an indication of problems associated with the development of knowledge-based systems in particular.

1.9. Human factors integration in commercial KBS development

> "The HCI *(human-computer interaction)* researcher who believes his or her findings to be of practical relevance has... to consider the interface between researcher and practitioner as well as that between system and user." (Hammond et al. 1983; p 1)

The above extract captures, succinctly, the essence of the approach which was adopted to meet the second objective in the overall research plan. In formulating the approach it was felt that in order to increase the awareness of important human factors issues, and to successfully integrate human factors techniques in KBS design, there must first be an understanding of the environments in which such design is taking place. The nature of such environments, in particular the constraints which exist within them, will have some considerable bearing upon the impact of particular design methods, such as those from human factors. The issue raised in the extract above has been echoed elsewhere in the

human factors community and, as shall be highlighted throughout the thesis, is now increasingly being seen as an important issue. Having developed an understanding of the 'target' environment, that is commercial KBS development, the next stage in the research involved the development and evaluation of methods for improving user interface design. Both development and evaluation of these methods was conducted in close association with representatives of the target environment. The extract below, taken from the Alvey Human Interface Club response to the post-Alvey, IT'86 (Bide) report, endorses the need for continuous reference to the target environment:-

> "The involvement of users in HI *(human interface)* research is to ensure that the deliverables of HI research are what is required, and in the right form, i.e. the deliverables must be receivable." (Underwood 1987; p 26)

In the context of this work, KBS developers can be thought of as potential "users" of human factors methods. The *Intellipse* project was the primary source for the work and provided one set of representative KBS developers. However, to ensure a more representative view of commercial KBS design practice, and to consolidate the work within *Intellipse*, further contacts were also made with others involved in KBS development.

1.10. The research plan

Taking into account the objectives if the research, and having identified some of the key issues associated with these objectives, a plan for the work was formulated. This plan, which involved several stages, is set out in Figure 1.3., at the end of this chapter.

1.11. Chapter outline of the thesis

Below is an outline of the chapters which will now follow, and which will describe the overall research in detail:

- *Chapter Two* describes human factors principles and methods in the general context of interactive software design. Existing approaches to the integration of human factors into software development are discussed and problems associated with these approaches are described.
- *Chapter Three* looks at the potential for human factors in the specific context of KBS development. Facets of present KBS development methodology are discussed, along with the implications that these facets have upon the effective integration of human factors methods in KBS design.
- *Chapter Four* presents the first of the case studies, based upon the development

of the *Advisor* system - the first module to be developed within the *Intellipse* project.

- *Chapter Five* outlines the objectives behind the subsequent research approach adopted. Findings from the first stage of this work are reported. The first stage involved a series of interviews conducted with industrial and commercial developers of KBS.
- *Chapter Six* involves a description of a set of methods, or procedures, which were developed to tackle some of the key problems related to user interface design practice that had been identified in the early work. There is also a discussion of the rationale behind the approach taken, with reference to alternative approaches which might have been adopted.
- *Chapter Seven* and Chapter *Eight* both cover further case studies of KBS development projects. Central to these case studies was the application of the methods developed previously. *Chapter Seven* describes the involvement on a KBS project (entitled IBS) carried out by the Trustee Savings Bank. The SAM project, which involved the development of a KBS to build large databases, is described in *Chapter Eight.*
- *Chapter Nine* brings together all of the lessons and experiences which emerged from the overall research. A discussion of the specific issues which arose from the work is given alongside more general statements concerning the integration of human factors within KBS development.
- *Chapter Ten* is the final chapter, and presents the overall conclusions from the research work along with recommendations for the future progression of work in this area.

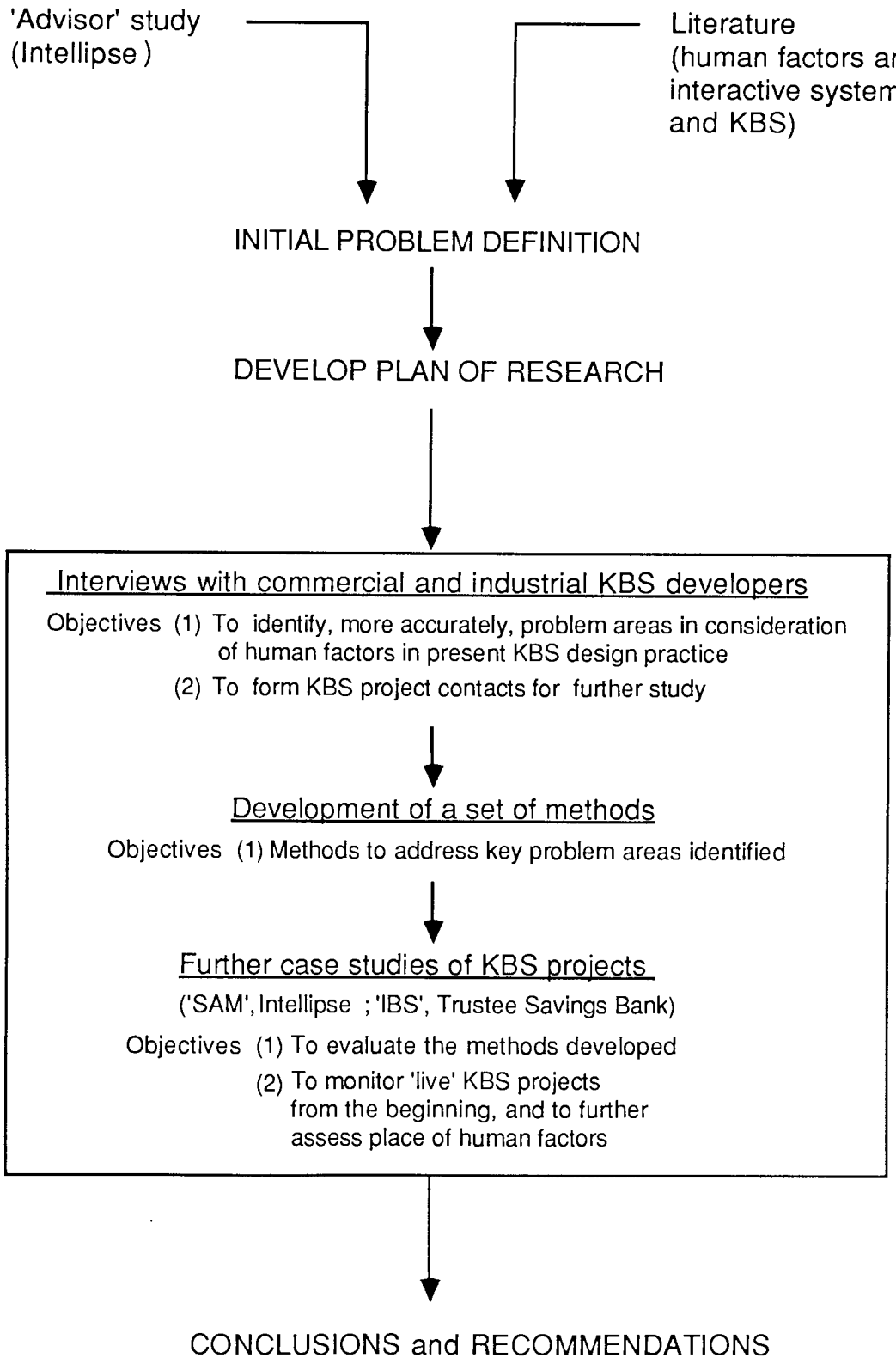


Figure 1.3. The stages of the research and their objectives

Chapter Two

Human Factors in Interactive System Design

Outline

At the macro level, the research described in this thesis centres upon the issue of the integration of human factors in the development of KBS. In this chapter, and in the following chapter, the two essential areas, of human factors and KBS, are reviewed in turn. This chapter deals with the major human factors issues, and concerns, which pertain to the user interface design of interactive computer systems in general - a class of systems of which KBS are a member. The following chapter will then give an account of KBS development and human factors issues which are related to those systems in particular. This present chapter discusses the key issues and aims of human factors in interactive computer system design. It is not concerned with a deeper level (psychological) analysis of design practice but is concerned with the 'pragmatics' of integrating human factors perspectives and approaches in the practical design of computer systems. This focus reflects the context of the research work carried out, both within Intellipse and through subsequent contact with other KBS projects. The chapter begins with a look at the scope of the term 'user interface'. It then progresses by describing the general philosophy, and approaches in human factors whose primary objective is the delivery of 'usable', as well as 'functional' computer systems. Following a description of more specific user interface design techniques and guidance, the problems of achieving actual human factors application and integration in 'real-world' design environments are then addressed.

2.1. Growth of human factors in system design

> "If we go back twenty years...The relationship between the user and the computer was sufficiently remote that it should be likened more to a literary correspondence than to a conversational dialogue." (Card et al. 1983; p 4)

The above quote from Card, Moran and Newell's book 'The Psychology of Human-Computer Interaction' neatly sums up the technological trend which has taken place within computing and which has given rise to increasingly interactive computer systems. With advancements in computer technology, particularly the increase in computer memory and processing power, there has been a movement away from punch-card, batch-mode processing to on-line computing, resulting in a more 'conversational' relationship between user and computer (Maguire 1982). The trend has been accompanied by an increasingly widespread use of computers by people who are not necessarily 'computer-literate'. They are, for example, being used by accountants for economic analysis, by secretaries for word-processing, and by engineers for design.

The facility for increased and more sophisticated user computer interaction - 'conversational computing' - is not, however, dependent merely upon technological

developments. The potential for this style of interaction is as reliant upon a firm theoretical basis, that is a conceptual understanding, of the process of user-computer interaction as it is upon the development of enabling (hardware and software) computer technology. The increased importance of such an understanding, exposed largely by numerous implementations of inadequate user-computer interfaces, has led to a growing awareness of the 'human' issues which accompany the design and use of interactive systems. These developments have led to an ever increasing potential for contributions to the system design process from *human factors* specialists - specialists from areas such as ergonomics and cognitive psychology.

2.2. Towards working definitions of the human-computer interface and user interface design

The area of human factors is laden with terminology much of which can be criticised as being vague. The potential scope of terms such as the 'human-computer interface' and the 'user interface' is considerable, even when applied to the single domain of computer system design. Indeed, the whole area contains a plethora of terms, and acronyms, whose boundaries are not always clear; terms which include, MMI (man-machine interaction), HCI (human computer interface, or interaction), user interface and user-system interface. For the purposes of putting this research into context it is important to narrow down the scope of the terminology and arrive at a manageable and appropriate set of definitions. At the broadest level the above terminology, as applied to system design, can cover all aspects of the relationship between both individuals, and organisations', use of computer systems. Smith and Mosier (1986) note that the term 'user-system interface' is:-

> "... broadly defined to include all aspects of system design that affect system use." (p 2)

Such a definition does not limit its coverage to the technical issues of design and implementation. This definition would cover design issues involving any aspect of the association and use of computer systems by individual users and by groups of people, such as work organisations. At the wider level, encompassing both organisational and technical considerations of system design there emerges a close link between human factors and the 'socio-technical' approach, as described by Mumford (1983) :-

> "A socio-technical approach is one which recognises the interaction of technology and people and produces work systems which are both technically efficient and have social characteristics which lead to high job satisfaction" (p 10).

This chapter takes a more focused view on the human computer interface (HCI), and user interface design. Such a view reflects the focus of the research work as a whole. These elements are seen as two constituents of the generic area of human factors; in effect two elements which are particularly important in relation to human factors in interactive computer system design. As the terms 'HCI' and 'user interface' are deemed to be synonymous (at least in this work) I shall in future use the term 'user interface' or 'user interface design' rather than 'HCI' or 'HCI design'.

It is now important to describe the boundaries of the term user interface. A useful definition for the purposes of this work is one provided by Card et al (1983) which suggests that a key notion underlying the user (*computer*) interface is :-

> "...that the user and the computer engage in a communicative dialogue whose purpose is the accomplishment of some task." (p 4)

Such a definition is useful as it introduces the scope of user interface design which can be seen, at a high-level, as being concerned with two central activities :-

i. **Establishing the appropriate functional requirements of the main task which is to be conducted through the user-computer interaction.**
   For example, in the design of a word-processor system, it would be necessary to establish what functions are required by the user in order to achieve the task of producing a document. This task would be broken down into lower-level, sub-tasks, such as "create", "save", and "edit" and these functions again would have their own sub-tasks.

ii. **Establishing an appropriate form of task representation through the user-system dialogue - enabling effective communication between user and system.**
   In the example of the word processor system above, having established the various tasks and sub-tasks, a second stage of user interface design activity would be concerned with the method used to represent, or convey, the task of document production (i.e a computer representation of a document and actions upon that document, such as 'edit'). Dialogue design is an important element in engineering this representation. Dialogue design establishes the channel, or channels through which the user and system communicate in order to carry out a task. These communication channels include elements such as the system prompts for a command, or data item

entry; the method for command selection or data entry by the user; and, in the event of 'breakdown' in dialogue, error and help messages from system to user.

It is important to note that the above definition includes some element of the activity of establishing functional requirements of the system. A key to the relationship between functional requirements capture and user interface design, proposed here, lies in the use of the word 'appropriate'. It is used intentionally to emphasise the need to ensure that the functions provided by a system are 'appropriate' to the overall task as it is seen by the user. This issue stems from the fact that the user interface cannot be entirely independent from the underlying application and its functions - they both have some consequences upon, and must have (at some level) communication with, each other. (Edmonds 1982; Green 1985; Cockton 1987).

Having set some boundaries on what is meant by user interface design the following section will look at the 'human factors' role and perspectives which are pertinent to the activity of user interface design.

2.3. The general role of human factors in user interface design

From the descriptions and definitions given so far it may appear that user interface design is the sole concern of those in human factors, this however is clearly not the case. The majority of system design is carried out by software engineers and by computer scientists. The systems they produce invariably, and inevitably, have some form of user interface. The two user interface design activities described above, be carried out to some degree in all of this system design. So what is it that distinguishes the role of human factors in user interface design? In short the answer is one of emphasis - from the human factors perspective the importance of the user and user behaviour in the user-system scenario is utmost. The extract below, from a paper by Moran (1981a), highlights the importance of psychological issues of system use, and gives a definition of the user interface which neatly illustrates the human factors perspective:-

> "The designer usually thinks of the user interface as the terminal hardware plus the software that receives, interprets, and sends messages (and displays) to the user. This definition may satisfy the system designer and implementer, but it is inadequate from the user's point of view. Psychologically, the user interface is any part of the computer system that the user comes in contact with - either physically, perceptually, or conceptually." (Moran 1981a; p 5)

A common view of computer scientists and software engineers is that they are seen as 'computer-centric'. That is they are only interested in aspects of the computer system, such as programming, operating system, and processing power. The user is seen as essentially peripheral to the computer system. Such a bias is seen as significant contributor to the failures of computer systems in the past :-

> "A person writing a program... does so within a background of assumptions about how the program will be used and who will be interpreting its responses. ... Many of the biggest failures of mundane computer systems (management systems, inventory systems, etc) have come not because the system failed to do what the designers specified, but because the assumptions underlying that specification were not appropriate for the situation in which the program was used." (Winograd 1985)

The aim of human factors is to reorientate the 'computer science' perspective by placing much greater emphasis upon the user and user behaviour. A primary objective of user centred system design in human factors is to reduce the amount of inappropriate assumptions (see the above extract), made either implicitly or explicitly by designers of systems about their users.

The essence of the reorientation mentioned above is exemplified by Shneiderman (1980) in his discussion of 'Software Psychology'. He sees the focus on human issues as concentrating on the five main areas of software design (see Table 2.1a). He points out that the software psychologist, that is the person interested in the human factors of software design, whilst primarily dealing with these five key issues, is also aware (at least in principle) of issues concerning the computer constraints upon design (see Table 2.1b).

Table 2.1a Key human issues in system design †

i.	Ease of use
ii.	Simplicity in learning
iii.	Improved reliability
iv.	Reduced error frequency
v.	Enhanced user satisfaction

Table 2.1b Computer constraints on system design †

i.	Machine efficiency
ii.	Storage capacity
iii.	Hardware constraints

† From Shneiderman's 'Software Psychology' (1980)

The human factors perspective is probably best illustrated by describing the design philosophies which are encapsulated in the various design principles, techniques and guidelines that exist and are promoted by those in human factors. The following five sections of this chapter will contain a description of some of the key elements in this battery of human factors information. A detailed description of single principles or techniques will not be given, but instead the intention is to elucidate the main essence of the human factors perspective in system design, concentrating, in particular, on user interface design.

2.4. The importance of 'Usability'

A high-level issue in human factors and interactive system design concerns that of system *usability*. There is an increasing awareness that the usability of a system is an important aspect to be considered in system design. In a review of several empirical studies of the effects of usability on system use, Goodwin (1987) concludes that :-

> "... investing in usability is as important as investing in functionality. Failure to consider usability can lead to system failure. At best a system with poor usability will cost its users time and effort; at worst it will not be used at all, and its functions may be removed because their utility has not been demonstrated." (Goodwin 1987)

Commonly those in human factors see that there is an over emphasis by system designers on system 'functionality' without much consideration of usability. They argue that usability has a significant effect on whether a system will be used or not, and therefore whether the functionality which may have been built in to a system will ever be accessed. In essence, poor usability will obscure functionality. Eason (1984) suggests that the importance of usability will increase as systems become more sophisticated and can carry out more and more functions. He states that the gap between 'potential utility' of a system, that is the total system functionality, and it's 'actual utility' is largely determined by the extent of system usability.

Whilst a definition of a system's functionality is quite straightforward, usability is not so easily defined (Goodwin 1987). However, a clear and 'workable' definition of usability is required if it is expected that system designers can take actions to properly consider usability and to build it into their systems. Vague notions as to the importance of usability, and to the way that it can be achieved are unlikely to be sufficient in the practical construction of interactive systems. There is an awareness of this problem within human factors, and there is a movement to try and produce practical definitions of usability.

Shackel (1986) proposes a four characteristics for an operational definition of usability. The definition is 'operational' in that it is proposed that it can be used to generate 'usability goals' against which numerical values can be set. These goals, and their associated values (such as "use by 90% of the target range of managers, secretaries and professionals") should be set at an early stage in design, during system requirements specification. Although the complete definition is not given here, the four characteristics of Shackel's definition are given in Table 2.2. below. Originally each characteristic is expanded upon by Shackel in terms of its relationship to system use by some part of the intended user population.

Table 2.2. **Four characteristics of Usability**
(from Shackel 1986)

Effectiveness
Learnability
Flexibility
Attitude

2.5. Approaches to designing for Usability

As mentioned earlier, there is an awareness of the need to go beyond just a definition of usability - to operationalise, and make practical this concept in terms of real system design. Many attempts have been made by those in human factors to develop a set of design principles which designers can apply as part of the overall system design process. Table 2.3. outlines just three more recently documented sets of principles. They encapsulate many of the ideas that have been developed, and so I shall just elaborate on the rationale behind these three approaches.

Table 2.3. Three approaches to designing for Usability

Gould and Lewis (1985)	Shackel (1986)	Gould (1987)
Early Focus on Users and Tasks	User Centred Design	Early and Continual Focus on Users
Empirical Measurement	Participative Design	Integrated Design
Iterative Design	Experimental Design	Early and Continual User Testing
	Iterative Design	Iterative Design
	User Supportive Design	

There is a considerable level of consensus in all three approaches outlined above. In the main, the variation in the number and headings outlined results, from either a more developed division of what are seen as the basic principles, or from the use of different terminology. Together they highlight the human factors approach to interactive system design at the top-level, that is they are prescriptions for the approach and 'philosophies' which should pervade the entire system development process, from onset of a development to system delivery.

One important philosophy in all three approaches is that of user-centred design. At an early stage in design it is seen as crucial that designers thoroughly understand the potential users and their proposed tasks. Such an understanding can only be carried out by meeting directly with a set of users (through interview or observation). The user-centred philosophy should extend throughout the development process, being achieved by involving users as part of the design team (Participative Design) and by conducting continual user testing (Empirical Measurement and Experimental Design). User testing is to facilitate evaluations of the developing system, and is normally carried out using simulations or prototypes.

The function of continual user testing is generally to establish problems (for example user misconceptions or, bugs in the software) which can then be removed through redesign. This necessitates a high degree of iteration in design where a cycle of 'design, test and measure, and redesign' takes place. Iterative design is seen as a crucial, and unavoidable element of designing for usability generally, and indeed, as we shall see in the next section, is seen as an important element of user interface software design in particular. Gould (1987) describes three key requirements for iterative design - see Table 2.4.

Table 2.4. Key requirements for iterative design for usability

Identification of required changes
Ability to make changes
Willingness to make changes

The issue of iterative design, and of the requirements for it, are particularly pertinent when considering the practical application of these approaches by system designers. The issues which arise from iterative design will be discussed later in the chapter, when problems in the actual application of human factors are discussed.

Usability, and the approaches to designing it into systems, represent the central objective of human factors in the development of interactive systems - it is in a sense the ethos of human factors in this area. Usability is a general philosophy which, it is proposed, should pervade the entire development process. The intention is now to move from a description of the general philosophy to the specific with a description of some key tools (approaches, techniques and guidelines) that have been developed in an attempt to substantiate this philosophy. These are tools that are aimed at particular phases of the development process.

2.6. Iterative user interface design

Williges et al. (1987) reviewed a series of software interface design methods from which they developed a three-stage design process. In keeping with the general approach to usability, discussed earlier, the design process they describe is iterative. Similarly the process can be seen to encapsulate much of the approach to usability philosophy, but concentrates particularly on the software user interface - a focus which is appropriate to the research work to be described later in this thesis. The model is outlined in Figure 2.1., overleaf.

As well as again emphasising the iterative nature of user interface design, the process described by Williges and colleagues is useful in that it develops, at a more detailed level of design, a model for user-centred design. Of particular interest, is the distinction made between formative and summative evaluation. Formative evaluation refers to the evaluation of interface prototypes and is *'central to the iterative design process'*. Summative evaluation refers to the evaluation of the final system, where comparisons are made between the final system and either an alternative system or, in some cases a previous version of the same system.

Figure 2.1. A Three Stage Process for user interface design (Williges et al. 1987)

Despite the coherency, and potential usefulness of the model which is outlined,Williges himself adds a crucial rider, in a later paper, concerning the process model and its application:-

> "Such a user-centered approach to interface design requires that the designer, or design team, use appropriate tools to aid in the design process. Without such tools, iterative design is too cumbersome and too time consuming to be beneficial." (Williges 1987)

Once again, this is an important caution when we consider the practical application of such a process in the context of commercial or industrial computer system development, where development costs are a major issue. Issues concerning the practical application of human factors in these environments are to be discussed later in the chapter, and will be raised throughout the thesis, as these issues are central to the research work conducted.

Maintaining the movement in this chapter from describing general approaches in human factors towards (development phase-) specific techniques, the following two sections

will deal with two areas of user interface design where there is particular interest - the area of task analysis and user modelling, and the area of user interface design guidelines.

2.7. Task analysis and User Modelling for user interface design

Both the approaches to designing for usability, and the process model for software interface design that have been reviewed, place an emphasis on the need for early focus on users and their tasks. Without developing a clear understanding of users and the tasks which are to be performed through system use, there is likely to be a considerable gulf between the designers assumptions of system requirements, and what is actually required by users (cf. Winograd; section 2.3). Early focus on users and tasks helps the designer(s) to establish and specify the form of interaction that will be required and hence be able to specify the user interface requirements. Within human factors there has been much effort in developing task analysis techniques for this purpose - Table 2.5. lists some, but by no means all, of the major analysis techniques which have been developed. It is not the intention, here, to go into detail on particular analysis techniques but to draw out the central issues which are being addressed within task analysis as a whole.

Table 2.5. Exemplar Task Analysis Techniques

Command Language Grammar (CLG)	Moran 1981(b)
User Device Model (UDM)	Kieras and Polson 1985
GOMS model	Card et al. 1983
External-Internal Task Mapping Analysis (ETIT)	Moran 1983
Task Analysis for Knowledge Descriptions (TAKD)	Johnson et al. 1984
Task Action Grammars	Payne 1984

Task analysis techniques are aimed at capturing and representing the important elements of the interaction between user and system, that is the characteristics of the task, the system and the user's knowledge of both task and system (Hammond et al. 1987). Establishing the user's model of both task and system is crucial. As was stated in section 2.3, an important aspect of the user interface centres upon the users conceptualisation of the whole system (c.f. Moran Section 2.3). An important idea within the study of

human-computer interaction, and one which underpins most of the task analysis techniques in this area, stems from the notion of 'mental models' in cognitive psychology. At a fairly crude level a mental model can be described as a person's internal mental representation of a phenomenon, which allows that person to manipulate, in some way, that phenomenon (Johnson-Laird 1985). That ability for manipulation, can be seen to constitute a form of understanding. In the context of human-computer interaction and user interface design an important and allied idea is that of the 'user's conceptual model of the system, or *user model* -

> "...the *user's conceptual model* of the system - is an integral part of the user interface. The conceptual model is the knowledge that organises how the system works and how it can be used to accomplish tasks...The conceptual model must be taught to the user and must be reinforced by the behaviour of the system." (Moran 1981a; p 5)

The importance, ascribed by some, to the user model is exemplified by the work carried out in designing the user interface to the Xerox 8010 Star Information System (Canfield-Smith et al 1982). In the design of the Star system the definition of the user's conceptual model of the system was seen as so crucial that it preceeded full functional specification, and hardware and software implementation. The central element of this conceptual model, as it was eventually defined, was the 'Desktop' model which, as it transpired is a popular model for other systems (for example, the Apple Macintosh). It was estimated that thirty man years effort was expended, in total, on the user interface to the Star. Whilst such effort is probably unprecedented in relation to most interactive system development, the benefits accrued from putting effort into user modelling and the user interface as a whole proved considerable.

The notion of user models, and their representation through analysis techniques like those listed in Table 2.5., are seen as important tools for understanding and predicting user performance on a system (Williges 1987). Due to the variety of analysis techniques it is not easy to establish where individual techniques fit into the system development process. This is exacerbated by the fact that most of the techniques are still in their infancy, in terms of their applicability to practical design situations. One can, however, establish some distinction between the techniques concerning their role in the development process (Bellotti 1988; Wilson 1988). The techniques can be divided into ones which act as user interface methodologies and guide the user interface design process (for example Moran's CLG); and ones which act as evaluation techniques, predicting the complexity of, and likely user performance on, a given interface design (for example Kieras and Polson's UDM). This is a concise, high-level, distinction of the

differences between various classes of model. What is important, however, is to see that all of the models are an attempt to focus on users and their conceptual model at an early stage in the design of system. At the same time all of these techniques have some basis, to varying levels of degree, in a psychological understanding of user and computer interaction (Wilson et al. 1986).

Bellotti (1988) summarises the potential benefits associated with the various task analysis techniques for user interface design. A selection of these are described in Table 2.6.

Table 2.6. Potential benefits of Task Analysis techniques (from Bellotti 1988, *abridged*)

Thoroughness - By helping a designer to deal with user issues and identify inconsistencies and inadequacies in design.
Minimalisation of negative transfer effects - By helping to identify the knowledge that users bring to the use of a system, and the effect that this prior knowledge will have on their use of the system.
Highlighting complexity - By helping to establish difficulties classes of users might have with an interface and which would have a negative effect on performance.
Provide evaluation measures and also ensure earlier evaluation - By providing measures of performance (e.g., time to enter a command) which can be evaluated. Also the techniques can be used to evaluate design specifications (rather than software), thereby enabling earlier evaluation.

Despite such potential benefits as those listed in Table 2.6., many of the techniques are yet to be rigorously tested in 'real-world' system design. The nature of these techniques, in particular their psychological basis, has meant that there remain questions over the actual validity of the techniques - that is, whether or not they are based upon sound psychological models. There is still a need to validate these models (Williges 1987). At the same time many of the techniques have only been tested in restricted application domains (such as the design of text-editors) and there remains a question over the viability of some techniques in other application contexts. Wilson et al. (1986) raise the question of the usability of the techniques themselves, noting that some techniques are significantly complex to the extent that they require some training in cognitive science in order to gain a sound understanding of the underlying principles. These kinds of issues

will inevitably have an effect upon the up-take and application of the various task analyses by system designers.

2.8. User interface design guidelines

The types of human factors knowledge considered so far in this chapter have largely been concerned with methodology, that is with procedural approaches to user interface design. Within human factors there also exists a great deal of information on detailed design features of user interface design, concentrating on the more immediate interaction (dialogue) of the user at the computer terminal. In general this information has arisen over the years from experimental study and design experience, and has resulted in the formulation of design guidelines. These guidelines are aimed at the stage of design where actual interface software is being produced - that is where a specification for user-system dialogue is being implemented.

A recent and comprehensive set of guidelines for interactive software design has been produced by Smith and Mosier (1986). This set is an extension and synthesis of earlier sets of guidelines, many of which were initially developed as 'in-house' design guidelines (for example, Engel and Granda's (IBM) - 'Guidelines for Man/Display Interfaces', 1975). As an illustration of the extensiveness of Smith and Mosier's report in the 1986 version has a total of 944 guidelines. To show the general coverage of these guidelines the contents from the report is shown in Table 2.7.

Table 2.7. Contents of Smith and Mosier's User Interface Software guidelines

Data Entry
Data Display
Sequence Control
User Guidance
Data Transmission
Data Protection

Guidelines provide an identifiable and potentially useful information source from human factors, which can contribute to the design of usable interface software. However, the application of guidelines to particular design instances is not straightforward, and this can inhibit their use by designers (Maguire 1982). The problem lies in the need to tailor

guidelines to a specific design instance, that is the need to select appropriate guidelines, interpret them and then modify them to suit the particular design (Smith 1986; Marshall et al. 1987). Smith and Mosier are aware of the need for appropriate selection from their own guidelines, stating in the preface to their report, that only a subset of the guidelines in the report would be pertinent to a specific design. They endeavour to reduce the problem of generality in guidelines by giving example scenarios of where the application of a guideline would be appropriate.

Another problem with guidelines concerns apparent contradictions that can arise between them (Maguire 1982). Such contradictions do arise where differences in experimental design have contributed to conflicting results from which the guidelines have been generated. For example, in a survey of the literature on the use of various input devices, Milner (1988) found that trying to derive guidelines for the use of particular devices was extremely difficult due to the differences in variables (such as user and task type) which were inherent in the empirical studies.

2.9. The practical application of human factors

Coverage has so far focussed on the content of human factors information. There has also been some description of the problems, and cautions, associated with specific elements of this information. The overall approaches described, have in a sense been the 'human factors ideal' - what has been proposed by human factors specialists as the way to design usable interactive systems in general, and 'good' user interfaces in particular. Key elements of this 'ideal' are user participation, and an iterative design cycle within which evaluation forms a central role. There is some considerable evidence to suggest that this 'ideal' for design is not being achieved. This section will deal with a review of some of the general problems which pertain to the practical application of human factors information in the 'real-world' context of commercial and industrial system design.

In a recent report to the Alvey Human Interface Club, Klein and Newman (1987) reported that in survey work they had conducted for the Alvey Directorate they had noted a distinct lack of user interface design activity. Findings of this nature have has been echoed elsewhere. In a survey of 201 system designers, Smith and Mosier (1984) reported on overall inadequacies in user interface design practice. An inadequate approach to user requirements capture and documentation was reported by the majority of those surveyed, and there was also a reported lack of guideline usage. This was despite the agreement amongst the designers, that on average 30 to 35 per cent of system software developed was actually concerned with the construction of user interface software.

Various other studies of system designers and design practice have been carried out. Gould and Lewis (1985) carried out an interesting study on designers' views of usability design principles. They noted that when their usability principles (cf. Table 2.3., Section 2.5.) were recommended to designers, a common reaction was that they were 'obvious', in effect common sense. They subsequently conducted a study where they asked designers to state the sequence of activities which they would go about in designing and evaluating a computer system for end users. The objective was to see just how intuitive the usability design principles actually were. On the whole designers did not report most of the usability design principles, a finding which suggested that such principles were not as intuitive as had been reported *post hoc*.

From a series of semi-structured interviews conducted with five system designers Hammond et al. (1983) also report the poor utilisation of human factors in design. It was suggested (from a small sample size, as the authors point out) that this was not so much because the designers themselves were unaware of the appropriate need for human factors, but that there were other constraints at play - such as organisational and resource constraints. A similar study of commercial design practice (involving formal interviews with four designers) was carried out by Bellotti (1988). She reports that her study pointed to a distinct lack of application of user interface design and evaluation techniques in commercial practice; techniques such as those for task analysis described in section 2.8. of this chapter.

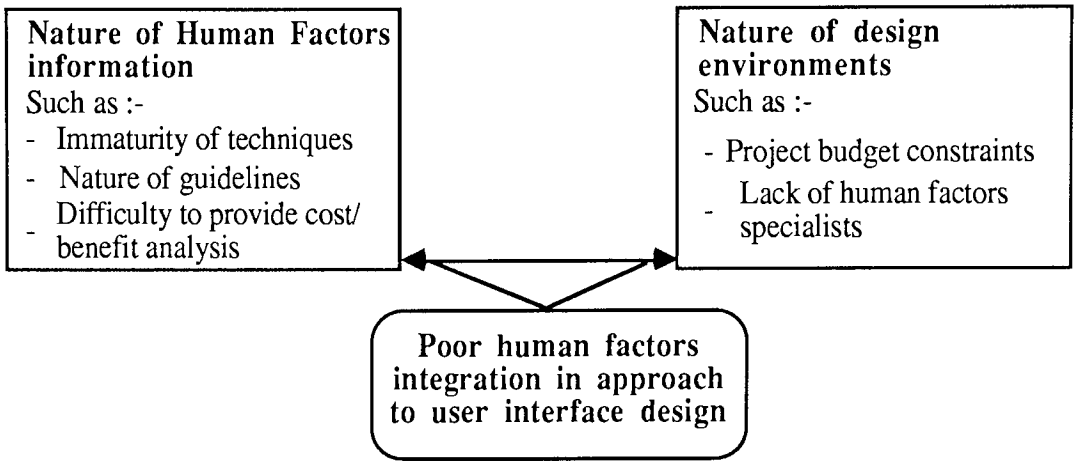
2.10. Some determinants of poor human factors integration

From what has been described in this chapter it is possible to make some postulations concerning determinants of the situation which exists in 'real-world' user interface design practice, as outlined in the previous section. At the top level two main determining factors for this situation can be isolated. One element concerns the nature of human factors information; the second concerns the nature of the actual environment in which design is taking place. The present situation that exists in human factors integration results from some interplay between these two elements illustrated in Figure 2.2., overleaf.

Firstly, one reason for the apparent lack of human factors application lies in the nature of human factors information as it exists at present. Further development of human factors techniques, such as task analysis and user modelling techniques are required before their application becomes more widespread. This means development of techniques in terms of their internal validity (i.e. the models of interaction upon which they are based) and in terms of the actual usability of the techniques by system designers in general. The

problem of generality of design guidelines, and the need for some form of guideline tailoring, to specific design instances.

Figure 2.2. Determinants of poor human factors integration in interactive system design



The second determinant relates to the design environment and the constraints inherent within it. Development time is a crucial factor in commercial system design, and the more rigorous approaches to design which are prescribed by those in human factors in a bid to ensure system usability inevitably lengthen project development time. A further constraint lies in the lack of human factors specialists and trained user interface designers (Klein and Newman 1987). This shortage means that the process of system development is being dictated, in the main, by those who do not have a sufficient appreciation of human factors or the requisite design skills to ensure successful human factors integration.

An extract from a paper by Norman (1986) nicely sums up the the two 'inhibitory' elements just described:-

> "Any real system is the result of a series of tradeoffs that balance one design decision against another, that take into account time, effort, and expense. Almost always the benefits of a design decision along one dimension lead to deficits along some other dimension. The designer must consider the wide class of users, the physical limitations, the constraints caused by time and economics, and the limitations of the technology. Moreover, the science and engineering principles necessary for proper design of the interface do not yet exist." (Norman 1986; p 54)

The resulting problem is largely one of cost-benefit. There is a need to convince those who at present dictate the approach to system design (for example, software engineers, computer scientists, project managers) that human factors, and 'good' user interface design are important and that cutting corners on these areas, although reducing development cost in the short term, will prove more expensive in the long term. Long-term expense will be incurred through the production of redundant, or largely unusable systems, which require considerable and costly redesign. Providing a cost-benefit analysis for human factors is, however, not easy, due in part to the intangible nature of benefits which accrue (Robertson 1987). This is seen as an important issue, however, and a recent paper by Mantei and Teorey (1988) attempts to provide some means for conducting cost-benefit analysis, at a quite detailed level, by comparing the cost of carrying out human factors procedures. They measure the cost of procedures such as task analysis and evaluation against the cost of system errors, and poor system utilisation and performance. This kind of approach is at present just in its infancy, however the objective behind providing a cost-benefit analysis is an extremely important one.

2.11. Alternative routes to human factors integration

Developing better cost-benefit analysis methods is one area which is being developed to better promote, and 'market' human factors. Another way is to integrate human factors principles, like those discussed, into traditional system design methodologies which exist. At present the human factors approaches, such as those for usability design tend to sit outside existing traditional methodologies and attempts are being made to integrate them (for example, Damodaran 1988). A major problem lies in the fact that the traditional methodologies do not allow for user interface design (Klein and Newman 1987); something which can possibly be attributed to the apparent asynchrony between the highly iterative nature of user interface design and the nature of the traditional 'life-cycle' development model which attempts to keep iteration to a minimum. As will be seen later in the thesis, the same issue does not hold in exactly the same way for the integration of human factors within KBS development, where the KBS development life-cycle is itself highly iterative in nature. This issue will be elaborated upon later in the thesis.

Another route to human factors integration and improved user interface design is through the design and construction of software support tools to aid the design process. For example Smith (1986) suggests the need for a computer-based tool which would translate general design guidelines into specific rules. Christie and Gardiner (1987) suggest the idea of building a database of designs (storing past designs), including

generic and organisation-specific designs. This database would be integrated within an overall design support environment. The idea of a computer-based interface design support tool, for the development of IT products, is being researched as part of a large collaborative European project called HUFIT - Human Factors in Information Technology - (Galer and Russell 1987). As a part of the HUFIT project the architecture for a support tool, known as INTUIT, has been described. The objective of this tool is to provide semi-automated, decision support for the designer *'by use of the skills and experience of a human factors expert in the development of the user interface'* (Galer and Russell 1987). The development of tools in this area, such as INTUIT, is however, likely to be fairly long-term (Atwood 1984). Before automated support tools can be built and successfully introduced into design practice there is first a need for a clearer understanding of the 'manual' process required to achieve 'good' user interface design. The issue of such tools for human factors will be raised later in the thesis in Chapter 6, as the development of an automated tool was an alternative that was considered in the direct context of this research.

Rouse (1987) agrees with the need for support tools alongside improved methods, stating that there is a need for human factors specialists to develop *'an understanding of synthesis methods and tools, rather than solely experimental techniques'*. However, he also adds the corollary that such methods and tools should take into account *'the interplay of the behavioral, organizational, technological and economic factors that affect design decision making'*.What he is suggesting is that all support tools and techniques for user interface design must be based on a firm understanding of not just the practice of user interface design, but also of the environments in which overall system design is taking place. This reiterates the points made earlier in section 2.10.

2.12. Concluding remarks

The chapter has dealt with major facets of human factors and its integration in user interface design. It has by no means been exhaustive, but the objective has been to describe both the potential human factors contribution to interactive system design, and the problems in realising that potential in the practical design setting. What has been highlighted here, is that a mismatch exists between *potential* and *actual* human factors application. Such a realisation is important in the context of this work. Indeed by highlighting the mismatch, the general context of the research work in this thesis has been laid down. The next chapter will take a similar approach to that taken in this chapter, looking however, at the specific context of the research. To that end it will look at the relationship between, and integration of, human factors in user interface design within KBS development.

Chapter Three

Knowledge Based Systems and Human Factors

Outline

As was discussed in the previous chapter it is proposed that successful human factors integration in the design of computer systems is contingent upon a careful consideration of many elements of the design environment and the issues which affect design decisions. For this purpose, the chapter looks at the background to KBS and KBS development, highlighting elements which will have some bearing on overall design practice and hence will have some bearing on user interface design practice - the particular focus of this thesis. Given this focus, the overview provided in this chapter also considers the potential human factors input to KBS 'end-user' interface design, alongside a review of some of the problematic issues in 'realising' this potential which have been reported by others. Emphasis is put on end-user' here, to differentiate between this interface and the system developer's (or, knowledge engineer's) interface. The latter interface is not the focus of this research.

3.1. Background to KBS

'Knowledge-based' or 'expert systems', have stemmed from research carried out in the field of artificial intelligence. Winograd and Flores (1987) give a condensed history of AI and the emergence of commercial interest in knowledge-based systems. They state that up until the mid-1970's AI research work had two main goals; firstly to extend the capabilities of computers, in terms of the types of problems and tasks that systems could address; secondly to develop a greater understanding of human intelligence, through computer-based cognitive simulation. This situation has changed more recently, with AI researchers, aware of the limitations of their findings concentrating their efforts more in the area of cognitive simulation. However, around the same time there did emerge considerable commercial interest in applying some of the techniques which had already been developed within AI. This interest centred upon the potential application of knowledge based systems.

Industrial and commercial interest in the application of KBS is growing rapidly. The United States is by far the largest investor in the technology, however activity in the UK is also relatively high. KBS featured as one of the four main streams within the Alvey initiative, with £40 million, out of the total funding of £350 million invested over the five years (Alvey Annual Report 1985). In 1986 the total expenditure on KBS in the UK was estimated to be £58 million (Ovum Ltd 1988).

3.2. Definitions for KBS

Any attempt to provide a concise definition for the term 'knowledge-based system' (KBS) is very problematic, and probably doomed to failure. The reason for this is largely attributable to the history of KBS 'technology' as outlined in the introductory section. As those people in academic AI, and those in commerce have held differing perspectives and objectives, concerning KBS, so numerous definitions have emerged, mirroring these varying objectives. This state of affairs has also resulted in the confusion over the various, although largely synonymous, terms which now exist. For example, the term 'knowledge-based systems' is seen as synonymous with the term 'expert system' (and as a result these two terms might during the course of the thesis be used interchangeably). Similarly there is little, or no distinction, between the term KBS and the related term IKBS (Intelligent knowledge based system) as used, for example in the Alvey initiative.

Having suggested that concise definitions are impractical, two previous attempts at encapsulating the essence of KBS are given below. This is for the purposes of introducing a preliminary overview of the area. The first definition comes from the British Computer Society's Specialist Interest Group on Expert Systems:-

> "...an expert system is a computing system which embodies organised human knowledge concerning some specific area of expertise, sufficient to perform as a skillful and cost-effective consultant." (cf. CCTA 1986; p 1)

A similar, but more slightly detailed, definition is given by Welbank (cf Hart 1986) :-

> "An expert system is a program which has a wide base of knowledge in a restricted domain, and uses complex inferential reasoning to perform tasks which a human expert could do." (in Hart 1986; p 19)

These two definitions provide the essence of the nature of KBS. They make reference to the association of KBS to human experts and also to the fact that the systems are aimed at restricted domains. Hayes-Roth (1984) states that expert systems are aimed at capturing *'the value of domain specific knowledge for solving significant problems'*. The two, now 'classic', instances of expert systems, MYCIN (Shortliffe 1976), and R1 or XCON (McDermott 1982), exemplify the central features contained in these definitions. The MYCIN system was concerned with the diagnosis of a particular set of blood diseases - here the human expert around which the system was based was the medical physician, or doctor. R1, or XCON as it later became known, was a system to help DEC (Digital Equipment Corporation) computer engineers configure VAX 11/780 computer hardware.

Giving these definitions in isolation is in itself limited, and possibly misleading. On their own they imply that a number of systems are in existence which conform to the characteristics outlined. There is still some considerable debate over the number of 'working' expert systems in existence (for example, Winograd and Flores 1987), that is the number of systems that are in regular, operational, 'everyday' use in the same way that other conventional systems are used. Many of the systems are still undergoing development and are not ready for full use (Buchanan 1986). A more detailed description of the present state of KBS development will be given in a later section on the commercial development and application of KBS. At the moment, however, it is worth adding that any definitions for expert systems are more accurately stated as 'goals' of expert systems work (Brachman et al. 1983), rather than a statement of what has already been achieved in the area.

To progress from these definitions, and to get a clearer pictures of what KBS are trying to achieve, it is worth looking at some of the main features which distinguish KBS from conventional systems, such as data processing systems. Waterman (1986) states that the basic difference between expert systems and conventional systems is that expert systems manipulate knowledge, whilst conventional systems manipulate data. In this context, knowledge can be seen as a combination of facts, beliefs, and heuristics Hayes-Roth et al. 1983). Waterman goes on to describe four main characteristics which distinguish expert systems from conventional programs (see Table 3.1).

Table 3.1. Four characteristics of expert systems
(from Waterman 1986)

Contained within these characteristics is the expert systems use of heuristic rules (well-informed judgments or 'rules of thumb') as opposed to algorithms (fixed rules with a determined outcome). The use of heuristics is an important feature of expert systems and

their use tends to create other system requirements, in particular those classified above, under the heading of 'Self-knowledge'. There is a significant requirement for explanation in expert systems, due to the types of problems, or tasks, that the systems are dealing with and the heuristic knowledge that is being used to tackle these problems. As Waterman states :-

> "While conventional programs are designed to produce the correct answer every time, expert systems are designed to behave like experts, usually producing correct answers but sometimes producing incorrect ones." (Waterman 1986; p 18)

Given the nature of expert systems, it is important therefore that the systems have some facility for explaining a judgment, solution, or diagnosis. Buchanan (1986) calls this dimension of expert systems "Understandability" - the requirement of the system to explain its reasoning. It needs to explain its reasoning, or at least allow interrogation to access the process by which it arrived at a judgment, for two reasons. Firstly, the system developer needs to be able to analyse the system's reasoning in order to make necessary changes and improvements to the system (such as, debugging and maintenance). Maintenance, updating the system's knowledge about the domain, is another important feature of expert systems. Because of the type of tasks that expert systems are aimed at tackling, developers need to make changes to the systems knowledge base. The system's reasoning therefore needs to be fairly 'transparent' to the developer, so that it can be updated and changed without causing significant, problematic side-effects (d'Agapeyeff 1984). The need for up-dating and maintenance is evidenced in the XCON system, mentioned earlier, which since its inception is constantly being augmented to cope with a changing domain; in this case changes in available hardware for configuration (Durham 1987). As well as 'understandability' for the developer, the system also needs to provide explanation of its reasoning to the eventual end-user of the system who, it is intended, will utilise the information provided. For the user to be confident to use the system's output, and in the particular case of expert system training systems, to be able to learn from the system, he, or she must be provided with some facility for explanation when required. These user interface issues of expert system explanation will be covered in more depth in later sections of this chapter.

3.3. Types of KBS

Apart from reference to two particular KBS applications, the tasks which KBS address have so far only been dealt with in vague terms. At the highest level KBS are aids to problem-solving and are aimed at providing support, at varying levels of intervention, in

problem-solving activities in which humans are involved. Hayes-Roth et al. (1983) outline ten 'generic' categories to which KBS are applied, these are given in Table 3.2., below.

Table 3.2. Categories of KBS application

Interpretation
Prediction
Diagnosis
Design
Planning
Monitoring
Debugging
Repair
Instruction
Control

Madni (1988) distinguishes between two categories of problem-solving into which the type of KBS application seen in the above table fall. These are *analytic* and *synthetic*. Analytic orientated KBS deal with such problems as diagnosis, classification or debugging. The essence of this form of problem solving is to narrow down the possible set of solutions (or in AI terms, reduce the 'search space') in a domain and arrive at a solution, or reduced set of possible solutions. An example of this would be a electronic fault diagnosis system which aims to locate the cause of a fault on an electronic circuit. Synthetic orientated KBS are involved in activities such as design, planning or repair. The essence of this form of problem solving is the generation of a solution or set of solutions from a basic set of primitives, or 'building blocks'. An example of this is a configuration system, like XCON mentioned earlier, which constructs an artifact (such as a computer hardware system) from a set of parts.

There is one further categorisation of expert systems which should be made. This is the categorisation between the *consultative* and *autonomous* role of expert systems (Fish 1988; Madni 1988). Systems which act in a consultative role are those which interact with the human user. Those systems which act in an autonomous role are those which do not interact with users but with another component of a total system, usually another computer system (for example, as part of a process control plant). The division between autonomy and consultation is not rigid, as many expert systems involve elements of both these roles, that is they interact with users and with other computer systems. However, in terms of the human factors aspects of KBS which are central to this thesis, it is those systems that, at some level interact with users that are of interest. Consultative systems can further be divided into two categories Madni (1988). Firstly there are consultative

systems which aid, or advise, the user in a problem-solving task. Secondly there are expert training, or tutoring systems which aim to teach a novice, or less expert person, some skill and level of expertise in a given domain. There will be a further discussion of these systems in a later section of this chapter when the human factors issues related to KBS are described in detail.

3.4. The basic KBS architecture

The architecture of consultative KBS consists of three main elements; a *knowledge base*, an *inference engine*, and a *user interface* (at this level the user interface should be thought of in a narrow sense - the computer terminal interface to the user). These elements are highlighted in Figure 3.2. Each will now be dealt with in turn.

The knowledge base of a KBS is the central store of information about objects and relationships between objects that exist in the application domain. This information is a set of facts and rules which make up the domain specific knowledge (see the schematic view of the knowledge base given in Figure 3.2.). This information can be stored in the knowledge base in various different formalisms. These formalisms are known as knowledge representations. Various knowledge representations have been developed in AI research , such as 'Semantic Networks' (cf. Woods 1975), 'Frames' (Minsky 1975), 'Scripts' (see Schank and Abelson 1977), and Production, or 'IF-THEN' rules (see Barr and Feigenbaum 1981). Within KBS a common form of knowledge representation is 'IF-THEN' rules, which have the following basic structure:-

IF <condition >
THEN < action >

which as an example might be instantiated as :-

IF person X has an umbrella
THEN ASK "Is it raining outside?"

The knowledge base correlates incoming information about the state of the world (which comes from an external source such as an end-user) with its current internal state. If this information matches a condition held in the 'IF' clause of a rule then this 'pattern match' will result in the action clause of a rule will be fired.

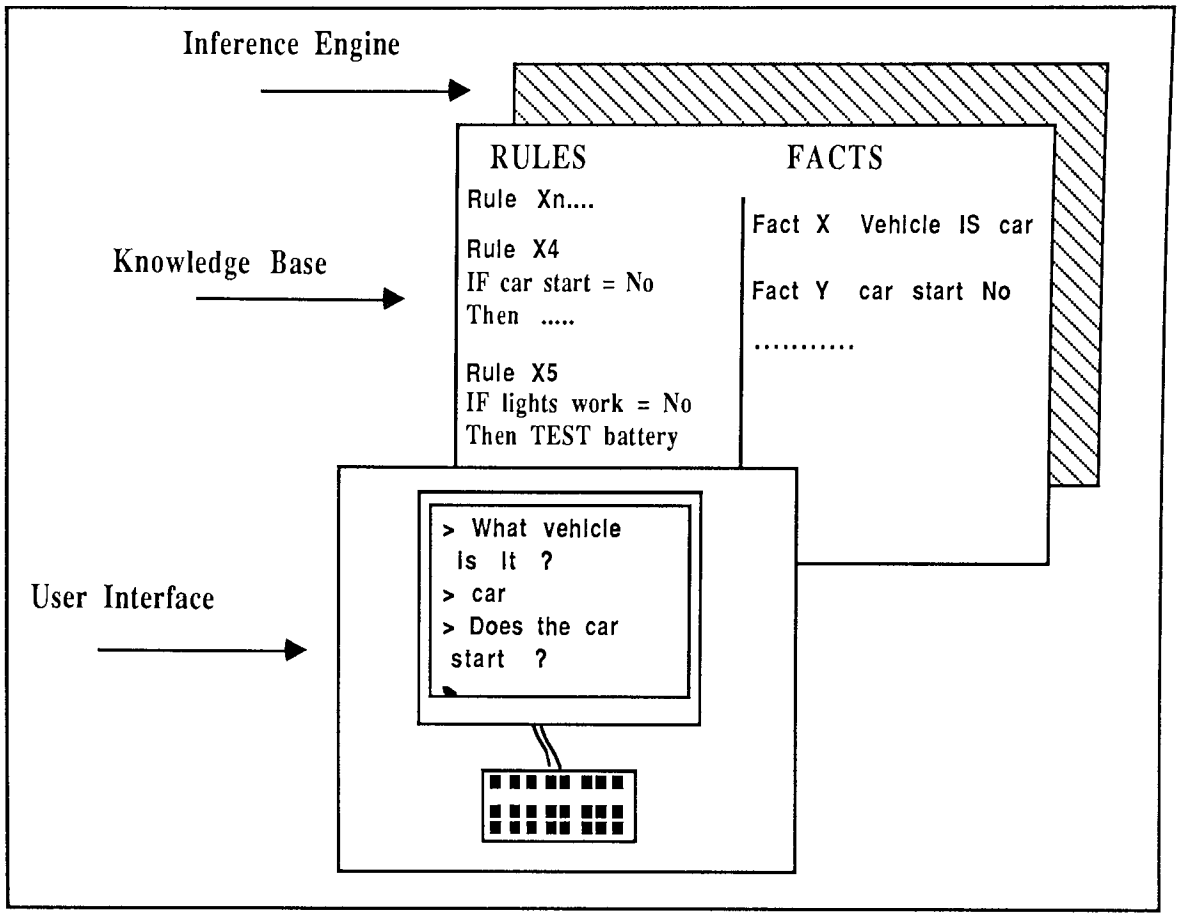


Figure 3.1. Schematic view of the basic KBS architecture

It is likely that at any one time there will be more than one match between the facts in the knowledge base and conditions held in the rules. It is here that the inference engine is important. The role of the inference engine is to select the appropriate rule to be fired. Basically, it does this by first identifying a set of appropriate rules, then selecting one of them (based upon a predetermined strategy) which it subsequently executes (see Benchimol et al. 1987). Inference engine strategies are based on general methods for problem-solving, such as data-driven and goal-directed search strategies (see Rich 1983).

Aspects of the expert system user interface will be dealt with in depth later on in relation to human factors issues. However whilst considering the basic internal structure of the KBS it is also worth mentioning some aspects of the interface here. Within the context of KBS the user-system dialogue is, in effect a dialogue between the user and the

knowledge-base. Through dialogue, the user feeds in information to the knowledge base, which in turn carries out activities accordingly. These activities might involve prompting the user for more information, or in the case of an 'integrated' KBS they might involve gathering information from another source. At some point the system should reach a conclusion, such as a diagnosis of a problem, and this will be relayed to the user as appropriate.

Numerous user-system dialogue styles exist for the scenario just described. These styles are the same as any found within interactive systems generally; for example 'question-answer' (yes/no) dialogue, menu selection, form-filling, and 'pointing' through direct manipulation and a pointing input device. An important difference in the expert system dialogue compared with other interactive system dialogue concerns the requirement for explanation, as mentioned earlier. Explanation from a KBS tends to be given, or solicited, for two main purposes; firstly to explain, at the end of a dialogue sequence how a certain conclusion (such as a diagnosis) has been reached; secondly to explain during a dialogue sequence why a particular input (such as an answer to a question) is required by the system. Also there has been an increasing awareness of the potential for KBS to provide *'what-if'* explanations (Jackson and Lefrere 1984). A 'what if' explanation involves the system in finding out, on demand from the user, what 'would happen if something was the case' (e.g., the consequences of the answer 'no' instead of 'yes' being entered to a question in a dialogue sequence). The system reports back to the user the result, but does not actually make any changes to its knowledge base. In a sense, a 'what if' explanation is a prediction of the effects of a certain situation, or set of situations, being in existence.

Two common types of format for explanation exist in present KBS. The most common format is that of the *'rule trace'* , which is a listing of the rules which have been fired up to the point in the dialogue where the explanation is being given. Some amendments to the internal (program) format of the rules are made before presentation at the interface to make them more comprehensible to the user. A second format for explanation is that of *'canned text'* which are textual explanations for points in the dialogue and are presented when an explanation is required at that point in the dialogue. However, canned text explanation is not as straightforward as a rule trace, because it requires the system designer to predict several possible permutations for explanation required during a dialogue.

3.5. Stages in KBS development

Iteration is a central feature of KBS development. The high level of iteration which is found in KBS, distinguishes them to a significant extent from conventional software engineering. The essence of conventional software development approach, with its concomitant 'waterfall' life-cycle model (Boehm 1976), can be summed up as being one of "Specify-and-Verify" (Partridge 1988). This approach relies considerably on the existence of a quite rigid system specification which is produced prior to verification and implementation (that is software coding) of the system. It is intended that each development phase is carried out one after the other. Each development phase in KBS development on the other hand follows a cycle which has characterised much of AI software development (Ford 1986); this cycle has been given the acronym RUDE - Run-Understand-Debug-Edit. In KBS development the existence of a specification, in the sense of a documented design specification, does not normally exist. Instead the specification evolves through, and is contained in, prototypes of the system. The intended application domain is often broken down, or reduced, into smaller components and these components are then prototyped in an incremental fashion (Hayes-Roth et al. 1983).

Iteration is most concentrated at the detailed design stage of development. This is at the stage where knowledge acquisition and prototype testing and evaluation takes place. Detailed design follows the initial identification and definition of the problem (the application domain). This first involves assessing the suitability of the problem to a KBS approach and then the identification of the key elements of the problem. At detailed design a process of refinement takes place where the products of knowledge acquisition are used to produce a prototype which can then be tested and evaluated. Figure 3.2. (overleaf) is a general and simplified model of KBS development. In practice, the stages of knowledge acquisition and prototyping are particularly difficult to separate due to their close, iterative relationship.

The knowledge acquisition and prototyping stages will now be discussed in some depth, as it is in these two stages that the processes involved are particularly important in the context of human factors issues that are to be discussed later on.

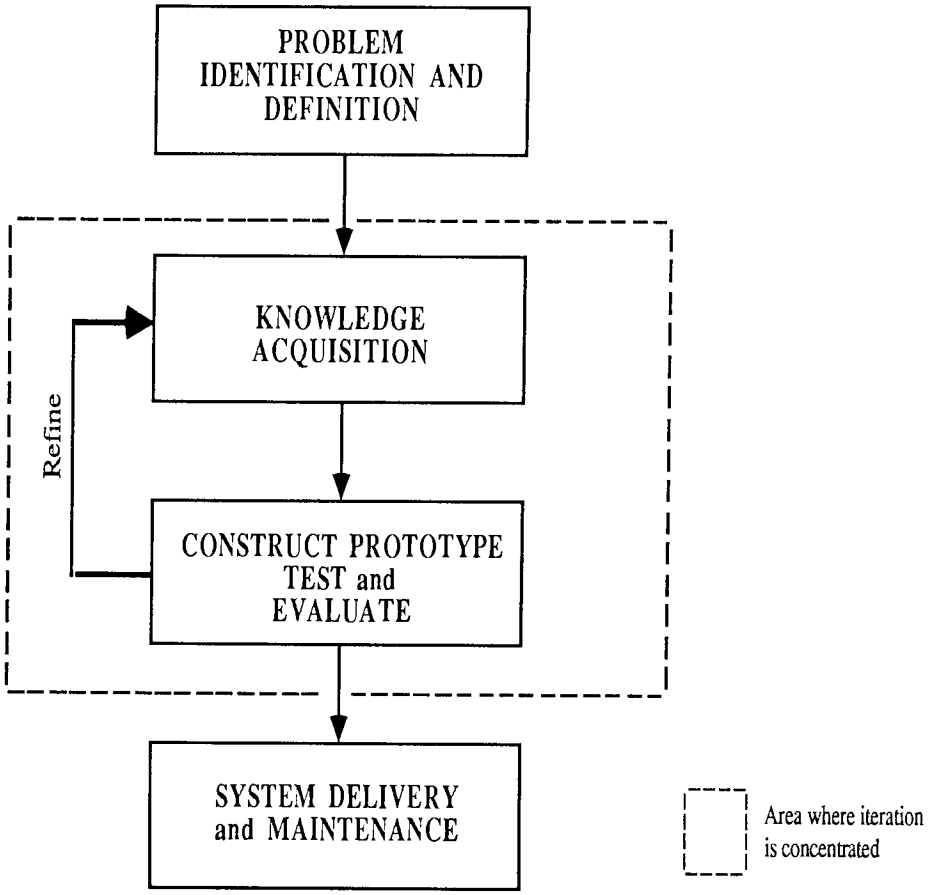


Figure 3.2. Main stages in KBS development showing iterative cycle

3.6. Knowledge Acquisition

Some debate concerning the actual boundaries of the knowledge acquisition process still exist (Hart 1986). However, avoiding such debate, knowledge acquisition is best summed up in the following extract from Buchanan et al. (1983) who state that :-

> "Knowledge acquisition is the transfer and transformation of problem-solving expertise from some knowledge source to a program. Potential sources of knowledge include human experts, textbooks, data bases, and one's own experience." (Buchanan et al. 1983; p 129)

Knowledge acquisition involves two main activities. Firstly, the activity of 'transfer' identified in the above extract is commonly known as *knowledge elicitation.* The second activity identified above, that of 'transformation', is commonly known as *knowledge representation.* These activities are initiated by the 'knowledge engineer', who has a role similar to that of the systems analyst in conventional systems development.

There are several potential sources for knowledge, as identified by Buchanan and colleagues above. The most common source used is the human expert, or group of experts. Various knowledge elicitation techniques have been developed for eliciting knowledge from human experts. It is not appropriate to go into a review of such techniques in the context of this research. However, for a review and evaluation of some of the most common techniques see Hart (1986) and Burton et al. (1987). These techniques range from basic informal interviews with experts, through to more sophisticated methods; including the use of protocol analysis of experts engaged in hypothetical case studies; and concept sorting using multi-dimensional scaling and repertory grids.

The product of knowledge elicitation is in effect a description of the problem domain in terms of the inherent objects and their relationships (for example described in terms of rules). The format of this description depends upon the elicitation techniques used. The next stage, knowledge representation, involves the codification of this domain description into a computable form - in a form which can be programmed in a particular computer programming language or implementation tool (such tools will be discussed later in the chapter). A number of key knowledge representation formalisms were mentioned earlier in this chapter (cf section 3.3.). Garg-Janardan and Salvendy (1988) describe knowledge representation thus:-

> "Knowledge representation is the symbolic and structured encoding of information and expertise (gleaned from humans and other sources) in the knowledge base which is amenable for use and manipulation by the control structure (*inference engine*) to arrive at quick and accurate solutions..." (Garg-Janardan and Salvendy 1988; p 329)

Knowledge acquisition is considered by many to be the most problematic area in KBS development, indeed it is commonly termed the 'bottleneck' (for example, Barfield 1986; Cullen and Bryman 1988). Several reasons for this exist. Buchanan et al. (1983) say that one of the major problems is the mismatch which exists between the description that is elicited from the human expert and the representation required by the implementation program. They go on to state that problems also exist due to:-

> "the human's inability to express knowledge possessed, limits on expert system technology, and the complexity of testing and refining the expert system." (Buchannan et al. 1983; p 153).

Barfield (1986) cites the problem which can occur in the selection of an appropriate expert. Here problems can arise when more than one expert is used and contradictions in knowledge between experts emerge. Garg-Janardan and Salvendy (1988) suggest that

using experts who have different levels of expertise, which is almost certainly likely to be the case, will result in different kinds of knowledge. Whilst this effect might be used to enhance a system, it might also be a contributory factor to apparent contradictions between experts in a single domain.

Knowledge acquisition is a problematic and time-consuming phase of KBS development. However, as will be seen later in the discussion of human factors issues, knowledge acquisition is not the only problematic issue which puts a constraint on producing effective, operational systems.

3.7. Prototyping and evaluation

Rapid prototyping plays an important part in KBS development. Although software prototyping is also emerging as an approach within conventional software engineering (Hekmatpour and Ince 1986; Cornes 1986, 1987), its use is more allied to the highly iterative AI RUDE paradigm rather than the conventional Specify-and-Test approach, (both of which were discussed earlier; cf. section 3.4.). One role of the prototype is as a form of system specification. However the prototype's main role is to act as a communication medium for testing (debugging) and evaluation, which is conducted with the various 'players' in the development process (primarily system developers, experts and users).

Evaluative feedback gleaned from prototyping is also crucial to the development process. Gaschnig et al. (1983) stress the importance of evaluation of a KBS before it is fully implemented. Such evaluation, they say, should be carried out at predetermined 'checkpoints' in development to ensure that the goals for the system are clearly defined and being seen to be met. In the same paper, Gaschnig and his colleagues outline the two specific functions for evaluation of KBS. They distinguish between evaluating for program *'accuracy'*, and program *'utility'*. Evaluating program accuracy involves evaluation of a prototype with the expert(s) who check the correctness of the knowledge base, and the reasoning of the system. Whilst this is fundamentally 'expert-centred', the second function of evaluation is 'user-centred', in that it involves the end-user in an assessment of *'the extent of its (the system's) capabilities, its ease of interaction, the intelligibility of its results, its efficiency and speed, and its reliability'*. As shall be discussed later in this chapter, and again later in the thesis, there is a tendency in current KBS development practice for formative evaluation to involve the expert but not so much the end-user.

Construction of rapid prototypes has been made easier with the advent of high-level AI programming languages and implementation tools (Ince 1988). These tools are particularly helpful in that they support incremental KBS development (cf section 3.4.; Hayes-Roth et al. 1983) in an evolutionary way. The next section will deal, briefly, with the kinds of software implementation tools which are used in KBS development

3.8. Software implementation tools

Various software tools exist to help developers to implement KBS. Some consideration of the tools used is important, as the nature of these tools has a considerable bearing on the way systems are developed. The tools have an internal structure that will inevitably dictate to some degree, the final appearance of the KBS, not least in terms of the way that knowledge is represented in the knowledge base and handled by the inference engine. Of equal importance are the user interface facilities provided by the various tools, facilities for graphics, implementation of dialogue styles, and explanation.

Three sets of tools are commonly used, at present, to implement KBS. Firstly there are programming languages, usually high-level AI languages such as LISP and PROLOG. Secondly there are the expert system 'shells'. These tools have been used extensively in KBS development work in Britain, in particular (d'Agapeyeff 1987). Expert system shells were developed from specific expert systems by removing the contents of the knowledge base, leaving a basic framework of an inference engine and empty knowledge base into which domain specific knowledge could be added (Barber 1984). A similar, but more extensive framework for KBS development is provided by the third class of tool, namely development environments. These are tools with more sophisticated facilities provided alongside a powerful editor. These enhanced facilities include a greater choice of knowledge representation formalisms and inference strategies, and a greater capability for user interface design. The three groups of tool are listed, along with their general attributes, in Table 3.3. The tools listed only represent three major types of tool. In reality there are many more on the market which have not been mentioned, some of which fall in between the above categories. More accurately, the table outlines three groups of tool which lie on a continuum of increasing sophistication, primarily in terms of the development facilities they provide.

3.9. Human factors in KBS development

The chapter has so far dealt with the general features of KBS design and development. Considering these aspects is an important step in moving towards an appreciation of where human factors can guide and support the development process. This section will deal more specifically with the potential human factors input in KBS development.

Table 3.3. An overview of KBS implementation tools

Tool	General features	Examples
Programming languages	- Used to build entire KBS architecture from 'scratch' or, - Used in conjunction with other KBS implementation tools, especially development environments	LISP PROLOG
Expert system 'shells'	- Provide empty knowledge base along with inference mechanism(s) - Usually PC-based - 'High-level', easier to learn implementation language - Limited set of knowledge representation formalisms - Relatively fixed 'end-user' interface due to limited user interface construction facilities - Aimed at more straightforward, tightly bounded problem domains - Cheaper purchase cost	'Xi Plus' (Expertech Ltd 1987) 'Crystal' (Intelligent Environments Ltd 1986)
Development Environments	- Provide more extensive knowledge representation, inference mechanism, and user interface construction facilities than shells - Include powerful editing facilities - Run, primarily, on 'dedicated' (e.g. LISP-based) workstations - Use involves some considerable amount of 'low-level' coding (e.g. LISP). - Aimed at more complex problem domains - Primary use as rapid prototyping tool, rather than for final delivery of system - More expensive purchase cost	KEE (Intellicorp Inc, 1987) ART (Inference Corporation 1986)

As shall be noted, such potential covers several elements of KBS development, and so in order to maintain the context of the research emphasis will be given to user interface design in particular.

At the beginning of this chapter it was noted that there is some question over the number of operational KBS (cf section 3.2.). Several reasons have been put forward to explain the lack of 'working' KBS. Many 'inhibitory' factors have been identified including logistical problems concerning the availability and willingness of experts for a KBS project (Bell 1985; Ovum Ltd 1988); technological issues concerning the basic immaturity of KBS techniques, in particular knowledge elicitation and representation (Gaschnig et al. 1983); the reluctance of management, in the commercial sector, to invest in KBS development (Morris 1986; Mumford 1988; Cullen and Bryman 1988); and philosophical issues which question the role and validity of the whole expert system approach as it stands at present (Winograd and Flores 1987). One further problem area identified, concerns the inadequate consideration of human factors in expert system design (Kidd and Cooper 1985; Berry and Broadbent 1987; Morris 1987). An extract from a paper by Berry and Broadbent (1987) highlights this problem:-

> "One of the clearest lessons learned from the early pioneering expert systems is that excellent decision making performance in itself will not guarantee user acceptability. Attention also needs to be paid to the user interface. Failure to recognise the mmi needs of expert system users is probably the biggest reason for the disparity between the numerous expert systems which have been successfully developed in a laboratory and the small number which have actually made it into everyday field use... Outside the laboratory they will only be used if people find them useful and easy to work with." (Berry and Broadbent 1987; p 18)

Berry and Broadbent's paper followed a survey of expert systems and design practice that they had conducted in the UK. Similar surveys conducted in the UK echo the main findings reported by Berry and Broadbent. In a survey of 16 British companies working in expert systems Ovum Ltd (1988) report that the most common reason given as a barrier to expert system development came under the heading of *'aspects of (end) user awareness and understanding'*. Connell (1987) in a survey of eighteen UK firms reported that :-

> "One of the most important messages that seems to be coming out of the current research and development work in a number of organisations is the need for ... some sort of 'participative' approach to system development." (Connell 1987)

In a further study, which included a survey of the penetration of KBS technology in British industry and commerce (Cullen and Bryman 1988), *'human resistance'* was cited as the major constraint in development. This heading included the issue of hostility from potential users, along with other issues such as lack of senior management support, and resistance from experts.

User interface design is most certainly an important issue in the development of operational systems in this area. Consultative KBS offer an example of a highly interactive computer system where there is a considerable amount of interaction between system and end-user, or end-users. Whilst the systems that are being built vary in the amount of user interaction they involve, on the whole it is agreed, at least in principle, that the user interface is an important element of the whole system. Gaschnig et al. (1983) state that:-

> "The key question "Will the system be used?" should motivate the (*expert*) system-building process." (Gaschnig et al.. 1983; p 278)

As was seen in the previous chapter, system utility and usability are key aims in human factors as a whole. Consideration of utility and usability factors constitute the central elements of that pervasive question that is proposed by Gaschnig and his colleagues, above.

There are some reports indicating a growing awareness of human factors issues by industrial and commercial developers who have experienced problems on specific development projects (for example, Gillett 1987; Leitch and Dulieu 1987; Kielty 1987). There are also a few case studies of expert system projects which have followed user-centred design principles. For example Mumford (1988), a major advocate of user-centred design of computer systems, reports on the participative approach taken in the design of XSEL an expert system which followed on the XCON system, mentioned earlier. This system was aimed at giving advice to DEC's sales force, and acted in this way as a front-end to XCON. Future user's of the system (sales people) were involved from the outset in the design of the system. This lead to the successful introduction of the system in the USA. It was successful, Mumford concludes, not simply because it was technically well designed, but that it was also designed in such a way to give the eventual users a sense of 'ownership', and an appreciation of the design issues involved. In another expert system project, Eason and Harker (1987) describe a similar employment of user-centred principles. This project was carried out for the Electricity Supply industry. A central feature of the design approach adopted was the involvement of users (engineers) from the outset, whose role was first to help define possible

applications and then, as development progressed, to provide feedback on user and task requirements. Eason and Harker propose benefits of this approach which are similar to those reported by Mumford. They conclude that although the approach taken may have slowed down the knowledge acquisition phase initially, benefits would accrue in the long run as user participation greatly increased the likelihood of producing a more usable and acceptable system.

The examples just mentioned point to the benefits of user centred design. However, at present they are examples of rare cases. At the same time both projects were stimulated by firm advocates of user centred principles, choice of the design approach would not, it appears, be an automatic one in most design circumstances. Active concern over user interface design comes mainly from those within the human factors community themselves, and not from those who are developing systems in industry and commerce. In the majority of KBS design the application of user centred techniques and human factors information, such as design guidelines, remains scant. Candy (1988) states that:-

> "User-centred design in which users design and experts advise is a practice which is rarely seen." (Candy 1988; p 6)

In a talk given by that same author, Candy reported a case study from a collaborative IKBS Alvey project. She declared that the early experiences from this particular project had suggested that *'..those in MMI and those in KBS...were on different planets.'* (Candy 1987). As was seen in the previous chapter there exists a mismatch between 'potential' and 'actual' human factors integration in systems design, and this also holds for KBS - for many of the reasons noted in the previous chapter.

In an attempt to address this overall problem, we should first consider what the potential human factors contributions to KBS development actually are. Kidd (1983) outlines three main areas of contribution; these are expanded upon, in Table 3.4., overleaf.

Table 3.4. Key areas of human factors in KBS
(*abridged* from Kidd 1983)

These three areas outlined by Kidd are primarily concerned with the potential research contributions of human factors in KBS development as a whole. Whilst these are all undoubtedly important areas, indeed fundamental given the immaturity of the domain, the focus and concern in this thesis is on the pragmatic integration of user interface design approaches and principles for KBS development. The question really is - what, of that which is known and espoused already about 'good' user interface design, can be carried over to KBS design and development as it is practised in the commercial setting? For this purpose, the remainder of this discussion will be confined to two central issues in user interface design; namely user requirements capture and user-centred prototyping and evaluation.

3.10. User requirements capture

On the relationship between human factors and KBS Madni (1988) states in a recent paper that :-

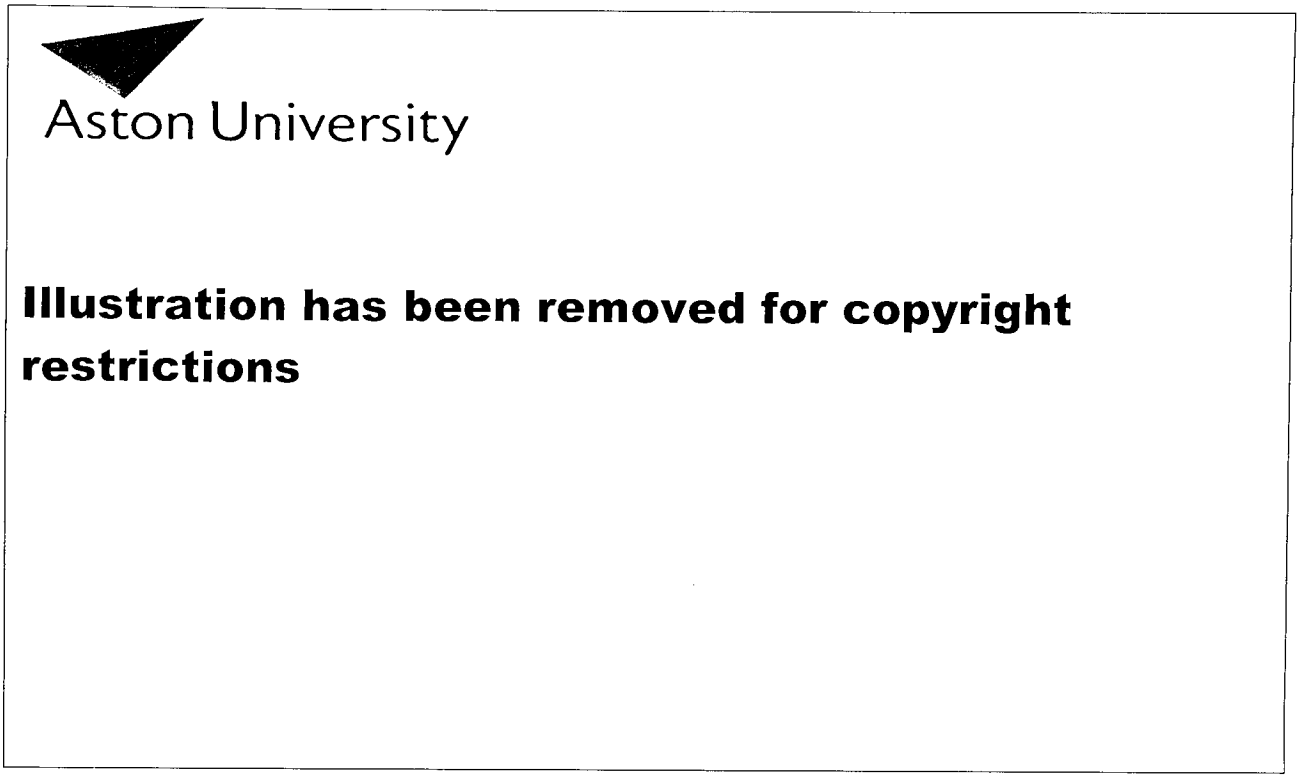
> "Perhaps the single most important human factors issue is understanding the requirements of the intended user." (Madni 1988; p 399)

Madni's comment, and indeed his whole paper, provide evidence of an awareness of the importance of user requirements capture in KBS development. It must be said, however, that there is little other evidence in the literature which confirms this view as explicitly. Often the user interface as an issue is given little more than a brief mention. The reason for this scant attention can be explained, in the main, by a preoccupation with other key issues such as knowledge acquisition. It probably also reflects a distinction between interest in what are, largely issues for human factors research associated with KBS (such as knowledge acquisition techniques), and what are issues for 'applied' human factors in KBS development.

A careful assessment and analysis of intended users of an expert system is important in two ways. Firstly, it important to design the systems to support the user in a way that is compatible with his, or her task requirements. Secondly, it is important to understand the more immediate dialogue requirements for the system (such as use of dialogue styles, and explanation facilities). These two elements relate to Kidd's terms of 'cognitive compatibility' and 'human/system communication', given previously in Table 3.4.

In the paper mentioned earlier, Madni (1988) provides a characterisation of expert system users and the associated types of expert system (such as expert consultant, or tutoring system). His taxonomy (see Figure 3.3.) outlines several variables, such as levels of domain expertise, and general familiarity with computer, which he says *'affects the design of the user interface, user-system interaction, prompts, and help facilities'*. The essence of this taxonomy is that it highlights that careful consideration of users is required in order to establish the type of system required initially, and the requirements for that system once chosen. For example, if the intended user is a novice in the application domain then he or she will require a training or tutoring system. Such systems will have different requirements from, for example, a consultation system for domain experts; particularly in terms of the type of knowledge representation, and the domain help facilities required.

Figure 3.3. **A Taxonomy of expert users**
(from Madni 1988; pp 395-414)



Some lessons can be learned about user requirements capture from empirical studies of advice-giving expert consultations in the human to human setting. These studies also emphasise the need to try to capture actual requirements, rather than requirements that are assumed by designers. The central objective of much of this work has been to first provide a classification of general features of expert advice as a basis for subsequently redefining the dialogue and inference processes which are deemed inadequate and misplaced in present KBS (Pollack et al. 1982; Coombs and Alty 1980, 1984; Kidd 1985). One of these studies conducted by Pollack et al. (1982) involved the protocol analysis of a radio 'talk show' in which listeners phoned in to a financial expert in a studio, for advice on personal finance. Overall the work of Pollack and colleagues led them to propose that the 'user'-expert dialogue was primarily a negotiation process, where elements of actual problem definition as well as problem solution were negotiated, or tailored by both participants during the interactive dialogue sequence. They suggest that such a finding undermines present expert system capabilities which are too rigid and

do not allow for this negotiation - most systems requiring that the problem is well-defined before dialogue can begin. This point is reiterated by Kidd (1985) who suggests that, in the case of diagnostic expert systems, users do not simply want a fault to be identified (something they have often already done anyway) but instead they want support in identifying and considering the implications of various alternative solutions.

The human to human expert advisory session has also been used in the development of a specific expert system, ExperTAX for tax and auditing advice (Shpilberg et al. 1986). Unlike the aim of the previous studies which tried to derive general principles from human to human interactions, the developers of ExperTAX used an expert-client simulation as part of their project development - to provide data for the design of ExperTAX. They set up a simulation where expert tax advisors were placed in the same room, but were screened visually from one another. They then analysed the dialogue which went on between the two parties, using the data that was gathered as part of the knowledge acquisition, and prototype system evaluation process.

The kinds of studies just described can tell us something about user-system dialogue requirements, and as exemplified by the ExpertTAX case, can tell us something about the task-specific knowledge requirements. As was described in the previous chapter, a source of human factors information for dialogue design lies in established guidelines (cf. Chapter 2, section 2.8.). Many of these guidelines are also applicable to KBS. Although not as well developed, or as centralised as those guidelines for general interactive system design (as for example, Smith and Mosier's guideline handbook mentioned in Chapter 2), there are also emerging specific guidelines for expert system dialogue (e.g. Kidd and Cooper 1985, Eike et al. 1986, Morris 1987). These guidelines cover KBS features, such as explanation facilities, and the use and selection of expert system shells or knowledge representation formalisms.

3.11. User-centred prototyping and evaluation

As was noted earlier in the chapter it is not easy to distinguish between the phase of knowledge acquisition and the phase of prototyping and evaluation. This difficulty in distinction also holds for the relationship between user requirements capture, and prototyping and evaluation. Formative evaluation provides a means for refining and further developing the user interface design. This overall situation stems, largely, from the nature of KBS development and user interface design, both of which are iterative processes. Both of these processes rely upon a sequence of evolution through prototyping, rather than a sequence of specification-to-implementation, as found in traditional system development.

The production and use of rapid prototypes as instruments for evaluative feedback is an important element in the iterative KBS life-cycle. During the construction of a KBS, all of the components that make up the KBS architecture (namely knowledge base, inference strategy, and user interface) need to be established and evaluated. Evaluation of system 'accuracy' and 'utility' is required (cf. section 3.7). This requires that both expert, and end-user parties are involved in the evaluation process. From a human factors perspective, the involvement of users in this process is crucial, and is an integral part of the whole emphasis upon user-centred design. Feedback from users on the prototypes are required to inform developers of potential system utility problems which exist at the task level, and at the dialogue level.

Although user interface evaluation is important in principle, it appears that it is not carried out to such a great extent in practice. In keeping with the general lack of user centred design, the user interface does not appear to figure greatly in the overall evaluation process as carried out by commercial developers. To paraphrase the findings from Berry and Broadbent's survey (1987), they found that evaluation tended to be centred upon *'the quality of a system's advice and correctness of the reasoning techniques used'*, rather than *'the quality of the human-computer interaction, system efficiency, or cost effectiveness'*. From a human factors point of view the scarcity of end-user evaluation would be of some considerable concern. It can be argued that the expert evaluation, whilst obviously an integral and necessary feature of the KBS development process, is biased and misplaced when equal consideration to end-user evaluation is not given. Problems in meeting user requirements are particularly likely when the end-user of the system is not of the same level of knowledge and expertise as the expert(s) around whom the system's knowledge base has been formed. There is a potential mismatch between the nature of evaluative feedback which the expert will give compared with a novice's feedback. As Rector et al. (1985) point out :-

> "There is ample evidence to support the intuitive notion that experts behave differently from novices. Experts do not just know more, they know differently." (p 240)

It can be argued that this expert-novice difference is likely to have an effect upon the sensitivity to certain elements of a system during the process of evaluation. In an expert's evaluation, inadequacies in the user interface may be overlooked due to the evaluators familiarity and knowledge of a domain. Possible inadequacies, such as poor explanation or help facilities, might only come to light when the system is given to end-users who do not have the same level of familiarity and knowledge.

3.12. Concluding remarks

This chapter has shown that there is considerable scope for human factors integration in KBS development. As appropriate to the research work in this thesis, the focus for human factors integration has been necessarily confined to user interface design issues. As has already been implied, in reality human factors issues pervade the whole area of KBS construction and use. However, even within the activity of user interface design there is considerable potential for human factors input, as has been highlighted in this chapter. Human factors can provide support for the overall approach to KBS development, ensuring that the end-users are taken account of at each stage. Support can also be given to discrete design activities, for example by providing guidelines for detailed dialogue design.

As had also been identified in the previous chapter, once again it appears that there exists a gap between the *potential* and the *actual* integration of human factors principles. By highlighting this gap these two chapters have provided the backcloth to the research work which will now be reported. The following chapters will report on the specific experiences which arose from the research; experiences which were gained when addressing the problem of human factors application on 'live' KBS development projects.

The next chapter describes a preliminary case study of a project carried out as part of *Intellipse*. This was a study which gave the basis for the future case study work reported in subsequent chapters.

Chapter Four

The '*Advisor*' prototype: A preliminary case study

Outline

On joining the Intellipse project, the author became involved in work on Advisor, a system which was aimed at giving advice to designers on a phase in DP system development. Advisor was at an early prototype stage when the author joined the project. Following a description of the background to Advisor, this chapter describes his role within its subsequent development. The work focussed on user issues and initially involved an appraisal of the existing user interface to the prototype. This appraisal highlighted some potentially fundamental problems with the existing system; problems which stemmed from an inadequate assessment of user requirements at a higher level than just the 'physical' user interface (screen layout, dialogue style etc). Some possible factors which had contributed to this situation are discussed, along with a description of my attempts at remedying this situation; attempts which involved a closer examination of potential users and their requirements of an advisory system like Advisor. The chapter concludes by reporting on the outcome of this analysis and also on the eventual outcome for the Advisor system in general. Involvement in work on Advisor enabled the author to monitor, closely, a KBS development project and to experience how issues related to human factors were dealt with in this process. This assessment helped in the formulation of the succeeding research plan that was then adopted, and which will be described in future chapters.

4.1 Background to 'Advisor'

In the initial project specification for *Intellipse*, a basic, high-level, architecture had been described (see figure 4.1). This architecture consisted of two modules; these in fact represented two stages in the proposed plan for development. The first of the modules - *Advisor* - involved the development of an advisory KBS to provide advice on activities and objects which made up the domain of DP system development. The second module, *Designer,* would contain a set of KBS to actively support and carry out various tasks in the system design process. A significant part of the research within *Intellipse* would involve the identification of appropriate tasks in DP design which could suitably be supported by KBS. The domain knowledge which would be utilised in *Intellipse* was based extensively upon the structured methods for commercial DP which BIS themselves have developed, and which they provide as part of their generic Structured Development Standards (generically entitled MODUS). Given the enormity of the whole DP domain, it was only realistic given the bounds of the project, to focus on one stage of the development life-cycle. The stage that was chosen was the design phase, which under MODUS is called "Structured Systems Design" (SSD). SSD follows the systems analysis stage and is occurs prior to system implementation.

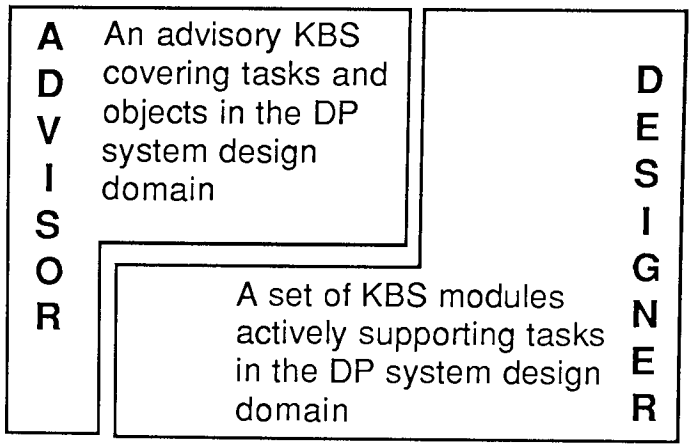


Figure 4.1. The basic architecture for *Intellipse*

A long-term objective was to integrate the *Advisor* module with the *Designer* module. In this way *Advisor* would act as an underlying reference point which described the basic activities and objects in the domain. These were described through 'passive' text and diagrams. *Designer*, on the other hand, would act as an active support tool, carrying out some of the activities which make up DP system design.

Work on *Advisor* constituted the first stage in the development of *Intellipse*. By the time the author was recruited to join the project, an architecture for *Advisor* had been drawn-up and the two major components of the system had been developed to an initial prototype stage. These components were the *'Knowledge Base Manipulator'* (KBM) - for entering the domain information concerning objects and activities into the knowledge base; and the Advisor front-end which allowed the user to access the knowledge base (see figure 4.2.) Up to this point, the prototypes had not been demonstrated to anyone outside the immediate project team. The prototypes had been scratch-built, using a PC-based dialect of the *Prolog* programming language, *micro-Prolog*. Development was being carried out on an IBM PC/AT with the intention that the final deliverable system would also run on a micro-computer.

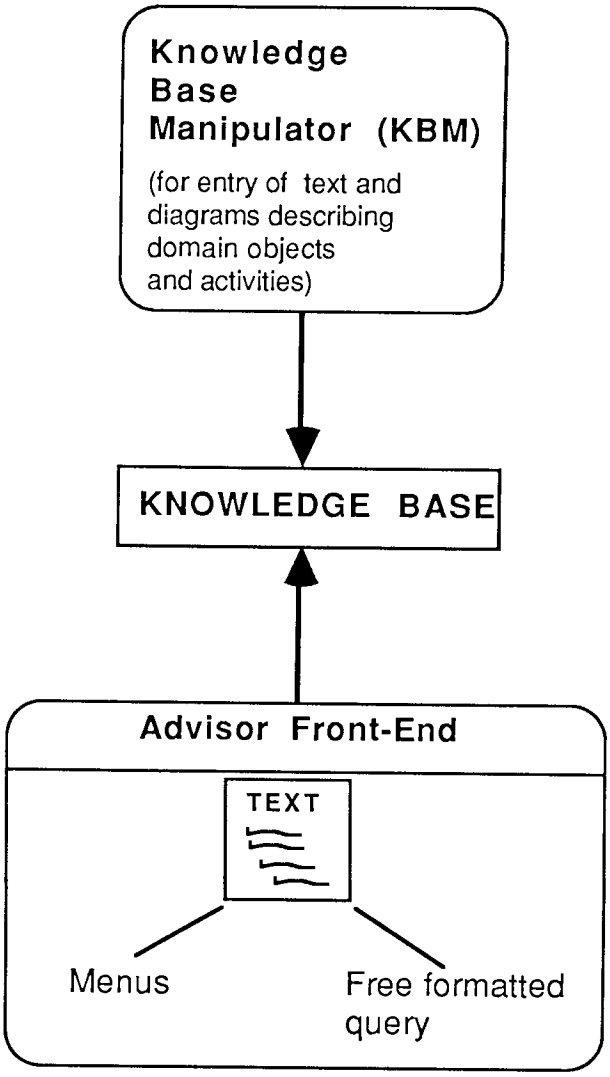


Figure 4.2. ***Advisor*** **architecture**
(including the front-end dialogue modes)

4.2. Project organisation

Early work on *Advisor* was primarily carried out by members of the collaboration at Aston University and at BIS; both groups being based in Birmingham. The third collaborator, BSC, had the role of system evaluators. Such evaluation, however, rarely took place in the early stages. Overall project management and control came from a Principal Consultant at BIS. Three other consultants made up the rest of the team from BIS; two in the role of knowledge engineers and one as chief programmer. In the early stages knowledge engineering was conducted through the use of existing paper-based manuals and it was not until a later stage that evaluation of this knowledge with BIS experts in the domain of SSD was carried out. At Aston, a fellow research officer had played a major role in designing the architecture for the system and was subsequently

involved in the knowledge engineering and feasibility study for *Designer*. The author joined the project with responsibilities for the user interface.

Series of 'localised' design meetings were held between the team members at Aston and BIS. These were supplemented by full project meetings, which were held on average every six weeks, and involved members from all of the collaborating groups. A monitoring officer from the Alvey Directorate also attended these full project meetings.

4.3. The Advisor interface

In the project proposal for *Intellipse* it had been proposed that one role for the Aston partner would be to look at the development of a 'natural language front-end' (NLFE) for *Advisor* and *Designer*. On joining the project one of the author's initial responsibilities was to look at this area. What was also requested was a general appraisal of the existing front-end to *Advisor*, for the purposes of outlining recommendations for changes and improvements to the user interface.

Dialogue issues

On analysis of the project situation which existed it was unclear how natural language would be a viable and appropriate mode of dialogue within *Intellipse*. This was particularly the case for the prototype *Advisor* where a fixed dialogue centred around the user requesting text on a particular topic within the SSD domain. The system was structured in such a way that types of query had been distinguished. This meant that the user could ask for 'What', 'Why', 'How', and 'When' information about a particular topic. The system would respond with a screen of text , retrieved from a text file in the knowledge base, that corresponded to the topic and the chosen query type. For example, if the request had been for "When" and "Data analysis" the system would relay information about when the activity of data analysis takes place. It is worth noting here that the reason that it had been decided to structure queries in this way was related to an approach used by BIS to teach SSD on formal courses which it provided to clients.

Two dialogue modes had been specified to enable the user to request information on a topic:

- *Menus*

  The default dialogue mode was a menu dialogue where the user selected (via the keyboard - the sole input device being used at this time) the query type from a menu followed by the topic from a hierarchy of domain options. This second set of menus represented the SSD domain hierarchy, and allowed the user to explore the domain. This menu dialogue mode had been identified

early on in the project as being appropriate for those users who were unfamiliar with the hierarchical structure of SSD and wanted to navigate and explore the domain. It was also seen as important if the user was unsure of the term used for a particular topic. A problem identified early on was that different structured methodologies use slightly different terminology for what are essentially the synonymous items; this could be problematic if a user was new to SSD but had experience of other methods.

- *Free-formatted query*

  The alternative method of accessing text on a topic was via a 'free-formatted query'. In relation to *Advisor*, this mode was seen as constituting the natural language front-end. By choosing this mode the user could type in a query in a relatively unrestricted form, for example:-

  Example 1.

  > What is data analysis

  Example 2.

  > When do I do third normal form

  Via a simple parser, written in micro-Prolog, the system would extract the query type and the topic name; the remainder of the input string was redundant. The rationale behind this dialogue mode was that it would enable a more experienced (in SSD) user to go straight to a particular topic, thereby short-cutting the menu hierarchy. It was also intended to allow someone to access a topic even if they did not know where in the hierarchy that topic resided.

It is important to point out that the assumptions concerning the user's dialogue requirements had been made by the project team themselves, and had not at this point been evaluated with potential users.

4.4. Report on the use of natural language in *Intellipse*

Following an appraisal of the overall project situation, along with a more detailed look at the *Advisor* front-end, the author produced a report outlining the role of natural language (NL) in the project (see Appendix 1). The central tenet of this report was that :-

> "...we ought not simply think in terms of natural language but also consider the alternative methods *(dialogue styles)* available." (Appendix 1)

An aim of this report was to show that in *Advisor*, a NLFE was largely inappropriate given the underlying structure of the system; a structure which consisted essentially of static text. Whilst some alternative form of dialogue style was almost certainly required to complement the menu hierarchy, the existing free-formatted query was not meeting this requirement. Entering queries, such as those examples above, required several keystrokes. Above all, the real parsing ability of the system, which only recognised the query type and a topic, was potentially misleading. This could give a false impression to the user concerning the robustness, and underlying structure of the system (perhaps leading to the formulation of an incorrect *conceptual model* of the system). This situation can be related to one of the often cited disadvantages of computer NL interaction - that of *'anthropomorphism'* (Shneiderman 1986). Here the use of NL can lead the user to overestimate a system's 'intelligence', resulting, on interaction, in a mismatch between the user's expectations and the system's actual behaviour and performance. In the context of *Advisor* the author suggested that an alternative to the 'free-formatted query' would be via 'form-filling', whereby the user would be prompted to enter just the query name and the topic name. This method would prompt the user for only that information that was required by the system in order to provide the appropriate text.

Utilising the existing literature on the subject of NL interaction, the report laid down the case both *for* and *against* its use relating existing general knowledge of this area to specific issues in the project. In the case of *Designer,* the author reported that because at this stage the system was nothing more than a concept it was too early to say what form of dialogue would be required for the various ingredient KBS. Instead the team ought to be aware of the various dialogue styles that might be available to us in design, and to evaluate the suitability of each of them accordingly. Given the significant problems that were likely to be encountered in developing a NLFE, problems of both a theoretical and logistical nature, it would have been inadvisable to become fixed on the idea of NL interaction. For example logistically problems were likely to occur in terms of computer memory overheads, particularly as micro-computers were being used for development and it was envisaged that they would be used for final system delivery. All in all, the NL mode of interaction would provide no foreseeable advantages and, indeed would probably be disadvantageous in certain respects.

4.5. Report on the Advisor front-end

Along with a report on the development of a NLFE, the author was also requested to carry out an appraisal of the overall user interface of the existing front-end of *Advisor.* This appraisal resulted in the production of a further report, which outlined existing inadequacies in the interface, along with recommendations for possible improvements

(see Appendix 2). Where possible, elements of the interface that were potentially problematic for users were identified. Once again, as far as was possible, this assessment was based upon design guidelines from the literature. For example, one problem that was identified concerned the display of certain options in a menu which were unavailable at some levels. This led the user, on selecting one of these options, to a 'dead-end'. This goes against existing guidelines on menu dialogue such as :-

> *"Design a menu to display only those options that are actually available in the current context for a particular user."* (Smith and Mosier 1986; section 3.3, p 238)

On reflection, however, the majority of those problems identified were specific to the application and consequently could not be mapped to guidelines in the literature. Features existed which were, at the very least, potentially irritating to users. For example, there was an the absence of page-scrolling; also the user was forced to view text on a topic by individual query-type rather than being able to view all of the text pertaining to a topic. These kinds of features were, however, specific to the application itself. On a general note, where guidance in the literature does not exist, features such as those mentioned can only be dealt with through system evaluation. This also holds for occasions where contradictions in the guidelines arise, and where some trade-off is required. The usefulness of raising potential interface inadequacies and issues in the manner described can only be realised if such an exercise is a precursor to subsequent evaluation with users. This was shown, quite emphatically, to be the case with *Advisor* and it is an issue which will be dealt with in more detail later in the chapter when the final outcome of the *Advisor* system is reported.

The report also proposed the use of a mouse input device, but included the caution that a decision should be made on this quickly as the use of a mouse was likely to change the layout of the interface. Initial reaction to this suggestion was negative, as it was emphasised that the target environment for the system was that of DP departments, where, it was suggested, mice were rarely found alongside the commonly used IBM, or compatible range of micro-computers. This was mid-1986, and prior to the release of IBM's PS/2 range; a range which are more commonly accompanied with mice. Some six months after the report, it was decided to develop *Advisor* with an interface which could be used with a mouse or via the keyboard.

The above 'mouse situation' illustrates the nature of some of the concerns, and the constraints which existed in the project as a whole. Such constraints, probably not

untypical in commercial development environments, have an obvious effect on the way that one can apply 'ideal' human factors guidelines and principles, as they did in the work on *Advisor*. As a result, it was situations like this that were important in formulating the research which followed.

4.6. Not *"Knowing thy user"*

The above report outlined recommendations for improving the 'physical' interface of *Advisor*. Improvements could be made to the menu content and hierarchy; and to other minor features of the screen layouts and navigation facilities. Whilst such changes would be important there remained one significant problem with the *Advisor* prototype. This problem concerned the fact that there was no clear definition of the potential end-user of the system. Making 'physical' improvements, like those mentioned above would not, and as it transpired did not, compensate for an inadequate understanding of who the end-users were and, hence, what their requirements of *Advisor* would be.

On becoming familiar with the project situation the author felt that the primary objective should be to establish who the users were. This, the author felt should take precedence over any further work on improving the existing physical interface. The first step in achieving this objective was to discuss with the project team, and in particular the project manager, who they thought the users were, and what was the role of *Advisor*. These discussions lead to the following, high-level, picture of its users :-

- Users were likely to be "intermediate-level" DP designers, that is DP designers who had a certain amount of training from taught courses.
- This class of designers were assumed to be familiar with terminology in the domain but probably had little practical experience in applying their knowledge. They may have had experience of other structured development methods in DP.
- It was stated that *Advisor* was not seen as a formal teaching aid, that is it would not act as a tool for computer-assisted learning.

Given the amount of work that had already been carried out on *Advisor*, such a description of the intended user environment was still essentially vague. The descriptions were at a superficial level and several assumptions had been made about the user and their requirements. For example, there had been no assessment of how much knowledge an "intermediate-level" designer would possess. Neither had there been an assessment concerning the likely form the questions from the users of *Advisor* would take; for example, were they likely to ask very general questions like "What is data analysis?" (a typical form of question assumed for *Advisor*) or were they more likely to ask questions which were very specific to an immediate design problem? Analysis of

these types of issues had not, on the whole, taken place. Early evaluation with appropriate people external to the project, had not been carried out.

With the benefit of hindsight it was obviously quite easy to point out inadequacies with the way the *Advisor* system had evolved. As the author had joined the project after it had already started he was probably at an advantage in being able to see such inadequacies, in a way he was acting as an 'external' source of evaluation (albeit not the most suitable one, as he was not a potential end-user of the system). What was important though, was that any lessons that could be learned from assessing how certain problems had developed, were learned. An assessment of these problems was particularly instructive for the author's own research as it gave indication of where support (in terms of user issues) could be given to the project team. Whilst it is not possible to point, with any great certainty, at the factors that had been instrumental in creating the situation described, it is possible to make some hypotheses.

4.7. Factors influencing the development of Advisor

Possibly the most significant factor effecting *Advisor*'s development concerned its perceived relationship with *Designer*, and hence its overall role within *Intellipse*. Advisor had not initially been seen as a stand-alone KBS, but as a module which served *Designer* - the key module which would actively carry out some design tasks. If *Advisor* could be provided as a separate stand-alone system then that would be seen as a bonus (especially commercially); this was not the prime objective though. In this way *Advisor*'s role might be seen as one of an on-line help system, providing reference on activities and objects in the design domain. It had been decided to first build *Advisor* as this module was seen to be the easiest to do, and so it was more likely to produce an identifiable system. In comparison, *Designer*, in the bounds of this project, was more likely to deliver conceptual rather than concrete systems. Separating the development of the two modules in this way, however, meant that *Advisor* was, to a large extent, being developed as a stand-alone system. This was mainly because *Designer* was at such an early conceptual level. The design activities which it was to actively support (i.e. the activities appropriate to KBS support) were yet to be decided, pending extensive interviews with domain experts. It was intended to integrate *Advisor* with *Designer*, but it was not clear at this stage what *Designer* would 'look like', therefore assumptions were being made about the requirements of *Advisor*, but these assumptions were not, as yet, being tested.

A further factor in *Advisor*'s development concerned the lack of evaluation with users. Unlike the factor identified above, which was probably very specific to *Intellipse*, lack

of user-centred evaluation is a problem which can be identified elsewhere in KBS development projects. It was identified both in the literature on KBS development (cf. Chapter 3; section 3.11) and it was a problem that was apparent in some of the subsequent interviews and case studies conducted as part of this research project. In respect of *Advisor*, lack of evaluation had led to many of the problems which have been outlined, namely those concerning the lack of analysis of potential end-users and their requirements. BSC, the third collaborator in the project, had the proposed role of providing end-users to act as evaluators throughout *Intellipse*. They represented a large DP installation, and so, overall, were an ideal primary source for evaluation. However, BSC are not users of SSD or any such generic structured method. This meant that they could not provide potential users who had an appropriate knowledge and background to be able to give detailed evaluative feedback concerning user requirements for *Advisor*. Instead the feedback which was gained from BSC, and from other sources (through various informal demonstrations of the system to other interested groups), tended to centre on general comments about the physical interface. These comments tended to be of a similar nature to those described in the author's initial report and mentioned in section 4.5. of this chapter. Feedback was given on such elements as screen layout, and scrolling facilities but comments could not be made on whether the textual explanations were adequate in content, or indeed whether such text was appropriate support at all.

A third important factor, which possibly had some effect on the approach to *Advisor*'s development, concerned the technical background and experience many of the project team, particularly those responsible for project control. *Intellipse* was intended primarily as a research and development (R&D) project, quite unlike typical commercial DP projects, where commonly impetus for a development project is initiated by a client, and where the phase of system analysis involves considerable reference to a user base which is more clearly defined. Although not all of those involved in *Intellipse* were from a DP background, indeed there was the academic involvement, the project management came primarily from BIS where the ethos is much more orientated to commercial system development. This led to the project assuming an awareness and consideration of some of the constraints that are more commonly associated with commercial development, rather than 'pure' research. These constraints were manifested in the choice of implementation tools, and the limited access to experts and end-users. What arose in *Advisor* was a situation of *'a system in search of a user'*, and whilst this situation might have fitted in with a wholly research setting, it proved more incongruous, and problematic in the *Intellipse* development. The importance of having "clients" in an expert system project has been raised by others. Trimble (1988) stated that the scarcity of experts systems in everyday use might partly be attributed to a lack of 'client-driven'

projects. The existence of clients are important in two respects; firstly they act as a source for validation and evaluation; and secondly they exercise financial and project control. Trimble said that if such control is not exercised, as had often been the case in expert system research, then the likelihood of developing an operational system is significantly diminished. In the absence of real clients, then he advocated the use of *'pseudo-clients'* - that is a group of person, or group, who administer an appropriate form of external control on a project.

The third factor identified was an interesting one as it highlighted another key issue to be considered in the future research plan. It emphasised the need to have some awareness of the environment in which design is going on, in this case the KBS design environment, and the background (in terms of past experience, views, and motives) of those involved in system development. These factors will have some effect upon the up-take of any information, methods, or tools, which are aimed at supporting those involved in the development process. The context of this particular work being the up-take of support for user interface design and user consideration in the design of KBS.

4.8. Attempts at defining Advisor's users

In an attempt to remedy some of the problems which existed with *Advisor* the author proposed that he should carry out an analysis of how advice was given to existing BIS clients on areas within MODUS. This analysis was to centre on the human-human advice giving sessions which took place between a BIS consultant and clients who were just beginning to use some of BIS's structured methods.

The author believed that such an analysis would be useful in getting some better direction on the requirements of *Advisor*. The author was, however, wary of developing an enhanced requirement specification for *Advisor* that tried to match directly onto an extant human-human scenario. Analyses of human-human advice giving interactions have proved interesting and useful, at a general level, in the past, for example the Pollack et al. (1982) study of the interactions between a human expert and 'users' on a radio phone-in programme (cf. Chapter 3; section 3.10.). They had viewed the advisory scenario as a 'negotiation process' which was comprised of four main elements - 'motivations', 'goals', 'strategies' and 'moves'. In another study (Coombs and Alty 1980) had identified three phases in advisory encounters. These phases were definition of the query (problem); formulation of the solution; and communication of the solution.

In the context of work on *Advisor* the author believed that a detailed analysis of the kind carried out in the above studies was not appropriate. The main reason for this was that the project had progressed too far for the team to consider any major change of direction that might be suggested by the findings of an in-depth analysis. Such a change of direction would not have been deemed acceptable, and so the intention behind the proposed analysis was to at least demonstrate another perspective on *Advisor* and its requirements. To this end the assessment that was carried out, concentrated on the high-level goals which clients have when interacting with the BIS consultant. The decision to avoid a detailed analysis was further vindicated when a clearer picture of the form of consultant-client advice sessions emerged (this will be explained in more depth below).

Two main activities constituted the above analysis :

- A detailed structured interview with a BIS consultant.
  This consultant was external to the project, and was involved in both giving courses in BIS structured methods and, importantly, giving follow-up advice to clients when they had begun to practice the methods.

- Attendance at a BIS course
  The author attended a week's course given by BIS to staff from outside companies. This course was on Structured Systems Analysis (SSA), another phase covered under MODUS. The aim was to get an idea of the information provided by these courses and would thereby provide some guidance concerning the level of knowledge someone might have following a course. Attending the course also enabled the author to talk, informally to both the course attendees, and the course tutors. This helped in developing a picture of the types of work environment that the attendees return to after the course (i.e. did they come from companies already using MODUS, or was this a complete introduction for both them and consequently their company?). Talking with the tutors helped ascertain, at a high-level, what kinds of problems were commonly experienced, and generally what help was required from them.

The first of these activities resulted in a report being written and distributed to the project team (see Appendix 3). The report described some of the characteristics of the environments in which advice takes place; this included information on the way that advice was given to clients, and the nature of that advice. It also included some general issues which had arisen from the interview, concentrating upon the consultant's own view of Advisor and the role that such a system should adopt. What also arose from the

interview was that it would be logistically difficult to continue a deeper analysis of advisory sessions (involving attending, and recording protocols from, advisory sessions) as had once been discussed. Problems in relation to the frequency and *ad hoc* advanced planning of advice meetings meant that such a detailed analysis would prove difficult and would not fit in with the overall project plan and time-scales for stages proposed for Intellipse. This problem of logistics provided further affirmation of the decision to conduct the analysis at a high-level only.

Some general features arose from the overall analysis; features which were likely to have some bearing upon the role of *Advisor*. The important features were:

- A high degree of tailoring of the methods, and standards, takes place in order to suit an organisation's present (often well-established practices). This meant that a system to support one organisation in applying a set of 'tailored' methods would have, itself, to be adaptable and flexible.

- There did not appear to be much, if any, commonality in problems experienced by clients when first using the methods. This probably links with the first point, as it might be explained as a function of the high level of variance between the needs of each organisation. It was reported that problems experienced were largely very specific to an organisation's design problem, and to the way it was using the methods. Again this pointed towards the need for a support system which could encapsulate organisation-specific information. The alternative would be a very general, possibly irrelevant, system.

- It was reported that the consultant's role was largely one of providing reassurance, rather than one of a formal teaching role. This reinforced the statement from the view expressed earlier that *Advisor* was not a computer based learning system. At the same time however, the consequences of this feature in determining a suitable role for *Advisor* remained unclear.

4.9. Outcome

The findings from the analysis were useful in demonstrating several important issues to be considered in the development of *Advisor*. Above all, the findings helped highlight the importance of obtaining a clear understanding of the potential end-users and their environment. Several issues had emerged from the analysis which had not been

considered previously, and these issues might have proved fundamental in finding an appropriate role for an advisory system in this domain.

Unfortunately, but again instructively, the findings had little impact on the future development of *Advisor*. Discussion and work on *Advisor* tended to concentrate, repeatedly, on changes to the physical interface, despite further calls from the author to analyse requirements on a wider level. This reluctance to change can almost certainly be attributed to the considerable effort already invested in the *Advisor* prototype. This was possibly coupled with the belief that either *Advisor* , with just some amendments, might still be integrated with *Designer* in some way. Alternatively, it was considered that the 'Knowledge Base Manipulator' which allowed information to be added to the knowledge base, might serve a useful as a generalised shell in which domain specific information could be entered. This alternative would itself have taken considerable reworking of parts of the system. It would, in any case, have diverted significantly from the initial objectives set out for *Advisor*. Neither of these subsequent aims for *Advisor* have been realised. The path taken in developing *Designer*, (to be discussed in chapter 8 as part of a further case study) was such that the role of *Advisor*, as an underlying module, became less and less likely.

4.10. Concluding remarks

Late involvement in the development of *Advisor* inevitably made difficult the task of both conveying, and having accepted, the important user issues. However, this situation proved extremely useful in providing insight into some of the key problems which exist when attempting to introduce human factors related information and approaches into design environments such as that of *Intellipse*.

An important factor identified from the study was that support for user interface design was not just important at the physical level (i.e. screen layout, dialogue style, etc) but that support was required in capturing user requirements for the system. Support was required at the analysis stage, before the implementation of the physical interface. Supporting physical interface design, alone, does not remove any incorrect assumptions concerning fundamental user requirements which might be made at an earlier stage in design. This had been shown to be the case with *Advisor*, where simply making improvements to the user interface which existed on the prototype, would not have overcome fundamental problems which had arisen due to an inadequate assessment of user's needs from such a system.

This one study could not be construed as totally representative of commercial KBS development, but it did raise some general issues which needed to be considered in later work. Taking this into account the next stage of the research was to develop a plan for the future work which would build on the experience gained from the *Advisor* study. An aim of the plan was to taker a wider view of present commercial and industrial KBS development. The subsequent plan of research will be described in the following chapter.

Chapter Five

The Interviews: Stage one of the subsequent research

Outline

This chapter outlines the aims of the research which followed the Advisor study. Stage one of this research involved a series of interviews with commercial and industrial developers of KBS. The interviews helped to develop a clearer picture of KBS development practice, and provided the basis for future contact with organisations carrying out KBS work. These contacts would serve as sources, alongside the Intellipse project, for feedback on the future work as it progressed. Various features emerged from the interviews. These features are discussed in relation to their potential effect upon the priority given to the user interface design activity within overall development.

5.1. Aims of the subsequent research strategy

Two sources had been tapped in order to develop a picture of the problem domain. Firstly, the literature on human factors integration into interactive systems, and KBS, development had provided a general insight into some of the problematic issues. This source had produced a high-level definition of the problem domain - human factors integration in interactive system development. Secondly, the work on *Advisor* had provided a much more direct source for insight, in the context of KBS development - the specific area of focus. It was important to define the problem area at a more specific level; at a level where some of the problem issues might be tackled, feasibly, by the subsequent research work. The *Advisor* study played an important part in this process. However, in order to get a more representative picture of present KBS development practice it was decided to broaden the contact with commercial and industrial KBS work that was going on. This would enhance the knowledge on existing design practice and would indicate whether problems similar to those identified in *Advisor*, could be identified elsewhere, or whether *Advisor* had offered only an isolated case.

Two stages were outlined for the subsequent research:-

1. To consolidate findings from *Advisor* study and from the literature.
   Further KBS work within *Intellipse* (as part of the *Designer* phase) would in part, act in this role. Further to this, it was decided to contact other commercial and industrial developers of KBS, so avoiding a narrow view of present development practice in these environments. Such contact would be a precursor to the following stage, described below.

2. To develop an approach, or set of approaches, to tackle some of the key problems that had been identified as a result of the first stage.
   The research objective of these approaches, or methods, was two-fold:

   - They would provide a 'way-in' for active involvement on live development projects (outside *Intellipse*). To ensure a level of active involvement in another project, or other projects, it was necessary to provide some identifiable support. Simply monitoring a project (i.e. passive involvement), whilst a useful exercise, would not it was felt, be as instructive as actually trying to influence the development process in some way. The methods that were developed, were a means to this end.

   - They would act as a stimulus for feedback from those (development project teams) involved. This feedback would serve two purposes :-
     - Specifically, the feedback would be used to evaluate, and indicate necessary improvements in, the approaches themselves.
     - The response to the approaches, in terms of their deemed usefulness and appropriateness to KBS development, would itself be illuminating, and help to further develop the overall picture of human factors integration in KBS. Assessing both actual, and potential, integration.

The first stage of the research, which involved broadening the contact with KBS developers, will be described in this chapter. The second stage, concerning the development of the methods, will be described in the following chapter.

5.2. Interviews with KBS developers

The aim of the first stage of the research, following the *Advisor* study, was to set up contacts with other developers of KBS in either the commercial or industrial field. These would help provide a better picture of existing design practice, and the approach being taken to human factors within that practice (Bright and Stammers 1988). It was not the intention to contact organisations involved in purely research oriented, speculative KBS development.

Obviously, for the research to be feasible in trying to tackle some problem areas, it was necessary to break down the identified 'problem space' into areas that could feasibly be addressed within the bounds of this project. To this end it was important to ensure that the content of information gathered from the contacts was quite detailed. The *Advisor* study had provided some preliminary detailed findings, it was now important to consolidate these findings through further involvement in KBS work. Various methods were considered for conducting this process of consolidation. Initially, the use of a postal questionnaire was considered as a means of contacting, and eliciting information from, those involved in KBS. Indeed, a pilot questionnaire was circulated internally to staff involved in KBS projects at Aston University. However, after further thought, and a poor response from the questionnaire at Aston, it was decided that a questionnaire might not be the most effective method; especially given the objectives of this stage of the research. Responses to postal questionnaires, which would rely upon recipients returning the completed form, were likely to be met with a poor response rate, especially if the would-be respondents cannot see any immediate value for themselves in the exercise. This limitation was likely to be compounded by the considerable amount of questionnaires and mailshots that had already been circulated as a part of the Alvey programme. It was important to avoid a dismissive response to 'yet another questionnaire'. Instead, what was required was an initiation of contact with other KBS developers, which could then subsequently be followed up. Rather than aim for a large population through questionnaires - which would have provided a wide but fairly shallow analysis of design practice - it was decided to aim for a smaller sample size but conduct a more detailed analysis of each resulting contact.

To this end it was decided to employ the following procedures :

- Identify potential individuals and their organisations, involved in KBS. Names of likely people were obtained from lists of delegates who had attended conferences or seminars on KBS and related subjects.
- Send letters to the above, describing the research project, and requesting a meeting to discuss their work in KBS. In appreciation of the fact that those contacted were likely be involved in work of a commercially sensitive nature, confidentiality of issues discussed in meetings was stressed.
- Follow up the letters with a phone-call to discuss the possibility of a meeting, and to arrange a suitable time if confirmation was obtained.

In all fifteen people were contacted, from which eight interviews were arranged. Interviews were then carried out with people from the following organisations :-

BIS Applied Systems†
BIS Banking Division
British Coal
British Gas
British Nuclear Fuels
Rediffusion Simulation
Rolls Royce Associates
Trustee Savings Bank

†Involving a team involved on KBS work unrelated to the *Intellipse* project

Initial interviews took place over a two-month period, between January and March 1988. As shall be mentioned later, some further interviews and contact were established with some of the above organisations.

Those contacted who did not consent to meetings gave various reasons for why it was not possible. Some stressed the problem of confidentiality, and gaining clearance from management for a meeting. This was understandable, although, as mentioned above, maintenance of confidentiality had been stressed in the original letters. Others said that they did not think that their work on KBS had reached a sufficient level where they could discuss user interface and human factors related issues. This was either because they viewed themselves as being preoccupied with other aspects of development (e.g., building and programming the knowledge base), or that they saw user interface design as being heavily constrained by the tools (primarily, expert system shells) that they were using.

Those meetings that were arranged were all held at the organisations themselves. On average the interview took place over half a day, although some continued throughout a whole day. Most of the interviews involved a discussion with one central person who had been contacted, along with his or her colleagues. Invariably a demonstration of some relevant software, usually prototype KBS under development, also took place. The interviews were semi-structured, each based upon a set of prepared questions from which discussion developed. Feedback on the style of the pilot questionnaire (see Appendix 5), distributed internally at Aston, had been used in formulating the questions.

Issues concerning the general approach to KBS development were discussed, followed by a discussion of issues relating to the approach taken to user interface design and human factors. At the general level the questions covered the following topics:

- the length of experience in developing KBS
- the hardware and software being used (in particular whether expert system shells were being used or whether systems were being programmed from 'scratch')
- whether the systems were being developed for 'in-house', or commercial purposes
- whether the systems being developed were 'stand-alone', or to be integrated within conventional software
- the development methodology (if any) being used. In particular the use of rapid prototyping.
- the background, and training of those involved in system design (e.g. conventional DP backgrounds or, AI specialists, human factors specialists etc).

At the more specific level questions concerning factors effecting user interface design covered the following topics:

- whether any human factors personnel were involved in the design team
- whether human factors specialists were consulted at any point
- whether guidelines were used, or sought, in the design of the user interface
- the knowledge elicitation techniques being used
- the level of end-user involvement,
- what point in the development life-cycle the end-user was involved
- what role (if any) the end-user(s) played in prototyping
- the classes of user for which the systems were aimed (i.e. novices/experts)
- whether the systems were to act in any form of training role

At the end of each meeting those interviewed were asked whether any follow-up work could be conducted with them. The primary aim of such work, it was explained, would be to obtain feedback on the approaches developed as a result of the analysis taking place through the interviews. Various levels of possible follow-up work were discussed, ranging from case studies of a project development at a more involved level; through to informal views, or opinions, from the developers on the approaches as they were developed. Of particular interest at the informal level would be the developers' views on the potential usefulness and pertinence of the approaches.

5.3. Interview findings

A range of application domains were being addressed through the KBS work of the organisations that were interviewed. These applications included 'help-desk' systems which were aimed at giving advice on domains such as transport legislation, computer network fault diagnoses, and software 'debugging'. These systems tended to cover tightly-bounded domains. There were also more sophisticated systems under development; these included *'intelligent'* business support systems and industrial process plant monitoring systems. These latter systems tended to have KBS modules as part of a larger integrated system. In comparison, the former, help-desk systems, tended to be 'stand-alone' systems.

Various features of KBS design practice emerged from the interviews. This section discusses these features and their likely influence upon user interface design. Some of the features highlighted, quite explicitly, the approach being taken to user interface design within the whole development process. Other, more general, features can be seen as potentially having a more indirect influence on user interface design practice.

Table 5.1. **Number of examples of types of KBS implementation tools used (software)**

Expert system shells (PC-based)	7
AI Development Environments	4
Programming languages (AI, or conventional)	2

Table 5.1. indicates the types, and number of implementation tools used to construct KBS, as reported by those interviewed. A considerable amount of work had been done using PC-based shells. In several cases there was a transition from shells to the more sophisticated, and more expensive, AI development environments, such as KEE or KES, which run on workstations (a description of these tools was given in Chapter 3; section 3.8.). The view held by most of those interviewed was that the PC-based shells had provided a useful, and relatively quick, introduction to the expert system technology, but that the shells also had limitations. They were seen as being only suitable for largely deterministic problems that were problem was well-bounded. They tended to be used for, analytic, diagnostic activities, such as fault-finding rather than synthetic, design or configuration activities.

A common viewpoint expressed here, of most relevance to this research,was that design of the physical user interface was largely constrained by the structure given by the shell itself. There was, however, a move by shell developers to make their products more flexible generally, by providing facilities to interface to other software packages, such as the 'C' programming language which is particularly suitable for implementing graphics. Greater flexibility would permit greater control in the design of the user interface. It should be noted that the viewpoint concerning present tool limitations was largely 'question-lead', that is, it was expressed in response to questions concerning the level of user interface design activity being conducted. It was not often raised spontaneously by those interviewed. From this it may be presumed that user interface limitations were not seen as crucial at this point - such limitations were outweighed by the acceptance that shells were a way of 'getting started' in expert systems. The various discussions indicated that criteria for selecting a shell did not appear to include potential user interface capabilities. Instead, issues such as cost, programming ease, and form of knowledge representation, were more prevalent criteria. There was one example, in particular, where choice of shell had been made without apparent assessment of possible user interface requirements. The (diagnostic) application concerned, so it transpired during development, required the capability for graphically representing electronic circuitry. This was beyond the capability of the particular shell chosen, and initially the developers attempted to represent the circuitry through textual descriptions, for which the shell was more readily adapted. The user would have been required to read, and enter data, through a series of *forms* which described the particular circuitry. However, mapping the electronic circuitry to a textual description proved too complex for those implementing it - undoubtedly it would also have proved complex for those who would have used the system, if it had ever been implemented.

The type of design decision just described appears naive in its lack of assessment of user interface issues. It can be explained, although only partially, if again a central objective for using expert system shells is considered; one which has already been mentioned. The objective being to gain familiarity with the technology, rather than necessarily producing robust, working, systems. This, however, can be seen to have created a vicious circle, where even the prime objective is not fully met. The tools (shells) chosen did not let the developers become familiar with the whole KBS development process. On the contrary the tools' limitations obscured other integral components of that process, such as user interface design. The trend which is apparent in the move towards the use of more sophisticated development environments may be construed as a realisation of these limitations, and signal an intention to build more robust KBS.

Programming languages, were seldom used to build systems from 'scratch'. The primary reason given for this was that to do so would be too time-consuming, especially where the initial objective is to deliver concept demonstrator systems or rapid prototypes. Shells were seen as far superior for these purposes. A further, related, factor against the use of programming was that many of those involved in KBS development were not familiar with AI programming languages, and a considerable learning period may have been required to become familiar with them. Shells, which provided a fairly 'high-level' language in which to construct the knowledge base, along with a built-in inference mechanism, were seen as a more feasible approach to at constructing KBS, at least to the prototype stage. As table 5.2. indicates, much of the KBS development had only reached the prototype stage, and few operational systems were reported.

Table 5.2. 'Working' KBS vs 'prototype' KBS (reported)

Number of systems in working use	5
Number of systems at prototype stage	14

Table 5.2. indicates the scarcity of working systems as reported by those interviewed. In the main, development had only reached the prototype stage. Even the definition of a working system, is open to some debate - at least one of these systems had become redundant after some initial use as the users had soon learnt the underlying rules and no longer required the system (this particular example would indicate a system with a tightly-bounded knowledge base, that is one with few rules and relatively transparent reasoning).

Leaving aside the debate on what constituted a working KBS, the trend highlighted in Table 5.2. has some definite implications for user interface design activity, particularly concerning the issue of user involvement. The large number of prototypes suggests that many of the systems had not been rigorously tested yet, especially not tested with end-users. It is foreseeable that such testing might throw light upon problems which had not been illuminated up to the point of prototype construction. Further support for this suggestion comes if consideration is given to the lack of prototype evaluation with users, which was indicated, this is indicated in Table 5.3.

Table 5.3. **Uses of prototyping (stated by interviewees)**

To demonstrate concept to management	4
For evaluation with expert(s)	4
For evaluation with user(s)	2

In keeping with the general state of KBS development, the prototyping process was a central feature of the development process being conducted in the work of those interviewed. Three roles of the KBS prototype were distinguished in these interviews. Significantly for user interface design, the priorities assigned to the roles varied. Evaluation of prototypes with end-users took second place to evaluation with experts. Such a finding is consistent with findings described by others. It is in line, for example, with Berry and Broadbent (1987) survey which reported the tendency for expert system evaluations to focus on the quality of a system's advice and decisions, and the correctness of the reasoning techniques used. Evidence from the interviews carried out in this work suggested that any evaluation that was conducted centred very much on the expert(s) evaluation and the testing of program accuracy, rather than on evaluation of program 'utility' with end users (cf Chapter 3; section 3.5.; Gaschnig et al 1983). Even where some contact with end-users in the development process was reported, there was no evidence that this was a particularly formal evaluation, beyond asking for fairly superficial opinions of a prototype.

A poignant example of the mismatch which can occur when consideration of end user requirements are not fed into the evaluation process was given by one of the interviewees in this research. It concerned a KBS which had been developed in close association with an expert, but no reference to the end-users up to the point of the system's introduction. The aim of the system was primarily to replace a human expert who previously had answered inquirers phone-calls concerning computer network problems. The KBS would act in a similar 'help-desk' role, conducting diagnoses of problems for the inquirers. On introducing the system it was found that the questions which the system responded to, that is the problems that the system could diagnostically address, were deemed totally inappropriate by the users themselves. They did not consider the questions to be the sort of questions that they asked when they called in the human expert. As a result the system was redundant and was not used.

In the context of the interviews conducted in this work, the argument for user evaluation is strengthened when considering the nature of the users that were reported as potential, or actual, end-users of the KBS (see Table 5.4.).

Table 5.4. **Classes* of target user of KBS (intended or actual).**
*By knowledge of application domain

Novices	4
Semi-expert	2
Expert	2

The classification of users in table 5.4. is based upon a broad distinction between end-users who were unfamiliar with the tasks included in the KBS application (novices); end-users who had some familiarity and expertise in the KBS application tasks (semi-expert); and end-users who had considerable application expertise (expert) and for whom the KBS would act in a more subservient task support role. Few of the systems were intended to support an expert, most being designed to support novices. This would tend, once again, to highlight that the systems being developed were not particularly complex, in their depth of domain knowledge.

The third role of prototyping identified (refer back to Table 5.3.), concerned the use of a prototype to demonstrate to management the concept of a KBS. This is quite a distinct use compared with that of evaluation, just discussed. It is however, an interesting facet, and not one which appears to have been considered in research and academic approaches to KBS development. The role of the prototype here was one of 'concept demonstrator'. Developing a demonstrator for management was seen as a necessary stage in development which preceded the development of more developed prototypes which would be used for more detailed evaluation. Many of those interviewed expressed the importance of 'selling' the idea of KBS to management. Management usually meant people in other departments in the organisation who would potentially fund the project. Most of the development projects described were 'in-house' developments (see Table 5.5.) but in some sense involved a 'client' or 'purchaser', in that the projects would be supported, financially, by a department outside the one that was building the KBS. This external department would also provide the 'problem' to be addressed by the KBS application. Few of those interviewed were involved in developing KBS systems commercially, for clients in the more conventional sense - that is for people outside the

organisation. However several did not preclude this as an added benefit if a completed system could subsequently be marketed externally.

As mentioned above, a form of 'pseudo-client' relationship (cf. chapter 4; section 4.7.) existed in most of the in-house development projects described. As a result, to some extent the constraints which exist in bespoke projects, such as limitations on time and budget, were also apparent in these projects. These constraints were explained as having an effect upon the approach taken to project development, and as a corollary as having an effect upon the approach to overall user interface design. One group, in particular, explained that the reason for lack of end-user involvement in a project being undertaken at that time, stemmed from the limitations on time and money imposed by the manager funding the project. The group responsible for the KBS development were actually concerned by the lack of reference to end-users, and were, it appeared, quite aware of user-centred design issues which might be related to their project. However, they were unable to act on their concerns because of the restrictions made by the manager, who would not fund any user evaluation at this stage.

Building 'concept demonstrators' for the reason described, is not an issue in most traditional systems development, for example in data processing (DP). In the case of KBS development it appeared that convincing management was a very real stage in the overall development process. This can most likely be explained as a result of the 'immaturity' of the KBS technology, and as a result those being asked to invest in a project are not as familiar, or as confident, in what such investment will bring. Some of those interviewed stated that there was some considerable reluctance on behalf of management, to fund KBS projects. This problem was compounded by various factors. In particular, the difficulty in outlining the benefits which would be accrued by the implementation of a KBS. Also estimating the time, and concomitant financial cost, required for a project is problematic. This is partly a symptom, once again, of the immaturity of the KBS technology where there are few successfully completed projects to act as precedents and so provide data for the project estimation process. It is particularly difficult to state, at the onset of a project, how many knowledge acquisition sessions will be required. Also, as was the case with many of those interviewed, there was something of a learning period required to become familiar with the implementation tools - the shells, development environments or programming languages.

Table 5.5. KBS for 'In-house' and commercial use

'In-house'	6
Commercial KBS	2

One organisation was working on a government funded collaborative project (Non-Alvey).

From the descriptions of KBS development given here so far, it can be seen that there are several factors at play. There are several competing priorities, such as meeting project constraints, gaining familiarity with the implementation tools, and the difficulty of knowledge acquisition. Perhaps, then, it is not surprising that amidst these factors that the priority that was reportedly given to user interface design, and user involvement, was not high. As has been shown, prototype evaluation with end-users was scant. Added to this, those involved in the KBS development did not have a background in human factors. Table 5.6. lists the various technical backgrounds of those interviewed who were involved in KBS design. It seems highly likely that the factors of background and experience will also have a quite significant effect upon the approaches taken to KBS development. Developing this further, it can be argued that if previous experience did not include some aspect of human factors, then the approach to KBS will not automatically be sensitive to human factors issues. It is unlikely that an issue such as user interface design will feature as an obvious priority, of greater than, or equal to, the importance ascribed to other issues with which a design team are more familiar.

Table 5.6. Technical background of personnel involved in KBS design

Traditional DP / Computing	4
Engineering	2
Computer-based training	2
Finance	1*
AI research	1
Philosophy research	1

*This person was building a KBS based on their own knowledge.

There was only one report of a person with human factors expertise being involved in a KBS project. Even in this particular case the organisation concerned was one which had previous considerable experience in systems development (in the area of training and simulation) which had a human factors component which was recognised as being significant by the organisation.

Poor utilisation of human factors was also highlighted in the lack of use of user interface design guidelines. When the issue of such guidelines was raised, some did show an interest but stated that they did not know such guidelines existed. In the main though, most pointed out, as mentioned earlier, that the shells provided the structure of the interface.

Finally, the analysis of the interviews indicated that there was little attempt to use a development methodology similar to those found in traditional systems development. There was, however, one particular example of an emerging 'in-house' methodology which described, at a high-level, the stages involved in a project development. This example will be gone into in more depth in a later chapter, as the organisation concerned provided a further case study for the research. In general the approach taken to development could be described as *ad hoc*; with the development framework primarily revolving around the production of prototypes. This made development a highly iterative process. There was no evidence of progressive, formal documentation, instead the system itself was seen as constituting the documentation. Given the size of projects it could be argued that there was not the same requirement for documentation which exists in large system development projects (Bright et al. 1989).

There was little evidence of a KBS methodology. However, more than one group expressed an interest in developing one. It was considered though, to be too early, at least in terms of the organisations' own experience, to develop structured, or semi-structured development procedures. The *ad hoc* nature of development that is identified here, raises questions over how easy it would be to introduce design approaches, such as techniques from human factors. It would be difficult to apply any structure to the KBS development process which itself was 'fluid', that is, it did not not have clearly defined stage points, but instead relied upon iterative refinement. Much of the research work which was carried out later on, in some way, addressed this issue. The whole issue of integrating human factors within the iterative KBS development life-cycle will be discussed again later in more detail (cf. Chapter Nine; sections 9.7, 9.11.).

5.4. Conclusions drawn from the interviews

Figure 5.1. brings together the central features of development practice which were highlighted in the interviews, and which can be seen as having a potential bearing upon the approach taken to user interface design. All of the features have some implications for the priority which is ascribed to the user interface design activity. Figure 5.1. is a representation of the main factors involved, it does not show any interplay between the four main features extracted from the study. To try and establish how the four features relate, and effect each other, would be difficult, and is not a requirement of this work. It is highly likely, however, that the four features have some considerable effect on each other, that is they interrelate. For example, lack of awareness of human factors issues and their importance, is likely to contribute to the biased emphasis found towards establishing system accuracy over utility (i.e. expert centred evaluation). At the same time, the features shown in Figure 5.1. represent more immediate factors which were apparent through the interviews. There are undoubtedly other factors at an even higher level. These higher-level factors include the 'immaturity' of the KBS technology, which will have implications for the existing practices, and main design considerations. Similarly the objective of gaining a familiarity with the technology, as discussed in section 5.3., has an effect upon the selection of implementation tools used.

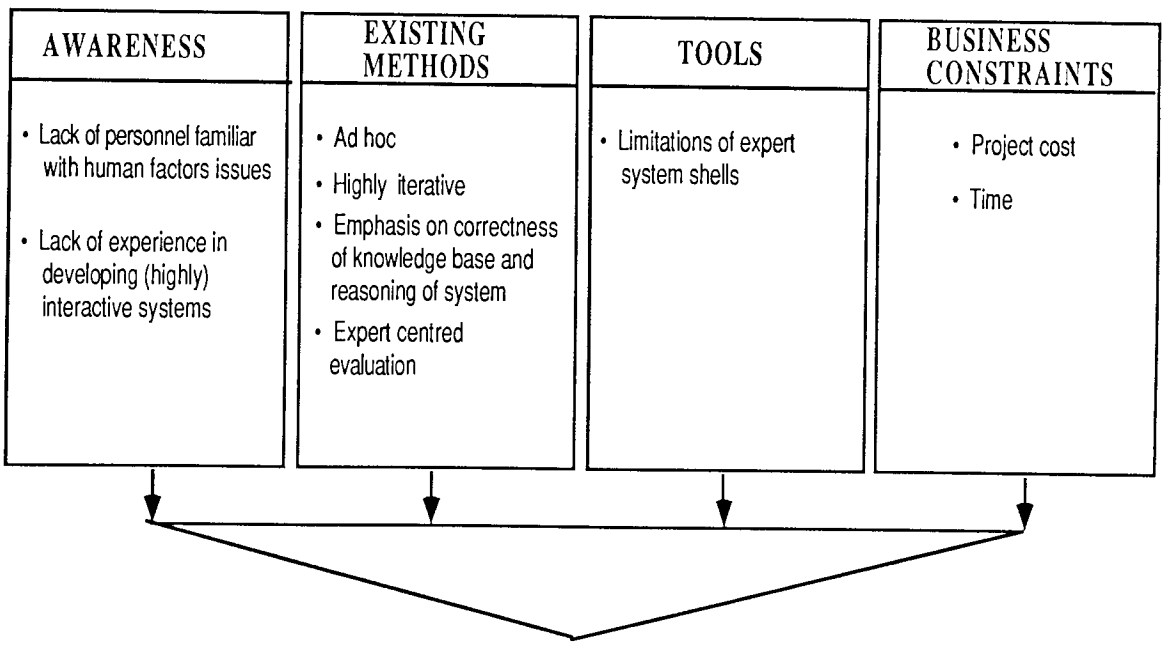


Figure 5.1. Features of design practice contributing to poor user interface consideration

5.5. Follow-up work with interview contacts

As described earlier in the chapter, a central aim of the interviews was not only to obtain more information on design practice but was also carried out to initiate contact for further follow-up work. To this end, each group interviewed was asked whether such further work was possible, and the different levels of involvement were discussed. In the event, two organisation's responded positively to the idea of my involvement in new projects which were about to begin. These were British Coal in Staffordshire; and the Technology Group of the Trustee Savings Bank (TSB) based in Manchester. The response from the remainder of those interviewed was that it was deemed inappropriate for any detailed involvement in future work. The reasons given for this were mainly concerned the fact that projects were already underway and it would be difficult to set up any *post hoc* involvement. Another reason given, concerned confidentiality, particularly where a project was being controlled (funded) by some external source. Even where more detailed involvement was not possible most groups were interested in maintaining some form of contact and offered to give some form of feedback on the approaches which were to be developed.

The case study which developed from the contact with the TSB will be described in Chapter 7. As it transpired the work which was subsequently carried out with British Coal did not go beyond an early feasibility stage. This was due to a change in corporate strategy which effected the contact department involved in KBS work. The work which it was possible to carry out, and the meetings which were held at British Coal, were useful in shaping some of the early ideas on the approaches developed. This work, although it did not warrant a complete chapter in this thesis, will be discussed in relation to the approaches that will be described in the following chapter.

5.6. Concluding remarks

The interviews had been extremely useful in reaching the objectives for this stage of the research, as described at the beginning of this chapter. Having gained a clearer picture of the factors at play in present KBS development practice, and having identified some specific problem areas concerning user interface design, the next stage was to try to tackle some of these problems. This would be done by attempting to influence positively, the approach being taken to user interface design. A set of approaches was developed to tackle some of the specific problematic areas highlighted in the *Advisor* study and in the interviews. The formulation of these methods will be described in the following chapter, along with a description of the specific areas of user interface design practice at which they were aimed.

Chapter Six

Developing Methods for Early Consideration of the User interface

Outline

The interviews had helped to further develop a picture of the current approach to user interface design within KBS development. Contacts forged through the interviews had also yielded two potential case studies of projects that were about to begin. By arranging active involvement on these projects the studies would supplement the project work going on within Intellipse, providing a first-hand and more representative view of KBS development. The author's involvement on all of these projects would centre on user interface design issues. Two options were considered for continuing the work; one option was to build some form of automated support for user interface design and to test this out with the developers; the second option was to develop manual approaches which could be employed on projects. It was decided to take the second path and to this end a 'first-pass' set of approaches, or methods, were developed which were aimed at stimulating earlier consideration of user interface requirements. In the context of the research work there were two main objectives to these methods. Firstly they would provide a firmer and more identifiable basis on which to maintain active involvement. Secondly they would help in assessing the extent to which the methods, in particular the principles underlying them, were taken up by the designers in the 'real-life' design setting. The resulting feedback could also be used to further develop and refine these 'first-pass' methods. This chapter discusses the evolution and content of these methods which essentially had two elements. The first element concerns a method for generating the potential high-level user interface requirements given an early specification, or description, of the system. The second element concerns the documentation of user interface design decisions. The chapter concludes by describing early experience of applying the methods on a KBS project which was begun with British Coal, but which subsequently terminated at an early stage due to internal restructuring of the KBS development group at the installation.

6.1. Deciding on how to progress the studies

The interviews can be seen as the end of the 'analysis' phase of the research. The information gained during this phase, which had come from the literature, the *Advisor* case study, and latterly from the interviews, formed a firm basis for the next stage of work. Acting on this information, and utilising the contacts with system developers which had been made, the next stage involved the development of approaches, or methods, which would address some of the problems which had been discovered.

Obviously, in such a large domain as human factors and KBS, it was only feasible to address some of the problems that had been found. The process of deciding which subset of problems to address was shaped by the problems that had been reported by, or were apparent in, the contact development teams. This was also the case with regard to the formulation and nature of the actual methods that were subsequently developed. For the purposes of maintaining fruitful contact with these system developers - namely within the *Intellipse* project, at British Coal and at the TSB, it was necessary to appear to

be meeting the needs of these groups. There was, in some sense, an element of 'marketing' involved. In the case of the *Intellipse* project there was a certain obligation, on the part of the project team, to the author's involvement in the project. However, the way that the contacts had been formed with groups external to the *Intellipse* project meant that no such obligation existed on the part of the TSB or British Coal. In these cases the author's continuing involvement hinged largely upon his being seen to address pertinent problems. The meetings that were to be held were to take up considerable amounts of time, which in these commercial organisations was a particularly costly commodity, one that was monitored and had to be justified to higher management. Although, as already suggested, a slightly different situation existed with regard to *Intellipse* there was still a need here to address pertinent issues as they were perceived by the senior members of the project team. As in the relationship with the external contacts, if the author's work appeared irrelevant then he would not gain constructive feedback, and hence would be unable to secure constructive involvement on the project. The need to demonstrate relevancy and applicability was increased by the fact that, as has been seen in previous chapters, the kinds of issues the author would be emphasising through his methods (essentially the need for early consideration of the user interface and its requirements) were often not seen as obvious priorities by the system developers themselves.

The situation just described raised some potentially problematic issues which the author had to be aware of when embarking on this stage of the work. Firstly, whilst the essence of this work was that it was 'designer-centred', that is it was sensitive to the existing design practice and design environments, it was important not to simply follow what the developers said they needed or wanted. If this had been done then it was likely that the author would have been consulted on relatively superficial features of design, for example one of the most common questions asked by those interviewed concerned opinions on the best expert system shell on the market. Whilst such features are important to a degree, involvement at this level would not have addressed more fundamental problems in design practice. More importantly, however, it was apparent that the system developers, to a significant extent, did not fully know what they needed - there was a lack of awareness of human factors issues in KBS design. This meant that they could not be expected to articulate, or be fully aware of, support requirements for improved user interface design. In developing the methods it was important to try and find a balance between, on one side, the requirements for 'good' user interface design practice as espoused by those in human factors; and on the other side, the apparent views of system developers as illustrated through their current approaches to user interface design.

A second potential problem may have arisen if the author had become overly concerned with the need to regularly justify the role and value of early consideration of the user interface and its design. Most certainly the strong need for justification was experienced at times during the work, particularly when setting up the other case studies. A problematic situation might have arisen if the efforts expended in attempting to justify the role of this work had interfered in some way with the formulation and execution of the actual work itself.

It might be construed that all that has been described so far suggests that the work was conducted in a particularly unconducive environment, where there was a constant need to justify, and argue the case for human factors issues. This was not the case, but as the author had initiated contact with these organisations (outside the *Intellipse* project), rather than vice versa, it was important for him to state clearly the objectives and proposed value of his involvement. That withstanding, the very fact that the other development teams were willing for me to be involved suggests that they were concerned to some degree about these types of issues - his involvement was seen as potentially worthwhile for both parties. Also the philosophy behind the case studies had from the beginning, been that any feedback elicited from involvement on projects, be it constructive or dismissive, would be valuable as it would all serve to aid the evaluation process for the methods. It would also further demonstrate the likely level of uptake the kinds of approaches that were being promoted. All feedback would indicate to what extent the teams involved were actively concerned with user interface design issues - that is, to what extent they would be willing, or able, to change aspects of their current development practice.

6.2. Deciding on the nature of support - automated versus 'manual' methods

So far this chapter has dealt, at a general level, with the decisions that were made concerning the progression of the work. At a more specific level, a decision had to be made on whether to provide support to the developers through an automated tool, such as an advisory KBS on user interface design, or through 'manual' methods, such as a set of prescribed procedures for user interface design. It was eventually decided to adopt the latter course. However, some considerable thought was given to weighing up both alternatives before this decision was reached, and it is worth describing some of the ideas that were generated in respect to the rejected alternative. This will also help to illustrate the reasons behind the final choice, and will provide a setting for describing the methods which subsequently evolved.

6.3. Ideas for automated support

As discussed in Chapter 2 (cf section 2.11) a possible route towards human factors integration in system design is to provide a set of software tools which would support the designer in user interface design activities. The INTUIT system, also mentioned previously (cf. Chapter 2; section 2.11), is an example of a potential user interface design tool which is aimed at supporting the designer of information technology products in general. The notion of such tools is akin to that of integrated project support environments (IPSEs) and computer-aided software engineering (CASE) tools which are being developed for software engineering (Bader 1988; Jones 1986, 1987). A key role of IPSEs is to provide a form of project management and control (see for example, BIS 1986; GEC Software 1987). The objective of CASE tools is to automate activities within the software development life-cycle, providing support for the specification, design, and implementation of software.

The development of tools, such as the INTUIT system, is a long-term aim which require several man-years effort (Atwood 1984). If it had been decided to develop some form of software tool within this research project, then the aims would have been less ambitious. When considering the automation 'route' it was considered that it might be feasible to construct a demonstrator system which addressed some aspect of user interface design activity. The author had some experience of, and access to, expert system shells, as well as experience in programming using AI languages. This experience might have been utilised to build a demonstrator knowledge-based system for giving guidance on user interface design.

Various 'scenarios' for the demonstrator system were produced, and these were based upon the information that had been gathered during the analysis phase of the research. Two central functions for a potential system were proposed, these will now be discussed below.

Function 1 - <u>To generate user interface considerations</u>

Here the system would provide information to the designer on the possible interface considerations for a system given the designer's general description of that system.

For the purposes of illustration, below are three 'skeleton' user-system scenarios:-

Scenario 1

User: I want to use menus

System: Consider :- Type of user
Structure of task options
Screen layout
.....

Scenario 2

User: Users will be both (task) experts and novices

System: Consider:- Dialogue style
Menus (novices)
'Forms' (experts)

Tailored help .....

Scenarios 1 and 2, would assume that the designer knew that 'menus' (in 1) and user type (in 2) had implications for the design of the interface. Scenario 3, below, illustrates a required dialogue sequence for cases where the designer did not have this knowledge.

Scenario 3

System: Have you identified the classes of user?

User: Yes

System: Are they task experts / novices / both / ? ....
.....
System: Based on your description, consider :- Dialogue style
Explanation facilities
Help facilities
.....

It was foreseeable that the advice a system could generate in the above scenarios would be limited to general pointers concerning elements of the interface to be considered by the designer. Various factors were seen to contribute to this limitation:-

- The present state of knowledge about user interface design. In particular the problem of 'generality' of design guidelines (cf. Chapter 2; section 2.8).

- The problem of 'application-specific' interface requirements. Here a mismatch can occur between what is thought to be 'good' design practice, according to general display guidelines, and what is required by a particular application in terms of displaying objects, actions, and their relationships in a domain. The system designer, using a support system such as the one connoted above, would have to exercise some degree of 'trade-off' between the system's advice and the specific design problem.

- The problem of hardware and software specific features. The implementation tools used, invariably place some constraints, often considerable ones, upon the features that can be provided at the user interface.

- Developments in display technology (both in hardware and software) create an 'unstable' environment for providing information on interface features.

The existence of these factors lead to the proposal for a second main function to be incorporated in the demonstrator system, if it was decided to take this path. It was intended that this second function might help reduce some of the 'specific' elements of interface design.

Function 2 - To store design decisions

This function would be to store specific decisions about an interface design so that this information could be used for future systems development. In this way it was seen possible to build up specific knowledge about 'good' designs, so helping to reduce the problem of generality mentioned above. It was intended that an organisation might build up their own specific guidelines which might be particularly appropriate, and useful, to the kinds of applications that they were developing. The result would be a database of design guidelines which included both existing, general guidelines, alongside organisation specific ones. The notion of such a database has been discussed by others (for example, Christie and Gardiner 1987).

For such a function to be feasible, this module of the system would have to be modifiable, allowing the designers to enter their design experience, evolving guidelines, and application specific information. The module would basically require two components:-

Component 1 *A database of information with appropriate fields*

The specific fields would be dependent upon a classification of important elements in an interface design (for example, class of user, nature of domain, implementation characteristics).

A database entry might have the following structure:-

		Instantiated example
[application name]	------->	[Insurance advisor]
[interface item]	------->	[static menu]
[item relationship]	------->	[dialogue style]
[general or specific ('in-house') guideline]	------->	[specific]
[guideline context]	------->	[task structure]
[textual description/explanation]	------->	[The static menu was used for the following reasons.....]

Component 2 *A front-end to the database*

For:- a) interrogating the database at 'run-time'
b) amending (i.e., adding or deleting) entries in the database

The database has two potential modes of use. It might have been used as the knowledge base to support the user-system scenarios outlined under 'function 1'. Here the system would generate advice derived from the current state of its data/knowledge base. Secondly, it might be used independently as an on-line database to allow the designer to look-up relevant guidelines in a similar way that he, or she, would use a paper-based handbook.

Similar ideas to those encapsulated in the scenarios described under 'Function 1' have also been reported independently elsewhere. Lenorovitz and Reaux (1986) describe a set of tools for human factors guidance which are being incorporated as one component in a software tool for user interface prototyping. Whilst only in the early conceptual stages of design it is envisaged that the component would be made up of three mechanisms - an on-line reference library of the Smith and Mosier guidelines (cf. Chapter 2; section 2.8); a computer-based dialogue evaluator; and a consultation expert system which gives advice on a specific design instance. Of particular interest is the latter mechanism which is similar in its orientation to the ideas proposed for meeting 'Function 1', above. Lenorovitz and Reaux proposed that the expert system mechanism would provide advice on aspects of the user interface (such as the use of certain dialogue styles) based upon the designers description of a system.

Also of interest in the work reported by Lenorovitz and Reaux was the reference to the usage of Smith and Mosier's guidelines. Guidelines were also considered as a potential source for constructing a knowledge base to support the functionality proposed for the demonstrator system in this work. The user-system dialogues described in the scenarios indicated that guidelines, such as Smith and Mosier's, would provide a good basis for the knowledge acquisition stage of the demonstrator system development. An advantage of these guidelines was that they offered an identifiable source of 'expertise' which already existed in structured form. Some thought was subsequently given to how these guidelines might be represented in a computable, and manipulable form (see Appendix 5).

6.4. Reasons against developing computer-based support

As already stated it was eventually decided not to choose the 'automated' route in the context of this research project. This decision was reached by generating, and assessing those ideas described above, in the context of this research project. A combination of various factors determined the final decision. The first three issues listed below concerned more global aspects which pertained to user interface design and KBS; the last issue pertained to the project situation and the relationship that had been built up with the contacts.

- System specific factors (such as those mentioned in the previous section) make the provision of constructive advice on user interface design difficult, beyond provision at a superficial level. The production of a computer-based advice system as envisaged above, was unlikely to remove this problem. In the context of this project (in particular, the time restriction) this problem was likely to remain

even if the system had some facility for carrying out the procedures to store specific designs as described. Given the slow 'turn-over' of design projects it was unlikely that useful feedback about the system could be obtained.

- User interface design is still a relatively immature field, where no established detailed methodologies exist and where 'good' user interface design is heavily dependent upon the design approach taken at the top-level (such as user-centred design) as much as the lower-level procedures or techniques (for example, for screen design and dialogue design). The kind of system connoted above was orientated at the lower-level design issues, focussing on the 'physical' user interface issues, and would not protect against inadequacies in early stages of design (such as inadequate analysis of users and their proposed tasks). The work on *Advisor*, described earlier, had already emphasised this problem.

- Knowledge-based system technology is also immature and this questioned the underlying efficacy of the proposed automated approach in the short term. Obviously the nature of research is that it is speculative and as such there would be little reason against attempting to employ KBS techniques. However the problem was compounded by the fact that not only is KBS technology immature, but so too was the proposed application domain (i.e. user interface design). A problem would arise when evaluating the efficacy of the automated tool. Potentially, it would be difficult to make distinctions between possible inadequacies in the KBS, and inadequacies in the user interface design knowledge employed by the KBS.

- There was a potential danger in trying to force the problem to fit the technology, that is forcing the problems which had been identified during the research into the KBS paradigm. This might not only have proved infeasible (see the above point) but might also have led to a betrayal of the 'designer-centred' approach taken to the work. It was likely that the construction of a demonstrator KBS would necessarily divorce the author, to a large extent, from the close involvement on 'live' development projects which had been secured with the contacts. It was desirable to implement a strategy for the subsequent work which would maintain involvement on these projects.

Against the background of these issues it was decided to become involved in projects via the development of a set of manual methods for user interface design which would be employed on the projects. As shall be seen, the central principles underlying these

methods were not at all dissimilar to the principles behind automated support. Indeed, the methods could potentially be taken as a form of specification for developing automated tools.

6.5. Defining the objectives of the methods

The methods which were to be developed were tailored to the kinds of problems that had been discovered during the contact with KBS developers so far. It was important to try and find a general solution to the problems, whilst not attempting to encompass a problem area that was too wide.

In very general terms the overriding problem that had been highlighted in both the work on *Advisor*, and the interviews, was a lack of awareness of the importance of user interface design in the construction of KBS. To highlight this problem in a more specific context, some instances that were described during the interviews are given below.
(To maintain agreed confidentiality the name of the organisation associated with individual problems is not given).

Example 1.

Problem - It was admitted by one group interviewed that whilst evaluation was conducted with the expert there was little evaluation with the user. Where there was user evaluation this was reported to be very informal. Although this problem was reported specifically by one group interviewed, it is a general problem in current design practice which has been discussed elsewhere (cf Chapter 3; section 3.11.).

Requirement - There is a requirement for a means of introducing focus on the user interface component into the overall prototyping process.

Example 2.

Problem - There was a reported problem where a manager (and project financier) saw user involvement as too costly, and generally unnecessary. This problem was identified during an interview which took place when the project concerned had been running for nine months (out of a proposed total of twelve months), at this time there had been no contact with intended users.

Requirement - a means of introducing at least a minimal amount of user evaluation into the development process.

Example 3.

Problem - There was a case of an incorrect selection of an expert system shell for a domain problem which subsequently proved to require a high degree of graphical representations. This led to the project becoming infeasible and a considerable early development work being wasted. In another example a different group had found that the explanation facility of the shell they had used to implement a system was inadequate. They consequently had to build a

scratch-built explanation facility on top of the shell they were using to meet the user's requirements

Requirement - A means for aiding early assessment of the user interface requirements, in terms of the forms of representation needed (for example, the need for graphical representation). This assessment could be used, at least, for determining the feasibility of developing a system using a particular tool; and at best, for selecting an appropriate implementation tool (shell, development environment, etc.).

The specific examples described, each come from different groups interviewed (including the TSB and British Coal), however the essence of the problems that are highlighted can be traced as common aspects of present development practice by others involved in KBS. At the same time these problems described were allied to the problems experienced in the *Advisor* work. It was sensible to use these problems as 'motivators' for defining the objectives of the methods, as in so doing it made it easier to maintain a close link between the focus of the work, and the real problems of the particular developers with whom the work was taking place. Those developers involved could associate more easily with the kinds of problems being addressed.

From this assessment a set of objectives for the methods evolved. These are listed below:

- The main objective of the methods is to generate the possible user interface considerations, and hence possible requirements, at an early stage in the design of a system. It is envisaged that these considerations are based on a description of the functions of the system.

- Generating user interface considerations early on in design will consequently prompt designers to ask 'early' questions concerning the interface.

- Once some form of user interface requirements document has been generated then it is intended that this is used as guidance for prototyping the interface, i.e. it will identify those elements of that need to be 'checked'.

- The above requirements document might also be used for guidance in the selection of appropriate implementation tools (that is, choice of shell, whether to scratch-build etc).

6.6. The first-pass methods

With these objectives in mind, the next stage involved the production of some concrete procedures which could provide the impetus for maintaining involvement with the contact development teams. For this purpose, a set of methods for generating potential user interface requirements was developed.

It was possible to obtain two descriptions, or early system specifications, of KBS projects which were about to begin. One came from within *Intellipse*, the other came from British Coal. Both were in fact descriptions of KBS modules which were to be integrated in larger systems, rather than descriptions of complete systems. The KBS from *Intellipse* was a module for monitoring, and reporting on, the performance of large databases. This module was itself part of a larger module, known as ITAM (Intelligent Total Advisory Module) which was to be developed as part of the 'Designer' phase of *Intellipse* (cf. Chapter 4; section 1.1). ITAM will be described in more detail in Chapter 7). The specification from British Coal was of an analysis and advisory module of a proposed system for detecting and controlling fires in mines. These two descriptions provided a basis from which to demonstrate, through a set of procedures (the 'first-pass' methods), how it was possible, and useful, to generate user interface considerations at an early stage in development.

Each system description was analysed in terms of the implications for the design of the user interface. Essentially, there were two steps to this analysis (see Figure 6.1.).

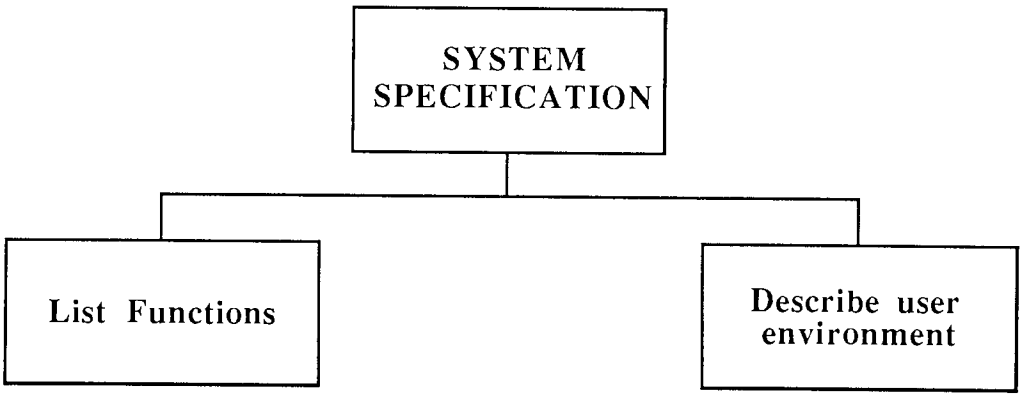


Figure 6.1. The two main steps in analysing the specification.

List Functions

The main objective of listing the functions was to establish the representations which would be required for the various inputs and outputs. To achieve this it was first necessary to identify those functions which involved some form of *input from the user to the system*, and those which involved some form of *output from the system to the user*. In many cases a function would have both inputs and outputs associated with it.

As the early system specifications contained only high-level descriptions of the system's functions (for example descriptions such as - 'monitor', 'advise', and 'diagnose'), similarly the functions, and their associated inputs and outputs, were only analysed at a high level. Functions were not broken down into their component sub-functions, as these had yet to be established.

A distinction was made between inputs and outputs (I/Os) which were part of the user-system interaction and other I/Os which were implied by the specification but which were not directly part of the user-system interaction, that is I/Os which existed from links with other modules or systems (see figure 6.2. below). It was the user *'associated'* I/Os which were of importance in this activity, the others were not of interest here. The distinction was made for purposes of clarity.

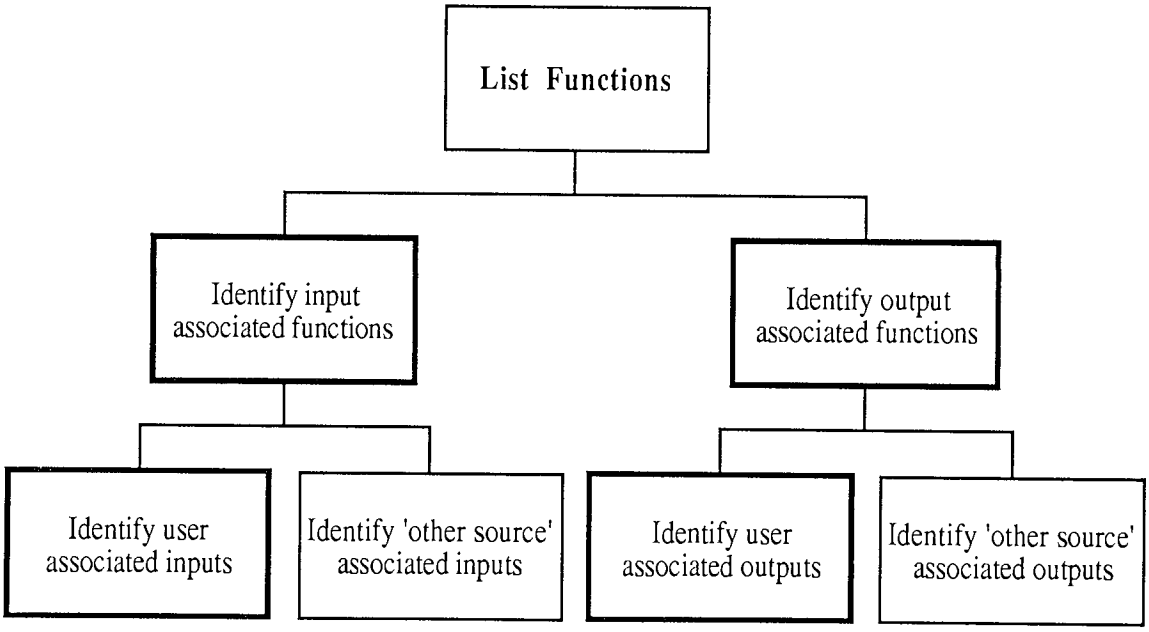


Figure 6.2. Identifying inputs and outputs in the user-system interaction

To aid the activity of distinguishing between the various inputs and outputs, a simple table was devised to help in the activity of documenting them (see figure 6.3.).

e.g.

Function	Input	Output	User	Other
Report file warning	✓	✓	✓	
Extract data trends		✓		✓

Figure 6.3. A table for documenting the functions and their related I/Os

Having extracted the I/Os that would require representation at the user interface , the next step was to try and identify what form each representation would take. To help do this a series of questions was devised which would help the designer consider what the representation requirements might be. It was intended that these questions would act as prompts for the designer. Explanations for the reasons behind the questions were given alongside the questions.

Basically two sets of questions existed, these pertained to inputs and outputs respectively. Below are a list of questions that were devised initially, like the methods as a whole, the intention was to subsequently build on them from experience, where necessary.

Dealing with inputs

i) *what information is required by the system* to facilitate a function (is it single item, yes/no, numerical, co-ordinates (graphical/pictorial etc ?).

ii) *what information is required by the user* to make an input, i.e. how is the user prompted for an input ?

iii) will it be appropriate to make the input at any time in the dialogue or is it context specific? (e.g. general system help should be accessible at any time during a dialogue, whilst explanation of a final diagnosis is only appropriate after the diagnosis has been made).

Explanation
Any input requires some form of prompt from the system, therefore one aim of identifying user associated inputs (see i, and ii) is to consider the form prompting will take (e.g. selection of an option from a menu, form-filling for data input). A second aim (see iii) is to consider the dialogue sequence which enables an input to be made.

Dealing with outputs

i) what information is required by the user to enable him to partake in the dialogue? (e.g. system and context help facilities).

ii) what is the content and nature of the information which the system must produce in order to fulfil an output associated function ?

Explanation
Outputs can be divided into :-

i. user solicited - where a user actively seeks the output of information through some form of option selection (e.g. seeking an explanation).
ii. user unsolicited - where the system generates an output, not user solicited, e.g. at the end of a diagnostic question-answering dialogue.

In the case of (i) then some form of prompt (to indicate to the system that information is required) is necessary. Here the same factors operate as those described in relation to input prompts.

Describe the user environment

The second part of the analysis concerned building a description of the user environment. It was intended that this should be carried out in parallel with the first part of the analysis just described. However, unlike the listing of functions, the description of users could not be gleaned from the specifications that had been provided. This activity would require some further consideration on the part of the developers. As part of this, the designers were prompted to identify some basic features of the intended users; these fell under two main headings (see figure 6.4.). In the context of the method as a whole, the inclusion of this element in the method was to try and encourage the designers to at least consider some high-level issues about their target users, even if they were unable, for one reason or another, to conduct a detailed analysis of users at this stage in the system development.

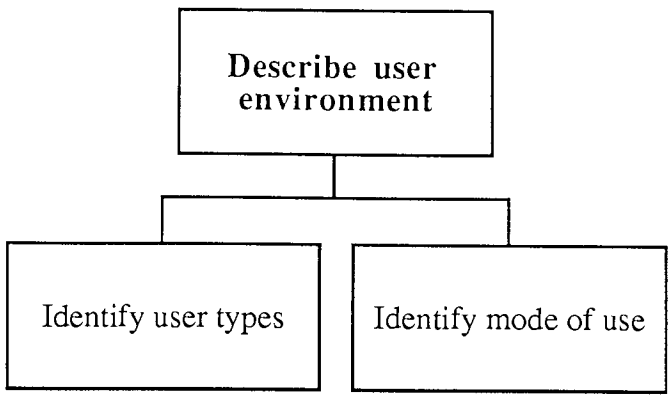


Figure 6.4. Steps in describing the user environment

Identify user types

This activity involved producing a broad classification of who was going to use the system. The typology was based around:-

i) A classification of users in terms of their level of expertise at the application task. Here it should be established whether the set of intended users were:

a) experts

b) non-experts

c) both (a) and (b)

This classification would help indicate whether it was likely that the intended users will be familiar with objects, actions and terminology in the application domain. This would have implications for:-

- the choice of dialogue style(s), that is menus, free format etc
- the likely levels of domain specific help and explanation required.

Here designers should be sensitive to potential mismatches between the content of the knowledge being engineered from the expert and the knowledge of the intended users. They should accommodate such problematic mismatches through regular system evaluation during development with users as well as experts.

ii) A classification of users in terms of their experience in use of computers in general.

a) Are the users experienced in using computers?

This has particular implications for the levels of general system help and required.

b) Do the users have experience of a particular input device, e.g. a mouse? *(this question is only appropriate if a design decision about a possibly unfamiliar input device has already been made)*

Identify mode of use

Firstly it should be established:

- whether the system is to act in any form of training role, or whether its role is purely an advisory one?

  If it is the intention that users will learn (explicitly, rather than incidentally) about the task domain through use of the system then this has important implications for the design of the system, for example in terms of the level and content of explanation given.

Secondly, although not as fundamental an issue as the one above, it should be established:

- whether the system will be used regularly or occasionally?
  This is for the purpose of assessing the potential need for reminders (either on-line or off-line) of command names and function key assignments etc.

The procedures just described could be applied to an early description of the system. The procedures had to take into account the iterative nature of KBS development and could not rely upon a detailed system specification which was set prior to design and implementation. Instead as the system specification itself evolved through an iterative loop involving some specification, followed by design prototyping and evaluation, so the user interface component could only be described in a form which began as a high-level description and then became increasingly detailed as development progressed.

Given the nature of the system description, the procedures outlined above, can only generate high-level considerations for the design of a user interface. For instance they deal superficially with more detailed design issues such as the design of the dialogue sequence. However, by generating these considerations, albeit at a high-level, they do create an awareness of, and sensitivity to, user interface issues early on in design, which it is intended will be carried through the development process. These procedures are also important in that they can help to identify potential interface requirements, such as the need for graphics, or the need for differing levels of help for different classes of user.

Appendices 6a and 6b give examples of the use of these procedures against the two specifications provided from British Coal, and from the *Intellipse* project.

6.7. Documenting design decisions

Alongside the procedures just described, one further element was devised, which was to be included as part of the method 'package'. This concerned the documentation of user interface design decisions as they were made during a system development. It was intended that this documentation procedure should begin when the first prototype was being produced and should then continue through development. As a follow on from the procedures which dealt with the early system specification the documentation procedure would maintain the focus on user interface issues.

A *pro forma* was produced (see figure 6.5.) and this covered the following features:

- The type of interface feature (for example, dialogue style, explanation facility)
- Alternative features considered (if any), and reasons for their eventual choice
- Evaluation procedure and results of that evaluation

There were deemed to be several advantages in this documentation, both for an immediate project development and for future project developments. These forms were to act as aids, by prompting questions about the user interface and its evaluation which might not have been previously considered by the designers. In the short term - within the term of a single development project - the forms were seen as potentially useful as part of system documentation overall. More importantly however, the forms encourage the designer to give a rationale to his, or her decision. In the longer term the aim behind the compilation of these forms was that, through building up and storing knowledge about past user interface designs, an organisation could develop it's own 'in-house' guidelines which might be useful in acting as more specific guidelines. Ideally these would supplement the existing, more general, user interface design guidelines. If an organisation is developing similar applications, and using similar implementation tools, then there is an increased likelihood that they can transfer some experience of a particular user interface design over to a future design.

The general idea for this documentation procedure had developed from an identified problem with existing guidelines. Ideas for some of the specific features in the *pro forma* were developed from part of a course, on user interface design, that the author had attended at the beginning of his work (Alvey Software Engineering Course 1986). The course notes had included explanations of text-editing interface 'styles' from four existing systems. The explanations consisted of reasons why particular interface features, as opposed to alternative features, were chosen by the designers of the system - an example is :-

> 'The designers chose to implement an *'undo' command,*
> rather than *no 'undo'* for the purposes of error recovery.'

Seeing the potential value of this kind of explanation to the designers in this work, the ideas were expanded upon and the following *pro forma* (see Figure 6.5., overleaf) was produced.

Application name :	Development tools : Hardware : Software :	
Interface element :		Evaluation procedure :
Reason for choice :		
Alternatives considered :		
Guidelines sought : (references)		

Figure 6.5. ***Pro forma*** **for documenting user interface design decisions**

One of the tutors from the user interface design course, just mentioned, has subsequently reported on work which is developing the notion of a representation for user interface style (Newman 1988). Newman and others are attempting to develop a representation for use in matching an interface style with a system's requirements, and for identifying inconsistencies or weaknesses in an interface design. Whilst this was not the main purpose of the author's research work there are some similarities in the philosophy behind the approach. In particular a feature that is common to both the documentation forms here, and Newman's work, centres on the idea of making explicit the designer's interface design decisions. The application of the documentation forms on the 'live' development projects will be described later, in conjunction with the case studies.

6.8. Obtaining initial feedback on the methods

A report outlining the methods was distributed to the various contacts for initial comments and feedback. The report contained a background description a list of objectives, a detailed description of the procedures for analysing a system specification, and an example of these with the system description from ITAM (British Coal received a

report which also included an example of the procedures set against the system specification that they had provided earlier). Primarily the report was intended for those groups with whom further work was to be conducted, that is the *Intellipse* project members, the TSB and British Coal. However, copies were also sent to two other contact groups (Rolls Royce Associates, and BNFL) who had said they would be interested in receiving any information for their subsequent comment.

In the main, the initial feedback was solicited by telephone contact with the various groups, although there was direct contact with members of the Alvey team from BIS. Overall the initial comments which came back from the various groups were favourable in terms of potential interest. However in the context of the research, the value of these methods and the likelihood of their application, could only really be assessed through their use on actual projects. Some minor points of clarification on the report were required at this point. In particular, there was some concern expressed by some over the use of the term 'functional specification', which had been used in the report to refer to the early system specification used to analyse inputs and outputs. Two contacts remarked that this term implied a rigid specification which seldom, if ever, existed in KBS development. As was discussed in section 6.6. the methods had been devised with an awareness of this significant development issue in mind, and were aimed at producing an increasingly detailed user interface specification as the overall system specification evolved. It appeared that this had not been made sufficiently clear in the report.

6.9. Further meetings with British Coal

The next stage in the work was to find suitable projects, in collaboration with the project contacts, in which to try out the ideas and to see to what extent the designers would take up the methods that had been devised. The following two chapters will deal with the two main projects which took place; projects which constitute the main case studies.

A further potential project had emerged from the contact with British Coal, this would have involved continuing work on the advisory system for detecting and controlling fires in mines (cf section 6.6.) . Two further meetings were held at British Coal to discuss the methods and their applicability to the proposed project. As it turned out the project was subsequently halted as a result of an internal policy decision. This decision followed restructuring of the particular KBS development division involved. The project had begun as a three man project, it was then reduced to one man, and subsequently the project was 'put on hold'.

Although the project was eventually postponed, some useful insight into the use of the methods was gleaned from meetings that were held with the intended project leader. The objective of the meetings was to carry out further work on defining the potential user interface requirements for the advisory system. This system was to act as part of a larger 'real-time' system which involved the early detection and control of spontaneous combustion and open fires in mines. Such fires are caused from a build up of gases underground. The advisory module would be linked to another module which constantly monitored the levels of gases underground; it would utilise information from the monitoring module as well as information from colliery staff. This information would then be analysed, and subsequently advice would be given. Advice focussed both upon the location and possible causes of increases in heat ("heatings") or actual fires, and upon how to tackle these occurrences. The early specification for the advisory module had been developed from preliminary meetings between the members of the KBS team at Cannock and an expert who was a qualified mining ventilation engineer. Some superficial knowledge elicitation had taken place and this had provided some indication of the content and structure of the domain knowledge. Some prototyping of the knowledge base was already taking place, this had begun using the Prolog language on a PC and was in the process of being transferred to the POPLOG development environment. There had been no work on the user interface, although the developers were beginning to think that it would be an important element. For this reason the author's involvement on the project was considered mutually advantageous.

The existing system specification contained a brief description of five high-level functions. These were :-

a) to analyse symptoms, locations and causes of heatings or open fires
b) to create hypotheses about existing or possible heatings
c) to offer advice about appropriate actions to locate heatings or fires
d) to offer advice on emergency procedures
e) determine the 'safest' route from specified locations underground

Discussion at the meetings that were held between the author and the team leader, was based around a discussion of the report which had come from the analysis of the specification using one element of the overall methods that had been devised (see Appendix 6a). The various considerations which had emerged from this exercise were then discussed. It was clear that to establish the answers to many of the questions which emerged would require further knowledge acquisition with the expert(s) and, in particular an identification of target users. It was quite apparent that up to this point there had been little definition, at least explicitly, of users. The definition did not go beyond a

reference in the specification to 'colliery staff'. A central issue in the discussions which took place concerned the question of 'who exactly were the users?'. The discussions lead to a brief, but explicit, description of the intended users, and the system's intended mode of use. This description stated that:

- The intended users were

  a) Ventilation officers who worked 'on-site' - at the coal pit

     These were technicians who supervise and monitor the day-to-day pit environment.

  b) Qualified ventilation engineers.

     These were the 'experts' and were only called to a pit if there was a major problem to be sorted out.

- In discussing the intended mode of use of the advisory module it was stated that:-

  a) the module would only be called upon in the event of a fire. Such fires only occurred once a year, or with even less frequency.

  b) the module will be accessed most commonly by the expert ventilation engineer, as opposed to the less knowledgeable ventilation officer.

Given these factors it was possible to identify some implications for the design of the user interface. Two particular implications identified, concerned user guidance, and interface consistency.

- Guidance

  The fact that the advisory module would only be accessed very infrequently meant that it was important, within design, to be aware that the user would not develop a familiarity through regular use. Therefore when this module was accessed it was particularly important that considerable guidance is provided on both accessing the relevant functions (or options) and interpreting the information from the system. Both on-line and off-line (in particular 'refresher' training in use of the system) should be considered.

- Interface consistency

  Whilst interface consistency (in terms of dialogue style, function naming etc) is always important there is a heightened need for consistency in this system. This is due to the infrequent use of the advisory module. It is therefore important to consider the need for interface consistency between this module and the other, modules, such as the monitoring module, which is accessed more regularly.

A short report outlining these issues was written up and sent to British Coal following the particular meeting. What was also stated in the report was that there was a need to develop a better description of the user's knowledge of the domain, particularly if the

less experienced ventilation officer was expected to use the system (he or she was probably less knowledgeable than the expert engineer being tapped as the 'source' for knowledge acquisition).

The project was then halted and as a result no further meetings were held. It was apparent that even if the project had continued, it would have required more resources (in terms of experts and access to users) than were then being provided. From the point of view of the research work, the meetings were useful in that they provided an initial opportunity to try out some of the ideas, and to monitor the subsequent response to those ideas.

It was apparent during the discussions in the meetings that the principles underlying the methods, that is early consideration of the types of users, and their potential requirements, were not at all obvious to the designers. Undoubtedly there were several factors which dictated this situation. However one factor which was highlighted in the British Coal example, concerned the chief motivation for the system development. The motivation for developing the system appeared to be driven largely by the desire to apply a KBS approach. The original concept for the system came from the designers themselves rather than from a set of potential users who had perceived a need for such a system and who had then gone to the systems developers. This meant that it was not automatically clear who such a system might be intended for, and consequently it was not easy to obtain a description of the users and their knowledge; features which would have had implications for the user interface requirements. This issue of *(KBS)* 'technology-lead' development and the effect it has on user involvement will be discussed in Chapter 9 during the general discussion, as it is a situation which is common to much of the present KBS work.

6.10. Concluding remarks

Having devised and disseminated the methods, the author now had a basis from which to secure involvement on projects; involvement that would be proactive. It was now possible to monitor the role that such methods had in a 'live' KBS projects. The British Coal project had provided an initial experience of applying some of the principles behind the methods. Due to the termination of this particular project, however, it was not possible to get substantial feedback. The following two chapters will deal in turn with two development projects which took place, and which provided a much richer source of feedback. Firstly, the experiences from the project in collaboration with the TSB, are described. This is followed by an account of a project which was carried out as part of the *Intellipse* project.

Chapter Seven

The Trustees Savings Bank 'IBS' project
Case Study Two

Outline

This chapter describes a case study of a KBS project which was carried out with a group from the Trustees Savings Bank. The project was concerned with the construction of a prototype KBS module, and would act as a first stage in a much larger, 'future system', project (entitled 'IBS') aimed at providing support for business managers. The chapter begins by giving a background to the KBS work which was going on prior to the start of IBS. It then describes an overview of the application domain along with a description of the initial objectives and 'set-up' of the project. The author was involved in a series of design meetings with the project team, his involvement in these meetings centred upon the application of the user interface design methods which were described in the previous chapter. The extent to which these methods could be applied is assessed. One 'product' of these meetings was a concept demonstrator system which was subsequently shown at two banking conferences, towards the end of the project. Secondly a report was produced that outlined the experiences gained and prescribed an approach for further developing the IBS work in the future. As part of this final document the author provided a report on user interface issues; that report is described here. Prior to the two demonstrations which took place, access to potential end-users had been scant. The implications of this, and other constraints which emerged through the process of development, are discussed in relation to their effect upon the level of user interface design activity which took place overall.

7.1. Background to involvement with the TSB

The Trustees Savings Bank (TSB) had first been contacted in December 1987, requesting an interview as part of the preliminary research (described in Chapter Five). An initial meeting followed in January 1988, and was held at Wythenshawe in Manchester, Lancashire, with a member of the Technology Division of TSB Group plc. This division had overall responsibility for the research and development work within the TSB Group, and within the division there was a team concerned with the development of KBS.

At the initial meeting it had been agreed, in principle, that the author might become involved in a future project at the TSB when an appropriate one came along. This would have to be appropriate both in terms of a project having a suitable (low) level of confidentiality, and in terms of the project start-point and its intended duration. Such a project, the 'IBS' project, did subsequently emerge, and it is this which is the main focus of the present chapter.

A short piece of preliminary work was also carried out prior to involvement on the 'IBS' project. This concerned another, on-going KBS project being carried out at that time, by the Technology Division. At the meeting the particular application under development was discussed and, on being given a set of screen layouts from a prototype of the application, it was agreed that the author would go away and report back on possible improvements and amendments to the screens and, in particular, the existing menu hierarchy. A report was duly written and contained, as far as was relevant to the particular application, some general recommendations about interface elements such as the use of function keys, menu selection, and prompting. Where possible these recommendations had been derived from the guidelines contained in the handbook of Smith and Mosier (cf. Chapter 2; section 2.8). Such an exercise proved instructive for the author, in that it highlighted, at first hand, some problems of guideline generality. Of the few suggestions that could be made, given the author's knowledge of the application and its concomitant domain, most revolved around specific items in the domain, in particular the way that domain objects could be structured hierarchically. This initial piece of work, although limited in what it could say *post hoc* about the particular application, was useful at a general level as it provided an introduction to the work at the TSB.

Following this initial work, and prior to beginning on the 'IBS' project in April, the set of devised methods (described in the previous chapters) was sent to the TSB for comment. These would act as a basis for making more obvious my research objectives in relation to involvement on a future project. This will be discussed in more detail later, in section 7.3.

7.2. Background to KBS work at the TSB Group

The TSB established a research group to appraise and develop KBS applications within their banking divisions. At the time of the author's involvement with them in January 1988, the research group had been in operation for three years. The first two of these years had been primarily concerned with appraisal of KBS technology and its potential, rather than 'active' KBS development which had begun in earnest in the last year. The developments were all in-house developments, involving the building of applications for other parts of the TSB. The TSB represents a large organisation (divided into four main divisions) so not only is the scope of potential applications considerable but so too is the number of potential in-house clients.

The research group at this time consisted of three members, two people from a 'traditional' DP background and one who had worked in the area of computer-based

training. These three people represented the 'core' of the KBS team, in essence the central facilitators for KBS projects. In reality there was involvement on projects from personnel in other divisions (such as DP staff and business managers). The rationale behind this set-up were partly strategic in that the situation was seen to encourage a greater appreciation of the use of KBS technology amongst business managers who would be most influential, both in terms of providing funds and personnel (experts and users), needed in any project development.

Tools

The research group had begun by using PC-based expert system shells (*Expertech's 'Xi Plus'*, and *Intelligent Environments' 'Crystal'*) for becoming familiar with, and implementing, KBS. From this early work two operational systems, using shells, had been constructed and were now in use. As a progression, TSB had begun to use a development environment, KES (Knowledge Engineering System), which was running on a DEC VAX mainframe. KES was seen as being a good choice of tool as it was Unix/'C' based and was therefore portable across a number of hardware systems which also ran under 'C'. The environment also allows greater flexibility than shells for constructing the user interface, although this requires knowledge, on the part of the developer, of the 'C' programming language and necessitates building the interface from 'scratch' rather than building from a set of general interface features which are already provided (cf. Harrington 1986). The particular application referred to earlier in section 7.1 was being developed using KES. At the time of the author's initial meeting with the TSB, the research group had just acquired another development environment - KEE (Knowledge Engineering Environment) which was to run on Apollo workstations. The 'IBS' project, to be described later in the chapter, was to be the first KBS project development using KEE.

Development methodology

In comparison with the reports from the other organisations that were interviewed during the early research, the research team at the TSB appeared to have a fairly well-developed, structured, approach to KBS development. Indeed they had formulated a methodology for KBS projects (Kielty 1987). This was an 'in-house' methodology which described a series of stages to be carried out for the development of KBS (see figures 7.1.and 7.2.).

In the main, the features of the methodology reflected the team's early experiences of KBS projects. There was, for example, an emphasis in the methodology on securing (business) management support when selecting an application. Much of the research

team's early work had involved presentations to business managers from the various divisions in the organisation. The central objective of these meetings was to establish potential applications for KBS and accompanying support for their development.

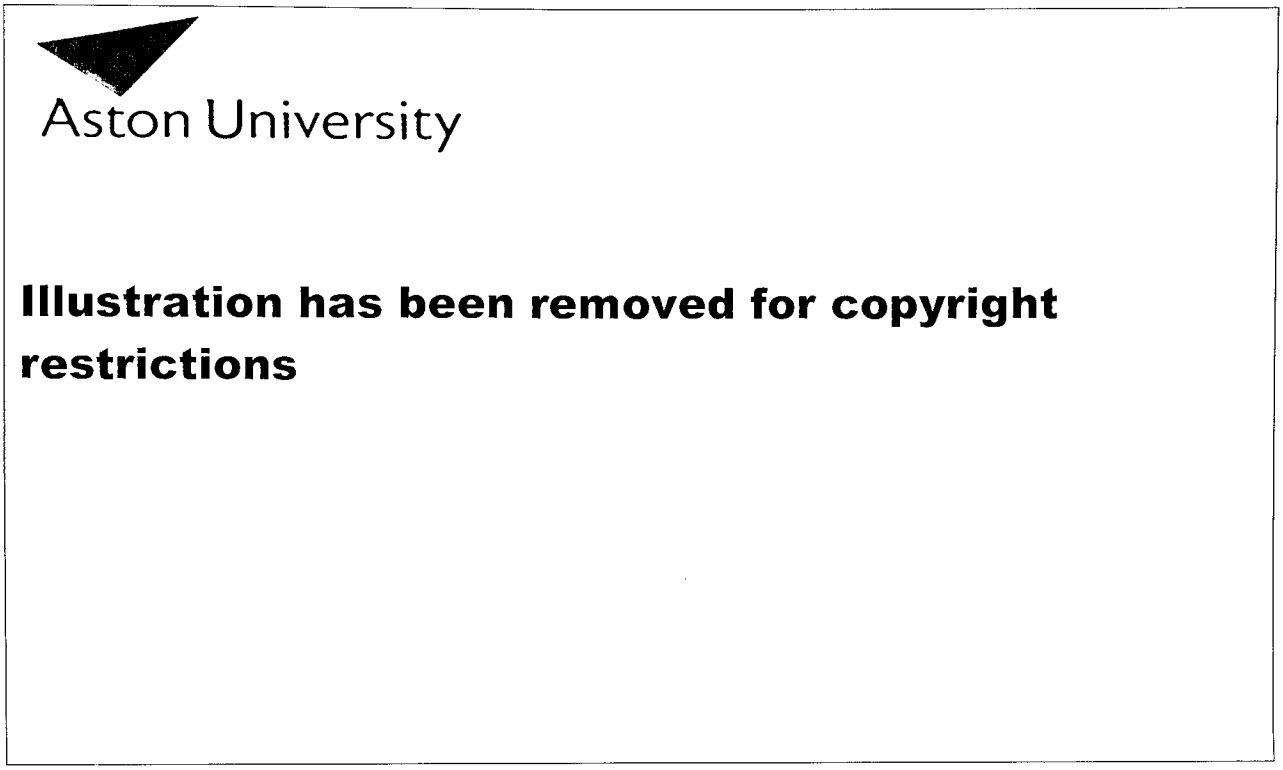


Figure 7.1. TSB's steps in selecting a KBS application (*abridged* from Kielty 1987; seminar notes)

The presentations covered a general description of KBS and more specifically a description of the relevant areas for their potential use (i.e. potential financial, or management support, applications). It was stressed during the presentations to the business managers that KBS were 'Just another tool in the DP tool kit' (Kielty 1987). Part of the reason behind such an emphasis lay in the fact that the team saw that the likely consequences of making KBS appear significantly distinct from conventional systems would be a poor up-take on the part of the managers who would see the systems as being risky, and peripheral to their needs.

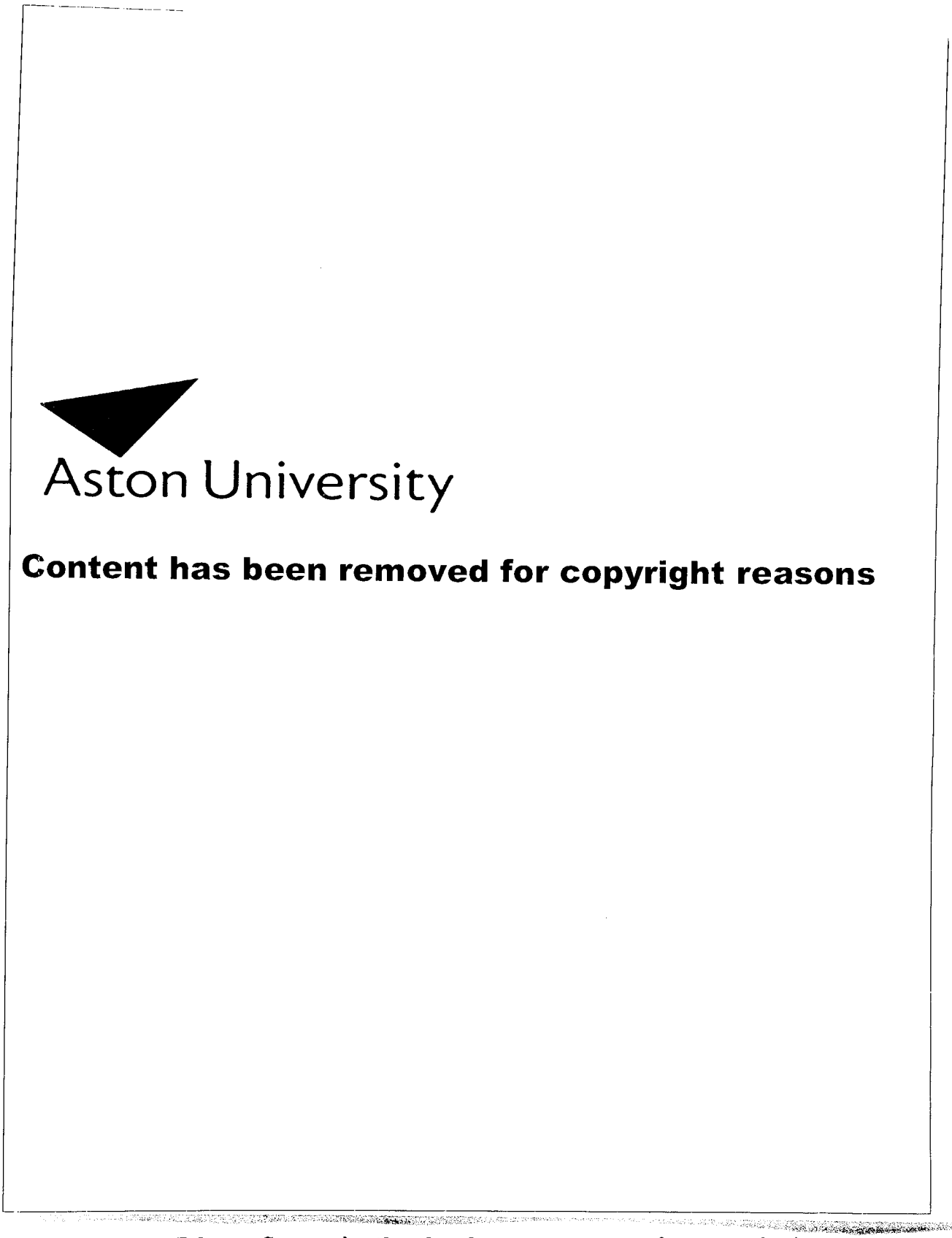


Figure 7.2. **Stages in the development phase of a domain 'component' (*abridged* from Kielty 1987; seminar notes)**

Of more specific interest in the context of this research work was the acknowledgement, at least within the espoused methodology, of the importance of user, as well as expert,

involvement. This awareness of the need for user involvement reflected early work on projects where it had been noted that of the major problems experienced, some had stemmed from 'Insufficient consideration of the user' (Kielty 1987). In particular, one system that had been built, failed to be used as a direct result of its expert-centred rather than user-centred development. It should be noted however, that there was no facility within the TSB, such as a human factors department, for calling upon specific user interface design expertise.

Although digressing somewhat from the present aim in this section of giving a general background to the TSB's KBS work, it is worth considering the wider implications of what has just been described. At one level, the appreciation of the importance of user interface design issues mentioned in the previous paragraph, probably contributed to the initial agreement to, and perceived value of, the author's involvement on a future project. It appeared that the TSB's own direct experience of system redundancy resulting tangibly from inadequate user consideration was a crucial learning experience in creating an active awareness of user issues. Taking a wider perspective for a moment, although speculative, it might be suggested that a full appreciation of the importance of user issues, and the role for human factors generally, will only follow from the direct experience of failure in the same way that the TSB experienced it on an early project. It is not as likely to follow, at least not as emphatically, from the promotion of human factors from an external agent (i.e. from human factors specialists).

Returning to the work of the TSB, it is worth considering the class of users at which the various KBS were aimed. Whilst the managers were important 'agents' in the instigation of KBS projects, these same managers had not been the actual target users of the systems which had been developed up to this point. Instead, the applications that had been addressed were targeted at less experienced bank personnel, such as junior bank clerks. An example of such an application was a system that was being developed for advising clerks on the appraisal of requests for secured loans. The view held by the TSB was that the use of expert system shells for implementing the systems was the main reason that the 'less expert' personnel had been the predominant target users. It was thought that on the whole, these shells did not provide sufficiently sophisticated facilities (such as knowledge representations, and reasoning mechanisms) to be able to support experts at their task. With the movement towards the use of workstations and the KEE environment, it was felt that it would be possible to build 'expert-oriented' systems (such as management information systems). This was not just because these tools were more sophisticated, but also because the less expert personnel, such as bank clerks, were unlikely to have as much access to workstations as they currently had to PCs. The

'IBS' project, with which the author became involved and which will now be described, was an example of a system aimed for use by managers. It was the first experience that the research team at the TSB had had of developing a KBS using the KEE tool.

7.3. Overview of the 'Intelligent Business Systems' project

As mentioned earlier, it had been agreed at the initial meeting that the author might become involved in a future KBS project, if an appropriate one emerged. One such project did emerge and began in April 1988. The objective of this project was to build a prototype KBS module as part of a long-term project which involved the development of an integrated business system. The module would serve to demonstrate some of the concepts behind the integrated system which would be made up of a suite of modular systems. The eventual KBS module would provide predictive and decision support for the manager in the formulation of a business strategy. It was envisaged that by utilising a sound business model, the KBS could reason upon data about the business; data which was being constantly updated and which could be accessed from other modules in the system.

One of the central, long-term objectives of IBS, as it was known, was to support the TSB business managers in all aspects of their work by providing up-to-date and more comprehensive information from which they could make business judgments. More specifically, the system was aimed at removing two existing problems in business practice. Firstly, it was intended that the system would conduct a preliminary analysis of the business data; then provide a graphical representation of the data along with identified 'warnings' where problems might be occurring. This would remove an existing situation which required the manager to read through reams of paper-based data from which he, or she, drew inferences, and then acted accordingly. Secondly, it was hoped that by providing various 'multi-faceted' views of the business, that is graphical comparisons of important parameters, there would be a reduction in the tendency on the part of managers to take only one, or a few parameters, as indicators of the present business situation. This tendency can be detrimental to the performance of the business, as the manager could omit to take into account important trends that might be emerging.

It was the underlying concepts behind these objectives which were to be explored in the initial prototype project. Central to the business managers' work was the task of formulating a strategy for setting the level of acceptance of loans made to clients. Each potential client is rated in terms of the likelihood that they will default on payment of a loan - such defaulting resulting in a loss to the bank. The rating, or 'credit score' is determined by the client's particular characteristics and is drawn from a model of the

overall client population and associated payment default probabilities. The manager must decide at what level to set the credit score required for accepting loan requests. When setting this acceptance level it is important that the manager, whose primary aim is profitability, finds an acceptable, positive, balance between the bank's gains accrued from repaid loans and the bank's losses which result from defaulted payment of loans. However, profitability is not just determined by the number of loans accepted. The manager must also consider other important business parameters which effect profitability, such as fluctuations in the bank lending rate, or changes in the minimum and maximum loan amounts. Some of these parameters might be under manager's control and some may not. A proposed function of IBS was to provide the manager with the ability to compare different strategies (i.e. different combinations of business parameter values) to see which would produce the best profitability. In effect, this would enable the manager to conduct a series of *'what-if'* analyses prior to implementing one particular strategy.

A second facet of IBS, but not one which was to be explored in the initial project, concerned support for the statisticians who provided the 'client' models (mentioned in the previous paragraph). These statisticians would also be able to view the business data, which the managers utilised to make their strategy decisions. However, the statisticians would utilise the data in a different way. They would regularly evaluate their credit scoring scales to assess whether they maintained an acceptable level of accuracy in assessing the repayment behaviour of clients with given population characteristics (that is whether they were accurate in predicting whether a client with a particular credit score would default or not on a loan). Evaluation would be done by comparing the predictive model with actual populations derived from the business data.

The schematic in Figure 7.3., overleaf, outlines the two basic views of business data that would be represented in IBS. As already stated the objective of the initial project was to concentrate on one of these views, by considering the interface between the manager, his, or her, business model and the business data.

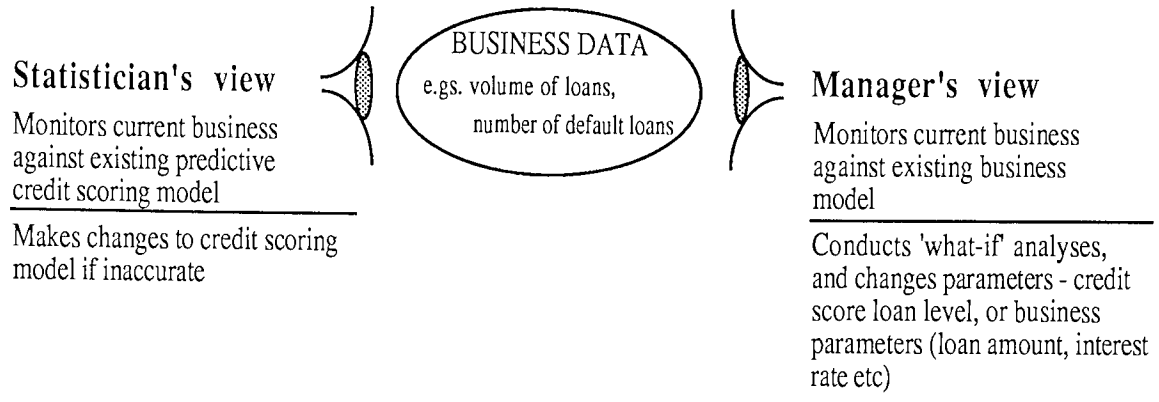


Figure 7.3. Schematic showing views of the IBS system

7.4. Objectives of the initial project stage

The idea for 'IBS' was conceived by a strategic planner who was a member of the TSB's Group Technology Staff in London. This person provided both the original ideas, and initial funding for the project, and acted as the project 'client', or sponsor. Funding was originally allocated for 150 man-days effort with the objective being to produce a first prototype, or concept demonstrator in that time. The project sponsor stated that he did not wish for there to be contact with potential end-users until a demonstrable system had been built. The reason for this appeared to be that the sponsor wanted to develop a more tangible system before exposing his ideas to users. Once the initial phase of development was complete then potential users (who at this point had not been clearly identified) might be contacted for their comments on the system, and the concept of 'IBS' as a whole.

The sponsor's criteria for success in the context of the initial phase of development would be determined by the following achievements :-

- production of a demonstrator system which was 'adequately' complete
- production of a system which is demonstrable and gives management a feel for the IBS concept
- funding for further development of 'IBS'

The reluctance on the part of the sponsor to involve users at this stage appears to have stemmed from the belief that the concept of 'IBS' was at this present point at too early a stage for such feedback to be solicited. Instead it was preferred to build a demonstrator first without reference to users. The motive for building the system - to provide a better

base of information for business managers to make strategy decisions and so improve their performance - was driven by the sponsor's view of current problems and difficulties as he saw them. However, although he had had past experience in the managers role, he no longer acted in that capacity.

At the beginning of the project it was decided to use the recently acquired KEE tool for building the prototype. It was seen that this tool would provide the more sophisticated facilities (especially graphics) which were foreseen as being important in what was considered to be an ambitious, 'state-of-the art' project. Members of the research team at Wythenshawe had recently been on a two-week training course on the use of KEE. The author was already aware of some of the constructs and principles behind the KEE package (such as frames and inheritance hierarchies) , but in order to get a better working familiarisation he spent a day in the Manchester office working through some of the short exercises contained in the tutorial manual. Although it was not possible to become totally familiar with the package in such a short period of time, the grounding the author was able to gain was useful as it gave him an appreciation of some of the constraints and problems which were to be experienced later on during the development. Such an appreciation made him more aware of what was possible and what was not with the given tool.

7.5. The author's proposed involvement in 'IBS'

Approval for the author's involvement on the project had to be gained from the project sponsor mentioned above. The objectives of the author's research, and his desire for participation in the project as a case study for the research were cited. In a letter from a member of the research team in Wythenshawe to the sponsor it was suggested that the potential value of the author's involvement, and hence his role in the project would centre on:

- observing, criticising and making recommendations on the methods and techniques that the development team use for the following functions:

  a. establishing the user interface requirements for the system
  b. designing interfaces to meet these requirements
  c. testing and evaluating the prototypes built

The author's involvement was agreed on the understanding that he produced a report at the end of the initial project outlining user interface design issues in the context of 'IBS'. The author was very keen to do so as such a report would act as an identifiable 'deliverable' and would hopefully generate an awareness of user interface design issues in the subsequent long-term development of 'IBS'. Just after the beginning of the project

the author gave a formal talk to the design team. This presentation described the various methods which had been devised (cf Chapter Six) and how these could be related to the development of 'IBS'. Above all the need for user involvement was stressed in the talk, alongside some examples of past problems in KBS design which were a consequence of neglecting users. Much of the talk was aimed more towards the project sponsor with the intention of indicating to him the need for user involvement. As is implied by the methodology described in section 7.2., those members of the research team at Wythenshawe were already quite aware of this issue, and indeed were keen for earlier reference to potential users than was being suggested at the outset of the project

7.6. The project team

At the outset of the project the development team was made up of four main people, as well as the author. Two members were from the research team at Wythenshawe. Their role would be to co-ordinate the running of the project and in particular they would be responsible for developing the prototype implementation using KEE. The two other team members came from the Credit Research division and they would be responsible for providing statistical models and data upon which to base the initial prototype. It had been decided to base the initial prototype on a relatively simple model for the purposes of concept demonstration. Finally, overall project control was with the project sponsor to whom progress was reported at monthly intervals.

7.7. Design meetings

A total of seven design meetings, in which the author participated, were held between May and November of 1988; each meeting taking place over one day. The meetings were supplemented by telephone and written correspondence between the team members. There was regular correspondence between the author and a member of the team at Wythenshawe, where interim progress and any interface issues that arose were discussed. All of the face-to-face design meetings were held in Manchester. They involved at least one member (usually two) of the research team at Wythenshawe, at least one of the team from the Credit Research division, and myself. On one occasion the project sponsor also attended for part of the meeting. Typically each meeting followed the same basic format :-

- state objectives for the day
- review progress and current problems
- "brainstorm" (around the workstation) to generate ideas for developing the prototype further
- discuss and allocate tasks for progressing work up to the next meeting

The meetings tended to revolve around an appraisal of the prototype which was being developed in KEE on the Apollo workstations by the research team in Wythenshawe.

The more immediate aim was to develop the prototype to a sufficient degree so that it could be demonstrated at an exhibition and and at a banking conference to be held at the end of 1988. This would be its first exposure to potential end-users and the first time feedback, outside of the design team, would be gleaned.

7.8. Developing the prototype user interface

The 'physical' user interface of the prototype consisted primarily of two components (see Figure 7.4.). There was a display of three graphs which plotted the relationships between various facets of the business model (such as profitability as a function of the level of loans accepted). Secondly, there was a table, or form, in which values of other business parameters could be entered (such as loan amount and interest rate). Amending values in this table would lead to a re-plotting of the relevant graph or graphs depending upon the relationship between the particular amended parameter(s) and the business model.

In the design meetings, work on refining the interface concentrated on two activities. Firstly, it was necessary to establish, with the statisticians from Credit Research, what were the important graphs and what were the important business parameters. These team members also provided the data, and necessary formulae, from which to 'run' the prototype, that is the data used to plot the graphs. For the purposes of the prototype, data was held in files in the system, however there was also some investigation into the feasibility, for future reference, of interfacing between an existing, PC-based, statistical package and KEE on the workstation.

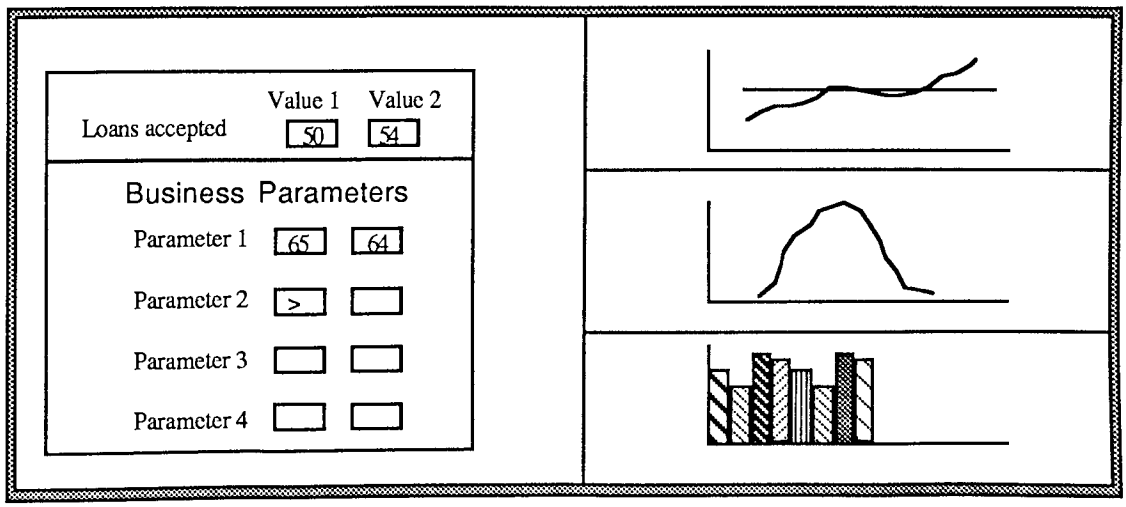


Figure 7.4. Basic screen features of the IBS prototype

The second activity during the meetings concentrated much more on the actual user interface design. In the main, discussion tended to concentrate on the following interface issues:-

- *Representation of the business information*
  including
  - display of graphs,
  - table of business parameters

- *Dialogue sequence*
  including
  - how many graphs to present at once
  - selecting graphs to view
  - zooming in on one graph
  - the sequence for comparing and weighing-up several possible 'what if' scenarios

- *Requirements for explanation during the dialogue*
  In particular explanation which described aspects of the underlying business model in light of changes made. It was decided to provide textual explanations, to back-up changes in the graph trends, of why changes in level of loans, or amendments to other business parameters, result in changes in profitability.

It was during this activity that the author was most involved. His role was to make recommendations and raise issues concerning the interface as they arose during the discussion. It was not easy to enforce a structure on this activity, largely due to the fact that the team was not working from any form of written specification to which the design team could refer. What emerged however was that the author's most effective role in this activity was to act as a 'prompt' for raising potential user interface considerations which could be associated to a particular feature that was proposed. For example, the requirement for an explanation facility, as just described above, was first raised by the author, and had not been considered up to that point. Such a facility, he proposed, was potentially important in at least two ways. Firstly by providing explanation of the model, the user would more likely have confidence in accepting the system's output. Secondly, and allied to the first reason, the manager would be able to identify any differences in the analysis methods of the system compared with his or her own methods. This comparison could help to reconcile any mismatch between the two methods and could allow the manager to use the information provided by the system in light of his understanding of the underlying model being used. This issue relates to empirical work previously mentioned (cf. reference to Lehner and Zirk 1987; Chapter 3, section 3.9.) which suggested that a 'sound' conceptual model of an expert system, even if it highlights a mismatch in problem solving style, is more important than a match between the user's and the system's problem solving strategy. A further specific feature

proposed for IBS indicated that there would be a requirement, at least in the future, for explanation to the user of underlying business models. This stemmed from the proposal that both statisticians and managers would have access to the system and that they would use the system for different, but not totally unrelated, purposes. The statisticians would use the system to assess the accuracy of their current statistical model and could, if required, alter features of an existing model if that model was found to no longer be accurate (i.e. if the model no longer accurately represented the population's loan and repayment behaviour). Some of the changes in the statistical model were likely to effect the view that the manager had of the business. The point was, therefore, that there had to be some mechanism (some form of explanation) whereby changes that the statistician user made to his, or her, model were relayed explicitly to the manager user; thus ensuring that the manager was not confused by seemingly unexplained fluctuations in the output being received.

In prompting the team to consider such issues the author acted as a 'conscience' by raising the level of awareness and concentration on what the implications were of certain design features. As was constantly emphasised by him at the time, this mode of consideration was not a substitute for actually evaluating alternative features with users themselves, but rather it should be a precursor to such evaluation. Given the sponsor's preferred distance from users at this stage it was vital that interface issues were considered in this way before design features became too entrenched. In this way it gave the design team a 'feel' for the possible issues which might emerge on exposure of the system to users.

The issue of explanation which had emerged was a high-level issue. Other, more detailed features, such as dialogue style, the naming of menu options, and the displaying of graphs, were also raised during the course of the brainstorming sessions. One further high-level issue that needed to be considered also emerged during the process of these design meetings. This concerned a problem which might occur where there was potentially a considerable mismatch between managers' current practices and that proposed and implied through the use of IBS. As has already been said, managers tended to consider only one, or at most a few, business parameters when deciding whether or not to change features of any current business strategy. The notion of IBS, however, was to provide a myriad of views of the business from which to appraise a strategy; it had been stated by the project sponsor that by so doing, the system would remove the existing deficiency in managerial practice. To this end the sponsor suggested that he thought that on entering the system the user should immediately be presented with a large number of graphs, or other statistical representations, (possibly as many as

sixteen) through which the ('multi-faceted') essence of the system might be conveyed. In a discussion which took place with the sponsor, the author attempted to highlight a potential problem with this proposal, and suggested that consideration ought also to be given to other alternative approaches. One alternative might have been to present a limited set of 'key' graphs, and allowing the user to call up others on demand). Not only might the presentation of so many graphs and figures be logistically unfeasible given limitations of screen size, but this proposal might also have deeper implications in terms of the usability and utility as perceived by the end-users (the managers). To be more specific, in relation to usability, if the system presented the users with such a mass of information then it was likely to require some considerable processing by them to establish the actual content of the information (i.e. the parameters being plotted in each graph) before considering the trend being portrayed by each graph or chart (assuming of course that they would do so). There was then, a potential problem of information overload. A second problem, in relation to utility was also highlighted. The central question here was - if the system provided such an amount of information in order to reduce the present 'narrow' view of the business being taken by the managers, might not the difference in the practice of the system and the practice of the manager be so significantly different that it would lead to the manager viewing the system as irrelevant to his, or her needs? Providing all this information was not a guarantee of a change in managerial practice, indeed it might only cause the system to become under used, or even redundant.

The above description does highlight some of the issues and potential design conflicts which were at play in the project. Most certainly the sponsor's objective of improving the current practice of managers was appreciated; the system should provide support which enhances the situation and should not merely attempt to automate current ('bad') practice. However, the central point underlying the counter-argument was that it was dangerous to make assumptions concerning the requirements for the system. Instead, it was important to evaluate some alternative ways of changing the current situation, and be aware that the careful consideration of the actual behaviour of users, both at present and on the introduction of the system, was an important factor in determining the successful utilisation of IBS. At the very least it was important to be aware that the users' acceptance of the system would be an issue and so effort should not be concentrated merely upon technical issues, such as the graphical display, or the interfacing between software packages.These other issues, while necessary, would become insignificant if the overall system did not fit in with user's requirements.

7.9. Consequences of lack of user involvement

The lack of reference to end-users during this project did not only present potential problems, like those just suggested, but also presented some actual problems for the development team as work on the prototype progressed. Whilst the brainstorming sessions amongst the team were useful, they were limited up to a point. The interface features which were generated during these sessions could only be 'best guesses', and were inevitably based on assumptions which it was not possible to test at this stage. Again the sponsor's objective to first produce a tangible concept demonstrator and then show it to potential users, can to some extent be appreciated. However, such a set-up had implications for the project as it meant that the prototype at this point was being built for an 'unknown user'. This made it difficult for the project team, especially as the team members were not as familiar with the particular users and their domain as the project sponsor, the latter at least having some notion of who the system was for. The sponsor however was somewhat removed from the actual development, apart that is, from occasional progress updates. Although it can't be proven, earlier contact with users might well have speeded up development as it would have provided a means for the team to develop a clearer picture of who they were designing the system for and what the characteristics of these people were. It has been stated by others (cf. Hekmatpour and Ince 1986) that rapid prototyping has a role as a learning experience for the developers. In this case this was being stifled somewhat by the reluctance to contact potential users where some form of evaluation of ideas might have taken place (even if it was not evaluation of an actual prototype). As was expressed by other members of the team, the development went on in something of a 'vacuum', without the existence of any clear specification, even a high-level one.

7.10. Problems in using KEE

A further problem which hindered the development team in this particular project came from the use of KEE tool. This, it should be stated was not altogether a function of the tool itself but also resulted from the fact that it was the first time that the TSB team had used KEE. As mentioned earlier, prior to this time the team had mostly used PC-based shells and more recently KES, the mainframe-based tool. With the transition to the use of KEE there was inevitably a 'learning curve' to overcome.

The degree of learning required for KEE was significant as the tool, coupled with the use of a workstation, provided a more powerful and sophisticated development environment (for example, more knowledge representation formalisms, and greater graphics and user interface capabilities). This increased power enables the developer to tackle a wider range of applications than the less powerful shells. This was one of the

main reasons behind the TSB's decision to acquire the tool. However, the increased sophistication of the development environments brings with it an increased amount of learning required by the developer to use the tool. For example, to utilise many of the tool's facilities (such as the graphical display) a considerable amount of LISP programming is required, as KEE is a LISP-based tool. Such programming is not generally required with shells which provide their own higher-level language.

An example of one of KEE's graphics facilities is the 'active image'. These are a set of prepared graphical images or icons, which the toolkit provides the developer. Examples of active images are 'dials' and 'accentuator scales', these are (directly) manipulable by the use of the mouse input device; for example the values of an object might be altered by changing the value on an associated accentuator scale. These images can be associated to objects in the application domain, and values can be passed to and from them. When work began on the IBS prototype it was thought that 'active images' could be used to display graphs and to allow the user to change values of business parameters using images such as the accentuator scale. However, on implementing this idea it was found that the system response time was totally unacceptable; it was taking up to three or four minutes to re-plot a graph following a change in any parameter value. Despite efforts to reduce this problem, response time remained unacceptable. Consultants from the vendors of KEE were called in to look at this and other problems the team were encountering. It transpired that to do what was required for IBS would need coding (programming) in LISP. It was decided to start again, building the prototype from scratch using LISP functions rather than the active images. This did speed up the response time eventually but a consequence was that some features from the first version had to be removed. This had obviously slowed up the development although as a result the TSB had learnt more about the tool and its limitations.

The case just described might be seen as specific to the situation in this project and not of wider interest. However, it transpired from talks with others that the problems experienced here were quite common. The author had had discussions with other KBS developers who were using KEE during the course of the interviews mentioned in Chapter 5. He also had a meeting with Intellicorp (the developers of KEE) whilst on a research visit to the US. From these discussions it emerged that several KEE users had found, against initial expectation, that they had to rely on LISP programming. They had also experienced problems of response time as a result of the amount of computer processing their applications required. These issues also have wider implications in that they present a further priority, or concern, which has to be dealt with and which further obscures other development issues, in particular 'soft' issues concerning user

requirements. As was highlighted in the IBS development a further reason for the deprioritisation of user issues was the 'tool-driven' nature of the project. That is, the KEE tool was selected right from the start and this resulted in a tendency to focus on technical issues.

It can be proposed that in an ideal situation the selection of an appropriate implementation tool should follow from the requirements analysis for a system. It is possible to draw a partial analogy here with the approach within traditional systems analysis where there is an emphasis on moving from a 'logical' (implementation independent) design to a 'physical' (implementation dependent) one (BIS 1987a). However, as was seen in both the IBS project and from the general view of KBS development such a situation seldom exists. This is as a result of various factors, mainly though it stems from the emphasis of producing rapid prototypes as a means of communicating requirements (either with expert or user). Inevitably the employment of a prototyping tool, such as KEE, in this way will lead to the problem of becoming 'tool-led'. In KBS development generally, as the construction of rapid prototypes are seen as central to KBS development, so the technical issues which arise during their use (like those seen in IBS) become prominent; ahead of such issues as users. This problem is probably further exacerbated by the fact that many KBS developers are unfamiliar, at least in the beginning, with the tools that they are using.

7.11. Demonstration of the HyperCard and HyperTalk tools

Given the problems that were emerging with KEE the author suggested that he demonstrated an alternative tool to the team which he believed might have been suitable to meet some of the objectives for building the IBS concept demonstrator. It was apparent by this time that too much time had been invested in producing the early prototype to revert to using any other tool at this point. However, it was possible that the information on these other tools might prove useful for the TSB team in the future.

At the time of the IBS project the author was also involved in another project ('SAM' - to be described in the next chapter) in which he was using the HyperCard tool along with its programming language HyperTalk (Apple Computer Inc. 1987), which run on the Apple Macintosh. Fuller details of these tools will be given in the next chapter. These tools were proving useful in the SAM project and the author believed there were lessons which could be carried over to IBS. The situation had arisen in IBS where the team were caught between building a concept demonstrator and an actual working prototype ('working', in the sense that it was based upon 'real' data and a partially complete knowledge base). This approach was particularly problematic given the experiences with

KEE. Together their development was being hindered by these factors - to put it succinctly the 'rapid' was being taken out of 'rapid prototyping'. The HyperCard and HyperTalk tools can be used by the developer to draw up application screens and to illustrate navigation between screens. Application prototypes can also be provided with a certain degree of functionality (for example, they can manipulate data from data fields). Whilst these tools do not have the knowledge base or inferencing capabilities that KEE, or indeed expert system shells have, they do allow you to develop a prototype of the user interface very rapidly. For the purposes of concept demonstration (the objective in the IBS project) this is ideal, as it allows the developer to obtain feedback from users very quickly.

During one of the meetings the author gave a demonstration of the tools to the project team and other members of the Technology Group. The demonstration included a review of the basic concepts behind the package, along with some basic examples which he had developed which incorporated screens from IBS. The KBS team had not been aware of these tools up to this point, however in the process of organising the demonstration it transpired that another part of the Technology Group were already using the package for the development of other on-line applications. Although an unplanned spin-off, this gave further value to the demonstration, as in uncovering this 'in-house' expertise there was an increased likelihood of using HyperCard in the future on IBS, or on other KBS development projects. This was seen by the TSB team as a definite future possibility.

7.11. Demonstrations of IBS

Two demonstrations of the IBS prototype running on KEE on the Apollo were given towards the end of the project during October and November 1988. One demonstration was given at an exhibition of banking computer applications; the other was given during a session, dedicated to IBS, of an international banking conference. Two team members from Wythenshawe took part in these demonstrations. They reported that feedback was favourable and that another division in the TSB had expressed interest in possibly providing funding for the further development of the system. Future meetings were planned to discuss this possibility.

7.12. The end of project report

As had been agreed at the beginning of the project, the project sponsor was to be provided with a report at the end of the project. This report was to outline the findings and experiences of this initial project. More importantly, the report was to describe the 'way ahead' for IBS, that is the process by which the project should be continued if funding became available.

In formulating this report the author was requested to write a section on the user interface aspects of the project, detailing what had been established so far, and the approach that should be taken to user interface design and user requirements capture. The full draft version of the report is contained in Appendix 7, however some of the points made in that report are outlined below (cf Table 7.1.). The overall report set out to clearly identify the activities of user requirements capture and user-centred evaluation from the activity of knowledge base construction alongside the expert. Particular information was identified, which would be crucial in building a picture of the user requirements. Explanations of why this information was important in the context of IBS were also given (see Table 7.1.)

Information needed	**Reasons for information**
Present tasks of user (i.e. 'decisions made')	- To establish system functions which will support (fit in) present tasks. - To establish where IBS can improve the existing situation.
Present information used to carry out present tasks	- To establish what the information "looks like" (i.e. graphical? tabular? etc) and hence the types of representation needed in the system. - To establish the source of information. - To see where more information is required and can be provided by IBS.
Variations in levels of user knowledge (in terms of domain and general computer use)	- To establish the type(s) of dialogue and on-line support (i.e, help, explanations) required.
Frequency of tasks	- Estimated frequency of functions in IBS will help indicate the degree of 'prompting' or reminding on how to access functions of the system

Table 7.1. Information for establishing IBS users' requirements

An iterative process of development was proposed in the report, whereby initially the user's functional requirements (for example, display graph, change parameter values); and subsequently the interface dialogue requirements (for example, graph selection, and form filling for parameter values) would be established. The cycle would be based around the construction of prototypes, and, continuing from the earlier work (cf section

7.11), the report discussed the potential advantages and disadvantages of HyperCard as a tool for carrying out this work.

A meeting, involving the author and two others from the project team was held to discuss the overall content of the final report which would eventually be submitted to the project sponsor. The final report would cover all of the issues which had arisen in the construction of the demonstrator and would propose an overall approach for continuing the project (of which the approach to user interface design would be a part). The overall report would not be aimed solely at the existing sponsor but would be sufficiently detailed that it would be possible that another development team (within the TSB) might take the project on. For this latter purpose, the author proposed that it would be useful to provide some record of the existing screen layouts along with explanations of why particular features had been designed so far (this would be supplementary to the existing software). He had already described, and provided copies to the team, of the 'design decision' *pro forma* for documenting design decisions (cf Chapter 6; section 6.7.). These had not been used so far in the project, largely due to the number of regular changes to the interface that had occurred during this phase. However, having reached the end of this initial project it was an ideal time to record the design decisions that had so far been made. Whilst it would be too early to see if this documentation had any long-term value in relation to guidelines for future projects, it most certainly would be of use within the single IBS project, particularly as the project might be taken up by another team. It was agreed that this documentation would be useful, and the feasibility of producing screen 'dumps' (that is snapshots of the screen layouts taken directly from KEE) was to be investigated for helping in providing this documentation.

7.13. Summing up user interface design activity in the IBS project

Involvement on the IBS project was very productive in terms of providing insight into another KBS development project. In particular, it was possible to see, at first-hand, some of the constraints being placed upon user interface design activity. These constraints became evident when, during the course of IBS there was an attempt to apply the methods which are described in the last chapter. The development path taken in IBS had implications not only for the application of the specific methods, but also for user interface design activity in a wider context. As had been the original objective, the attempted application of the specific methods, which drew upon some basic human factors principles, was a vehicle for identifying issues in the wider context.

At various stages in the project it had been possible to generate user interface considerations. The aim was to encourage the design team, and the sponsor, to consider

the user interface as a development issue which would be effected by other design decisions. This was done by considering the possible interface requirements for given proposed features (cf section 7.9.). However, as was mentioned earlier, it was not easy to do this with any rigour, mainly because documentation was difficult to generate due to the shifts in interface features which occurred during this process. The team was not working from any form of functional specification which could have been used to assess user interface requirements (that is, from which the functions' associated inputs and outputs could be analysed in terms of their representation requirements). Instead, during the course of the design sessions the team were prompted to consider the potential user interface requirements associated with particular proposed features, or functions of the system.

It is possible to summarise some of the main factors inherent in the project, all of which to a greater or lesser extent, effected user interface design as an activity within the overall development process:

- The sponsor was reluctant for the team to 'refer' to potential end-users during this initial project
- The lack of a specification from which the team could assess the system's functionality; and from which to exert some form of guidance or control on the process of building the prototype
- The development team did not have the same knowledge and insight, that the sponsor had, or more significantly the end-users would have had, of the domain and environment being addressed

These three factors combined to make user interface design difficult. Without referring to end-users, the team could not learn about the user environment, or evaluate ideas early on. By developing a better picture of the target users and user environment the team would have had knowledge against which they could assess ideas when proposing features for the system. This would have ensured a greater level of control on the generation of ideas which took place during the design sessions. At the same time the process of learning which the team underwent as the project progressed, was restricted by the lack of access to users.

Two further factors, which concerned the actual process of implementing the prototype, can be identified:

- It was the team's first experience of the KEE tool. Its use required considerable learning, particularly in relation to establishing what facilities it did and did not possess.

- The development was caught between building a concept demonstrator (which might have just highlighted navigation through 'dummy' screens) and a working prototype (i.e. having a partial knowledge base)

Overall this slowed down the time taken between generating ideas for the system and its interface, and the subsequent implementation of these ideas.

The factors just described, all effected in some way the approach taken to user interface design. The factors should not be seen as the result of an *ad hoc* approach to development but instead should be seen as stemming from the existence of 'other concerns' that were at play (that is other than user interface design). Concerns such as the wish to make concrete, what were speculative ideas, via a concept demonstrator; and the decision to use and learn about the KEE tool. Together these concerns, whilst being valid in themselves (at least in the more immediate term), had definite implications for the consideration shown to user interface design.

7.14. Concluding remarks

The author's involvement on IBS had been advantageous both for the specific project development and for the research itself. In relation to the IBS project itself, the author was able to create an early awareness of user interface considerations. During design meetings the team were encouraged to consider user interface implications of proposed features and objectives for the system. Awareness of user issues is a significant initial step, but it is not enough on its own. Instead what is required is the ability to act upon this awareness. To this end the report that was produced and submitted at the end of the project (cf. Section 7.12) made recommendations for future work. These recommendations were based upon what had been learned so far and laid down a development approach that was needed, both for maintaining the awareness of, and acting upon, user interface design issues. At the end of the day, the successful uptake and implementation of these recommendations would be determined by future project management of IBS. However, the work that had been carried out went some way in highlighting, early on in the project, that there were important user interface issues and that these issues would have some considerable bearing upon the future utility of IBS.

In terms of the research objectives, involvement on the IBS project provided further experience of a KBS development project. This allowed the author to assess the level of user interface activity and the nature of constraints upon that activity. In trying to draw general conclusions concerning these user interface design issues, it is important to consider the extent to which this project, as indeed the other case study projects, were representative of current KBS development. IBS certainly appeared to have several of

the features which were discussed in previous chapters - such as those described in the review in Chapter 3 and those which emerged from the interviews carried out (described in Chapter 5). A particular feature of the project was that it was concerned with the construction of a concept demonstrator rather than a full working system. This will inevitably have some effect upon the development approach taken, not least upon the approach to user interface design. The implications of this issue will be discussed in more detail later.

The following chapter deals with a further case study of a system called SAM which was conducted; this was carried out as part of the *Designer* module within the *Intellipse* project (cf Chapter 4; section 1.1.). As will be shown, this study presented the author with a greater opportunity for applying the user interface design methods that had been formulated, than had been the case in the IBS project. As a result the study provided a greater appreciation of the value and role of the formulated methods, as well as contributing to the picture of KBS development. A discussion of the issues which arose from both the IBS and SAM projects will be given in the general discussion of the research in Chapter 9.

Chapter Eight

The *Intellipse* SAM project
Case Study Three

Outline

This chapter describes a further case study of a KBS development project. This study was of another project which took place within Intellipse, and was concerned with the development of a KBS module, entitled SAM - a KBS to support the design and subsequent maintenance of large commercial and industrial databases. The chapter begins by putting this project in to context, by providing an update on the work that had taken place within Intellipse since the work on the Advisor system. Before describing the overall approach and lessons that were learned in the development of SAM, a background is given to the application domain of database design and maintenance. The development approach which was adopted was concerned much more with user requirements and the user interface than had previously been the case. The approach centred upon the construction of a user interface prototype. This prototype was produced prior to the prototyping of the knowledge bases, and in this way acted as a detailed concept demonstrator. The tools used to implement the prototype are described, along with a description of the resulting prototype itself. Through a series of evaluations with both domain experts and potential users (the latter provided by the BSC collaborators), the concept demonstrator was used for capturing both functional and dialogue requirements. As a result of this work the foundations were laid for future development work on SAM, which it was proposed, would continue beyond the Alvey project (which was by now almost over). The final documentation which was produced from the SAM project is then described. The chapter ends by discussing the lessons that were learned from the project in terms of the increased emphasis on user-centred development. A contributor to the shift in emphasis was the methods for early consideration of the user interface which had been developed previously in the research. The application of these methods is also discussed in the conclusion to the chapter.

8.1. The *Intellipse* Designer feasibility study

The third case study for this research came from a further project carried out within *Intellipse*. The aim of this project, entitled SAM (*SUPRA* Analysis Module), was to develop a KBS in the area of database design. SAM was concerned with the design of specific, commercially available, database systems generically known as 'SUPRA'.

The SAM development was part of the second, *Designer,* phase of *Intellipse* which followed on from the *Advisor* phase (see Chapter Four). Database design had been singled out as an activity, within DP system development, that was suitable for some degree of automated support through KBS. The project had emerged from a study that had been conducted by members of the collaborative team into the feasibility and suitability of applying KBS to activities within DP system development. This feasibility study had used as a framework, the activities described within the Structured System Design (SSD) phase under BIS's MODUS methodology. Each of the activities described in this SSD phase was investigated in terms of its applicability to KBS support. The

investigation was carried out through a combination of interviews conducted with BIS experts and an analysis of existing MODUS manuals. As a result, the activity of database design was selected as a potential area which would be addressed during the remainder of the project. It was suitable both in terms of the nature of the database design domain itself, and in terms of what was deemed feasible in the remaining duration of the *Intellipse* project (Bader 1988).

Prior to SAM, some work had begun on developing a KBS module for monitoring the performance of another database management system called TOTAL. Performance monitoring is an important database management activity as it indicates in advance where problems in the performance of a database might occur. The other module was entitled ITAM (Intelligent Total Advisory Module). An initial specification for ITAM had been used by the author as an early source for illustrating the applicability of the methods described in Chapter Six (see Appendix 6b). However, effort switched within *Intellipse* to the development of the SAM module. This switch was due, in part, to the fact that the collaborators from BSC (the immediate end-users and evaluators of the system) were moving away from the use of TOTAL to the use of SUPRA for building and maintaining databases in their DP installation at Port Talbot. This changeover represented a general trend towards the use of relational databases within commercial DP, and so it appeared that SAM would appeal to a larger market.

Whilst some work did continue on ITAM, the main focus was on the development of SAM. Due to some basic similarities which exist in the performance monitoring activity for any database, a certain amount of the outline analysis work which had been conducted for ITAM could be carried over to form part of the initial specification for SAM; these were elements concerned with performance monitoring. The SAM project provided a more extensive area of study for the author's user interface work. Even from the outset it could be seen from a superficial assessment, that SAM would require a considerable amount of user-system interaction. SAM would be more interactive than ITAM which, as well as containing little heuristic knowledge, drew much of its data from other software sources.

8.2. BIS's evolving KBS methodology

The SAM project began in earnest in May 1988. The experiences that had been gained by the parties in the collaboration up to this point were reflected in the style of approach now being taken to project development. These experiences had been gained from various developments and research which had gone on within *Intellipse*, and also from other KBS work. In particular, outside *Intellipse* BIS had been involved in the

construction, and subsequent commercial marketing, of a KBS for estimating the amount of man-years effort required for DP development projects (BIS 'Estimator' 1987b). From this experience the team were looking to apply a more structured approach to KBS development than had been the case in earlier projects.

BIS was responsible for ultimate project control of SAM, as they had been on the *Intellipse* project as a whole. Consequently the proposed approach for developing SAM mirrored very much the approach laid down for conventional DP system development. As a by-product of the work going on in *Intellipse*, BIS were keen to produce a set of KBS development standards which could be integrated with their existing MODUS standards for DP (see Chapter 1; section 1.3.). These standards support the whole development process by providing a comprehensive set of procedures, data modelling techniques (such as data flow diagrams) and documentation. Elements of the existing standards were used as a starting point for structuring the approach taken to SAM. It was appreciated that there were some dissimilarities between KBS and traditional system development, in particular the emphasis on iterative prototyping in KBS. However, it was the belief that there were sufficient similarities, at least in the basic logical progression described in the traditional phased approach, to use the existing standards as an initial framework. Through application of existing standards it would be possible to begin a pragmatic assessment of where these standards were supportive, and conversely where they would need amendments. Figure 8.1., overleaf, outlines the associations between the main stages in the traditional system development life-cycle and those in KBS development.

8.3. Implications for user interface design

Whilst it was important, particularly in a commercial environment, to try and apply a more structured approach to KBS development, applying established practice from DP system development undoubtedly had potential implications for the approach taken to user interface design. In viewing KBS as an extension of DP, where it was possible to transfer existing practices, there was a danger that the heightened level of user system interaction, and the need for careful user interface design connoted by KBS, might be obscured. The increased importance of user interface design might not become apparent until late in the KBS development. User-system interaction in a traditional DP system (such as a stock control system) can usually be described by a fixed set of functions (such as enter data, view data, and print data). Whilst an important task within system analysis is to establish the various views and paths through data that will be required by the users (such as "View a customer order"), these can be captured relatively easily. The dialogue design required to accomplish this level of interaction is commonly dealt with

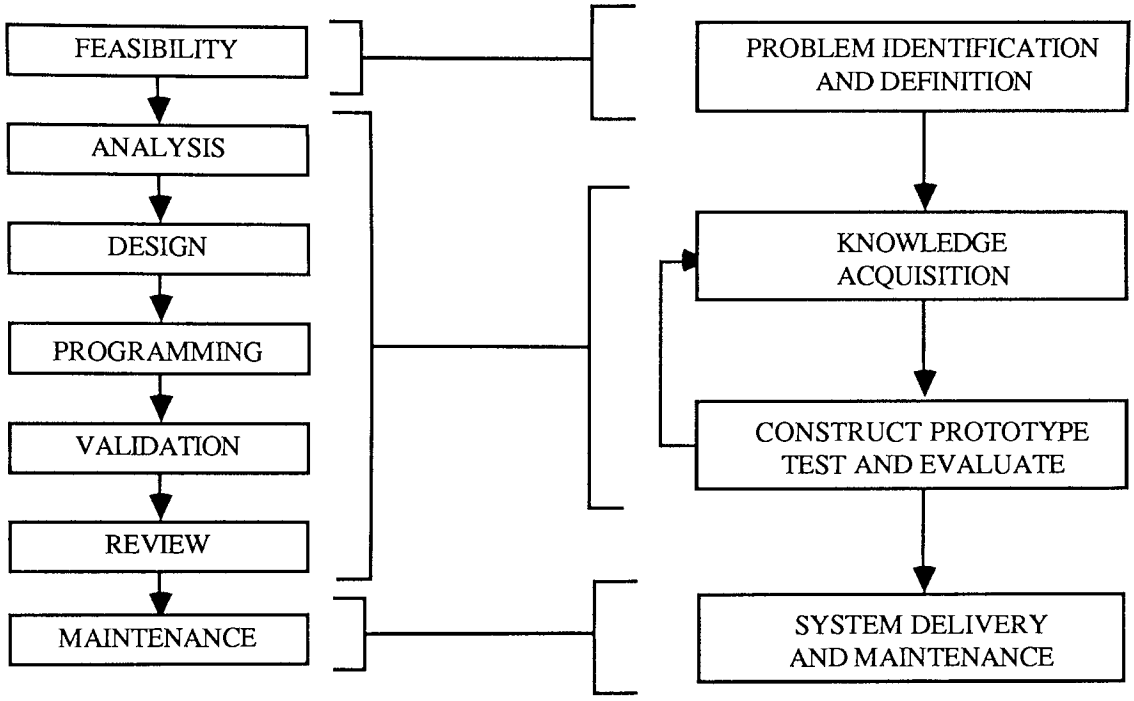


Figure 8.1. Development life-cycles for traditional systems vs KBS

by using a rigid menu hierarchy. As these system are based upon data rather than knowledge (with its inherent levels of uncertainty), there is no requirement for explanation facilities, or for long question answering dialogues which are often required features in a KBS.

8.4. The Supra Advisory Module (SAM)

As was mentioned in section 8.1., the SAM development resulted from work carried out during *Intellipse*; work which had identified database design as a domain to be addressed in the remainder of the Alvey project. Database design is a large and complex domain where there are a number of different database types, such as *hierarchical* or *relational*, all of which configure the data and data relationships in different ways. There are also a large number of activities involved in an actual database implementation. As a result, it was only feasible to focus on one particular database and the one chosen was SUPRA, a relational database produced by the CINCOM company. At the same time it would not be possible to deliver a working system within the remaining five months of the project (although *Intellipse* was later to be extended by a further six months). Consequently the objective was to get as far as possible in developing a prototype system and producing

accompanying analysis and specification documentation that would provide a basis for any future uptake and development of SAM. As it transpired, the work which was carried out on SAM was used as a central part of a project proposal submitted for a government funded research and development initiative which followed Alvey.

SUPRA was chosen for two particular reasons; firstly there was a SUPRA expert within BIS who would act as a knowledge source supplementing existing documentation manuals; secondly BSC were about to move over to the use of SUPRA, and so would be in an advantageous position to act as evaluators of SAM.

From an initial survey of the domain carried out by a team member from BIS (involving initial knowledge elicitation sessions with the BIS expert and analysis of the SUPRA documentation) three main activities were identified which it was intended SAM would support. The system would not only support the initial creation of a database but also the continued maintenance of a database once it was in operation. The three main activities are listed below:

- ***Build*** - This consists of constructing a 'physical' database design from an existing 'logical design' (created during the system analysis phase). Basically the logical design is an implementation-independent description of the data objects and their relationships, along with a description of the required views of the data. From this a physical database implementation is constructed which organises the data into files and which conforms, in this case, to the database architecture as laid down by SUPRA.
- ***Monitor Performance*** - Once a database is operational its performance can be monitored to assess whether it is performing optimally. Statistics are produced from the database which can be analysed to see whether there are current or potential problems in accessing data. These problems may arise from changes in demands on the database (i.e. as as result of an increased demand for particular information from the database, and if not ideally placed on a sector of the hardware disk this will result in poor performance), or from a sub-optimal organisation of the data files in the original database design.
- ***Tune*** - Alterations can be made to the physical layout of files (i.e. their size and organisation) in order to improve the performance of the database. Tuning might take place if a 'first-pass' design, generated during the build activity, does not meet with the particular requirements; or might occur after a database is operational where there are either changes in the demands on the database, or a general degradation in performance as indicated by the monitor activity.

It was proposed that SAM could be used to tune an existing database as well as create and tune a new one. Such a feature was important as much of the work at a DP installation, such as at BSC, centres on building upon existing databases rather than starting from scratch. For the system to be able to tune an existing database automatically, or at least for it to suggest amendments to a design, it would need a description of the existing database; that is the existing data items and data relationships would have to be fed in to the system in much the same way that they would when constructing a new physical design.

Whilst the three main activities all involved a combination of algorithmic, 'number-crunching', processes which are used to transform the logical design to a physical one, there were other processes which were identified as potential areas for the application of more 'intelligent', knowledge-based support. The main knowledge-based features for SAM along with the activity with which they were associated, are given below:

- Internal consistency checking of the logical data model as entered into SAM (Build)
- Explanation of the derived design and its predicted performance values (Build and Tune)
- Facility to allow a series of 'what-if' scenarios resulting from amendments to a design. From these scenarios the best version from the alternatives could be chosen. (Tune)
- Identification of possible causes of performance problems with an operational database (Monitor performance)

8.5. Project and team set-up

The overall project team structure and project management was similar to the previous developments carried out during *Intellipse*. BIS maintained overall project control, with BSC acting as main evaluators; by this time the other research officer at Aston had left the project and so the author was the main Aston team member.

The main development work was carried out by three members from the collaborating team; this included the author from Aston, along with two consultants from BIS and constituted the 'core' development team. An overall meeting structure was established along the same lines as had been conducted previously.

This involved:

- Design meetings involving the members of the core team. These were held at least once, often twice, a week over a six-week period. The objective of these meetings was to work on the development and refinement of the SAM prototype.
- Localised technical meetings involving the core team, the project manager and another consultant from BIS. As had been the case throughout *Intellipse* these were held monthly and were used to discuss immediate progress and future work.
- Full project meetings involving team members from all collaborating parties and a monitoring official from the Alvey Directorate. Again these had been in operation throughout *Intellipse* and were used to discuss the project progress and future work amongst the whole team.

8.6. The SAM development process

The primary aim was to produce a specification for the SAM system. This was done through a process of iterative refinement (see Figure 8.2.). Some general background to database design had been provided by the previous work carried out on the ITAM module (see section 8.1.). The ITAM work had acted as a feasibility study for the SAM work and had identified the three main database design activities of Build, Monitor Performance, and Tune. On beginning the work on SAM it was then necessary to become familiar with the specific procedures employed in those three activities in respect of SUPRA databases. The information gathered from the initial knowledge acquisition sessions was used as the starting-point for developing the SAM prototype. The prototyping work was carried out using the HyperCard package and its associated programming, or *'scripting'*, language HyperTalk. These tools will be described in detail later on, in section 8.9.

The consultants from BIS were responsible for the initial knowledge acquisition phase of the project. Initially two knowledge sources were used, from which a high-level description was derived. These sources were:-

- A BIS expert in SUPRA database design
- SUPRA documentation manuals

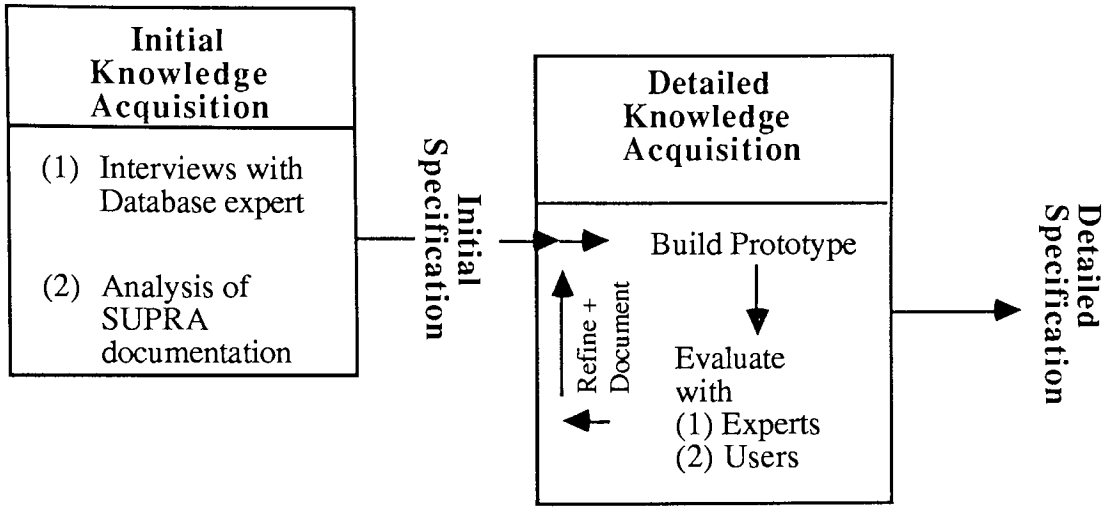


Figure 8.2. The analysis and specification process for SAM

Once the initial specification had been produced regular design meetings were held between the author and the two consultants from BIS. On most occasions these meetings were held at Aston University, the remainder being held at the BIS office in the centre of Birmingham. These meetings usually took place over half a day. The close proximity of both the University and the BIS offices meant that regular contact between the three team members presented no problem.

8.7. The author's role in the development

In keeping with the previous work the author's main role was to provide coordination of the user interface design aspects of the SAM development. This role centred upon the application of the methods that had been developed for generating user interface considerations. To this end the author was involved in the following three activities:-

1. Deriving potential user interface requirements by analysing the initial specification in terms of its associated inputs and outputs and their potential representation requirements (cf Chapter 6; section 6.6).

This was then used in:-

2. Building the SAM user interface prototype.

The first prototype version was then used in:-

3. Identifying and describing the potential end-users and their existing environment (cf Chapter 6; section 6.6).

This process is described illustrated in more detail in the schematic overleaf (see Figure 8.3.)

The activities outlined in Figure 8.3. will be now described in more detail in the following three sections.

8.8. Deriving potential user interface requirements from the initial specification

The initial specification was the basis for generating user interface requirements. The specification contained a description of the three activities (Build, Monitor Performance, and Tune) along with a details of the information required to perform each activity. The next stage was to decompose the description of each main activity into its respective set of sub-activities; and to establish the information required (by either system or user) for each activity, and the best form of representing or soliciting that information (see Figure 8.3).

This work took place within design meetings amongst the core development team. These meetings involved 'brainstorming' whereby each proposed function and its associated requirements were established and documented. Between each meeting the author worked on developing the prototype, instantiating the specification which had emerged during the meeting. In the following meeting the evolving prototype would then be assessed, and this would be followed by continuing the work on specifying the functions and their requirements. This process continued until a first version of the prototype had been developed sufficiently such that it could be shown to BSC, as end-users, and to various database design experts.

8.9. Prototyping and the use of the HyperCard and HyperTalk tools

The objective of each meeting was to further develop a prototype system. In *Intellipse* in the past systems had been built in an evolutionary way, that is the prototype consisted of a knowledge base (partially complete) upon which a physical user interface was built. Prototyping for SAM was different and began with the creation of a user interface prototype, from which requirements were ascertained and a detailed specification developed (refer back to Figure 8.2.; section 8.6.). As it transpired the user interface prototype was to be the final deliverable prototype from this project. There was not sufficient time left to begin prototyping the underlying knowledge bases which would be required to support the functionality of SAM.

As already implied, the decision to build the user interface prototype first, without any underlying knowledge base, constituted a different approach from the previous projects in *Intellipse*, which had all begun by constructing the knowledge base from the outset. The reason behind taking the user interface-centred approach stemmed largely from the

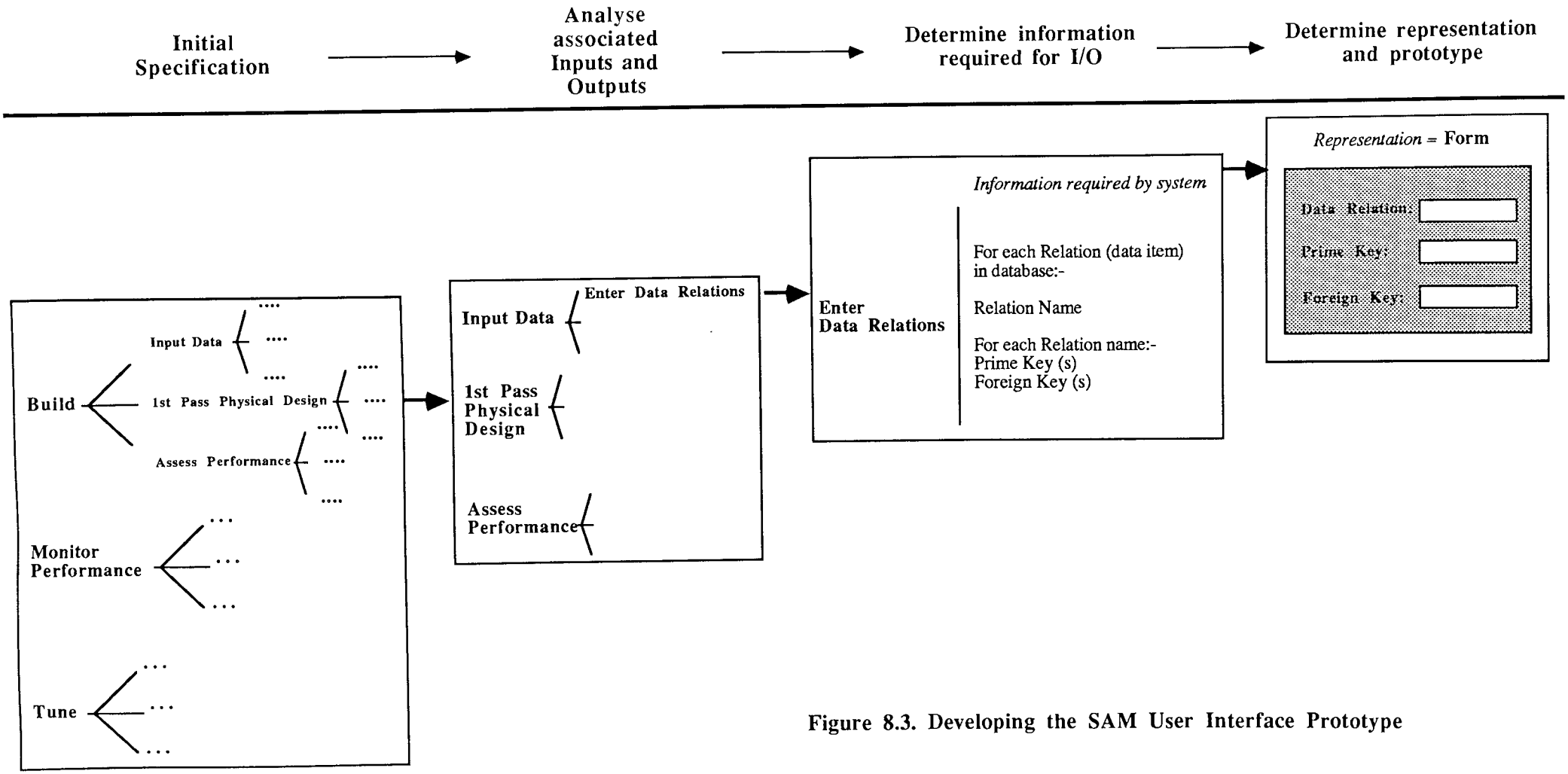


Figure 8.3. Developing the SAM User Interface Prototype

belief that the user interface would be very important part of the system. It was acknowledged from early on in the project that there would be a good deal of user-system interaction required in order to construct and tune a database. To ensure that a sufficient amount of both the functional and user requirements were captured the author proposed that a user interface prototyping approach should be adopted. The following two arguments were put forward in favour of this particular approach:-

- It would be possible to carry out user requirements capture much earlier in development, by producing a user interface prototype which, in the first instance, would act as a concept demonstrator. Requirements capture both in terms of the functionality and the 'physical' interface (dialogue) requirements could be carried out.

- It was important to be aware of the complexity of the system that was being proposed; becoming focussed on prototyping the knowledge base was likely to obscure such overall complexity. Focussing upon prototyping the knowledge base would undoubtedly have delayed the process of requirements capture. Consequently a significant amount of time would be spent working from assumptions about user requirements, which given the acknowledged level of interaction in this system was likely to be very risky. It was important that the team learned from the mistake that had been made in the development of *Advisor*, where there had been insufficient consideration of potential end-users (see Chapter 4).

To facilitate prototyping in the way just described, the *HyperCard* software package was used (Apple Computer Inc. 1987). This package runs on *Apple Macintosh* PCs and was used along with the *HyperTalk* programming language (Goodman 1987). Although not its only possible use, HyperCard can be used for the rapid prototyping of user interfaces. It has particular features which lend itself to this activity. One important feature is the speed in which quite complex user interfaces can be constructed. A concise description of HyperCard, HyperTalk, and the facilities relevant to interface prototyping is given below (see also Appendix 8):

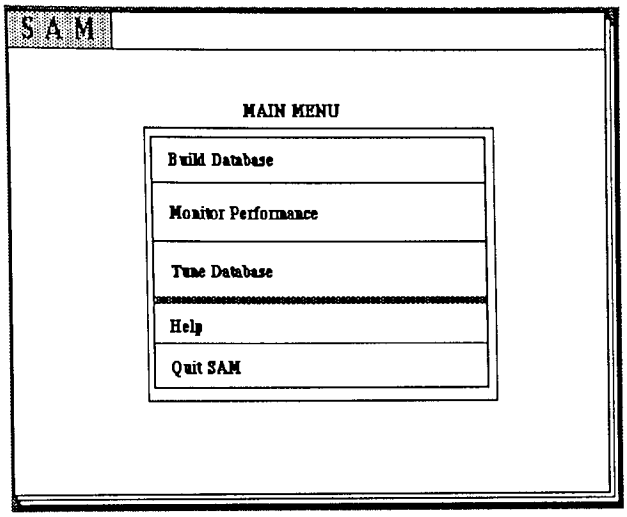
- HyperCard provides the developer with a hierarchical set of objects. At the top of this hierarchy the developer can create "stacks". Each stack consists of a series of individual screens (or "cards") which can be linked together in a sequence specified by the developer. Utilising the graphics drawing facility provided, it is possible to design screen layouts, and to

database design experts, including the expert who had been involved in work on the initial specification. Extracts from the prototype are shown in Figures 8.4. and 8.5. overleaf.

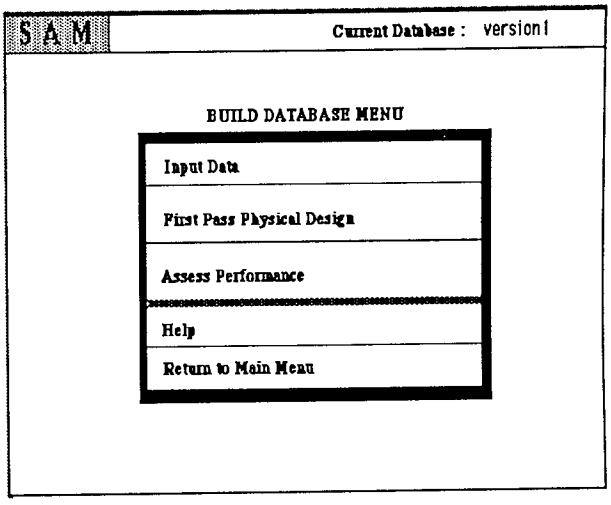
The prototype illustrated both the functionality and general features of user-system dialogue.

Functionality - The functionality of the *Build* module was conveyed through the listed menu options and the screen layouts which were linked to their associated options (see Figure 8.4.). In this way it was possible to illustrate the activities that would be required. In the case of *Build* these activities centred around the user entering in information about the data items and their logical relationships, along with specifications of what views of the data would be required and the paths through the database needed to achieve these views. From this information it was intended that the eventual system would generate a *'first pass physical design'* which would be a first attempt at a design based on the information provided by the user. This physical design is represented to the user in two forms; firstly through a data flow diagram showing the various data files; secondly through a set of numerical values assigned to parameters which describe the important characteristics of the files, such as their memory size. Finally the performance of the first-pass physical design is calculated, with the user being able to see these calculations.

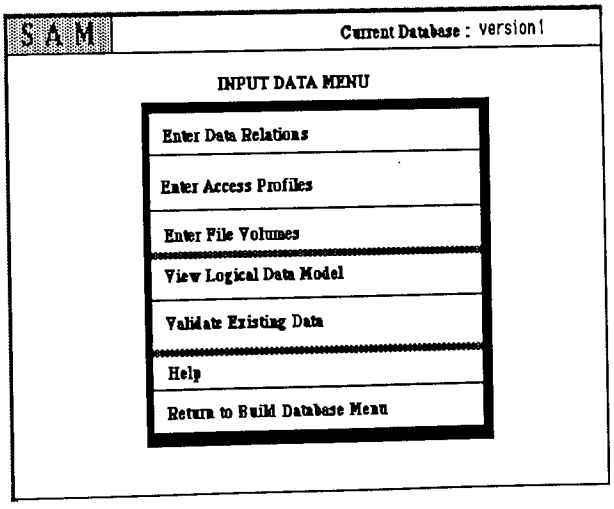
Performance is measured primarily in terms of the time taken to carry out various transactions on the database, along with the total physical disk space required for the particular database being built. It is from the first pass physical design that the user can then (in the Tune module) tune, or 'tweak', the design they have been given, until it conforms to constraints which may exist in their particular installation (for example, a maximum disk space capacity). This tuning can be carried out iteratively until a satisfactory version is established.



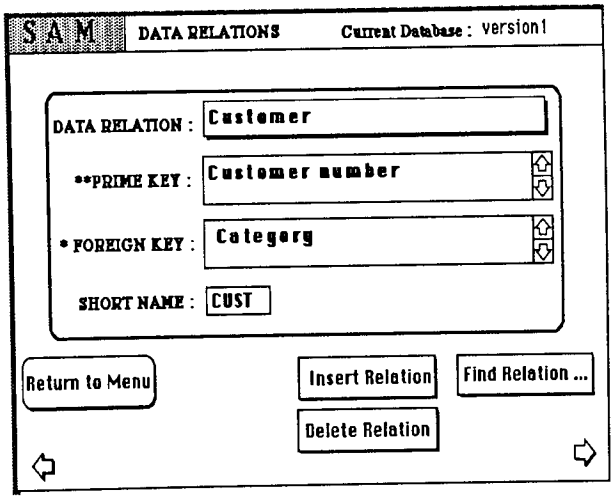
Screen 1: Select Build Database



Screen 2: Select Input Data

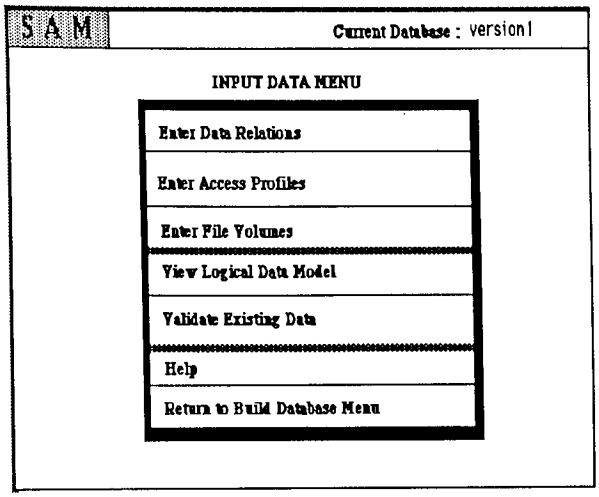


Screen 3: Select Enter Data Relations

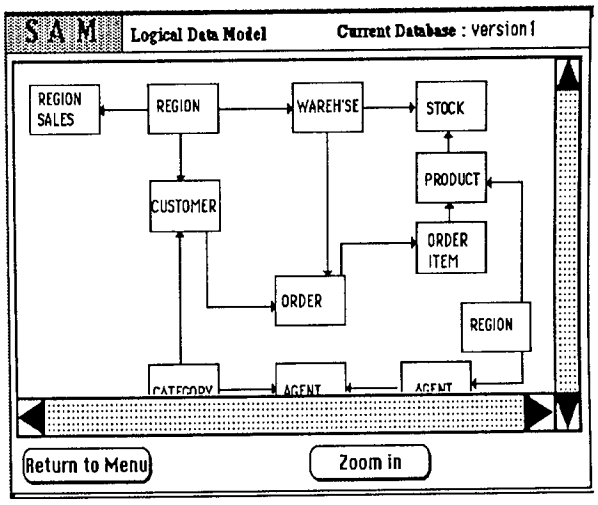


Screen 4: Form for entering a Data Relation

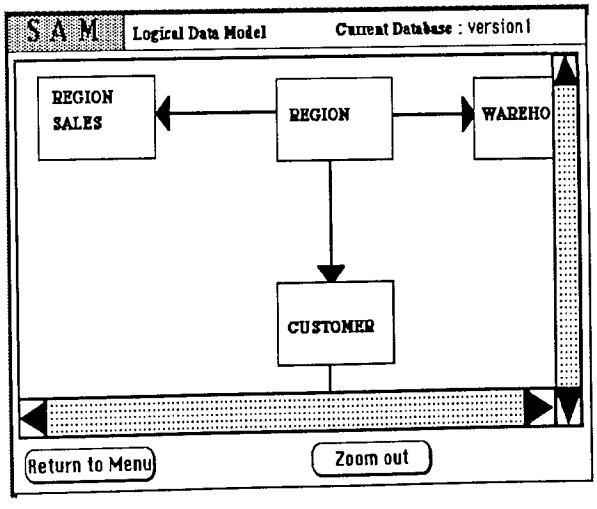
Figure 8.4. Example One: A screen sequence from the initial SAM prototype



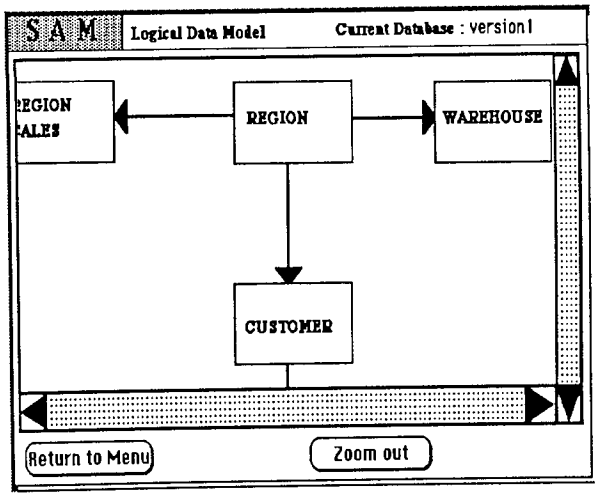
Screen 1: Select View Logical Data Model



Screen 2: Select Zoom In



Screen 3: Scroll right



Screen 4: Result Scrolled Right

Figure 8.5. Example Two: A screen sequence from the initial SAM prototype

Dialogue - Dialogue design was also represented in the first version prototype. The following elements were illustrated:

- Dialogue sequence (shown through a menu hierarchy)
- Navigation through the system
- Menu style (static menus and pop-up menus were used)
- General system dialogue (e.g. soliciting the names of current database from the user)
- Form-filling for entering data relations and file parameter values
- Interactive construction of "Access Profiles" (these are diagrams which represent paths through data)
- Scrolling and zooming in, on (data flow) diagrams
- Points in dialogue where *explanation* and *help* were likely to be required
- Reporting of inconsistencies and errors in data input

In designing the dialogue features for the prototype a conscious decision was made to illustrate what were seen as the minimal requirements for SAM if it were to be built. This was reflected particularly in the interface design features where, although it would have been possible to simulate more sophisticated features in HyperCard, the original design only employed features which were seen as the least necessary for the proposed application. For example, pull-down or pop-up menus were not simulated in the initial version; instead static menus were employed which were displayed on a whole screen (cf. Figures 8.4. and 8.5.). Such consideration stemmed from the fact that it was not known at this stage the level of sophistication that would be possible to implement in the final delivery system for SAM (i.e. the software and hardware to be used). The core development team were aware that they could become 'carried away' in simulating features in HyperCard, and as a consequence an unrealistic specification could result.

8.11. Evaluation with BSC: identifying and describing potential end-users of SAM

The prototype was shown to members of the team from BSC on two occasions. On the first occasion there was an introductory demonstration of the system during a full project meeting of all project members. A more detailed demonstration, and subsequent gathering of evaluative feedback took, place over a one-day meeting held at the BSC installation at Port Talbot, in South Wales.

The objective of the first demonstration was ostensibly to explain the development approach being taken, and its rationale, to the other team members (BSC in particular).

The objective of the demonstration at this stage was not to obtain specific feedback on the prototype itself, instead more detailed evaluation would take place later at a subsequent meeting with BSC. A background explanation which was given at the first demonstration was an important precursor to a fuller evaluation, as several of the features of the approach were novel to the other people involved. It should be appreciated that to those involved, the rapid prototyping approach and the kinds of tools being used were all quite unfamiliar elements; they were not familiar with the Apple Macintosh system with its mouse and overall (*WIMP* - windows, icons, mouse and pull-down menus) style interface, and they were not familiar with the prototyping approach as being conducted in this project. The 'briefing' that took place was to ensure that such unfamiliarity did not interfere with the main objectives of evaluation, something which might have occurred if the prototype had appeared somewhat unreal and hence distant from the evaluators' own high-level requirements. It was important to explain to the evaluators that the prototype only covered the user interface and that this was being done as a step in capturing both functional and dialogue requirements for SAM. The use of the HyperCard tool was also explained, and in particular it was stressed that neither this tool, nor the Apple Macintosh PC would constitute the final delivery system.

The objective of the subsequent meeting at Port Talbot was to obtain more specific feedback on SAM and its potential role. Two sets of issues needed to be established from this evaluation. Firstly feedback was required on the functionality and dialogue as represented in the existing prototype. Secondly, it was necessary to develop a more detailed picture of the potential users of the system. Up to this stage in the development of SAM we had only a very high-level classification of the potential end-users of SAM. This classification had included database designers and database administrators (administrators being personnel in an installation, such as the Port Talbot DP installation, who both develop and maintain databases). With the existence of the prototype we now had something more concrete from which to solicit a more detailed description of potential end-users and their environment.

An agenda for the meeting was sent down to Port Talbot in advance. A checklist was also prepared which set out those elements of the user interface to be checked; these were primarily dialogue elements. This checklist provided a guide for recording feedback comments during the demonstration. For the purposes of the demonstration a Macintosh PC was transported down to Wales. Three people from the BSC team attended the demonstration, these included the chief database administrator at Port Talbot and two systems analysts who also had responsibilities for the maintenance of the

database. The three members of the core development team travelled down to Port Talbot for the meeting. The prototype was demonstrated by running through the system three times. Any points of clarification which arose were made, and at the same time any relevant comments were recorded during each run-through. Following the demonstration there was a discussion to identify potential users of the system and gain more background information to the present activities of the database administrator and system analysts.

A description of classes of users, their current roles, and the implications for the SAM user interface emerged. This description, summarised below, was documented and would act as a reference point for furthering the development work:

Potential users of SAM

The two main classes of users were confirmed as being:-

1. Database administrators - using the system for tuning and monitoring the performance of a database.

2. Systems analysts - using the system for building a database, entering data relations and access profiles etc. Such information would be entered from existing paper-based documentation that the analyst would already have created.

A third possible class of user was also identified:-

3. Clerical staff to input certain information to the system where such information can simply be transferred from an existing off-line source, and requires little or no knowledge of overall database design (for example, it might involve such tasks as the 'keying-in' of data relations).

This third possible class of user emerged as a result of the belief expressed from those at BSC that relatively mundane activities such as keying in the data relations, or constructing prepared diagrams, might deter the more experienced administrator from using the system, especially when making slight amendments to an existing design. This was seen as a relatively common scenario. On such occasions it was suggested that the administrators would rather do quick calculations themselves ('on the back's of envelopes'). For SAM to be useful, however, it would have to be 'kept informed' of any changes, consequently this task might be taken over by clerical staff.

If these three classes of user were to use SAM then it would be important to consider some distinction between each user's view of, or interface to, the system. Certain classes of user (particularly the clerical staff) might have a different level of restricted access to the system. This was particularly important in terms of ensuring that alterations were not made unwittingly by a user who was inexperienced in database design, or indeed that the use of the system by one user did not alter another's present view, that is a user's present stored design, or information.

simulate navigation between screens by specifying appropriate links. Within each "card" it is possible to place other objects which HyperCard provides.

The main objects are:-

Buttons - a bounded area, which can be either opaque or transparent and which upon, selection with the mouse, results in some specified action, such as activating a link to another screen.

Fields - areas where textual or numerical data can be placed. This data can then, if required, be manipulated in a way that is specified by the developer.

- The specification of links between cards; actions resulting from clicking on buttons; and data manipulation in fields, is done through the use of "scripts". These scripts are programs (written in the accompanying language HyperTalk) which can be bound to a particular object and which will be called, on the selection of that object. For example, a series of individual buttons might be placed under a list of functions which has been created to represent options in a a menu; selecting one option thereby activates the button script whose instructions might be to go to an appropriate screen for that option.

- The Apple Macintosh environment includes a mouse input device. This allows both the developer of the prototype, and the end-user at run-time (if appropriate to the application), to interact using direct manipulation of objects. For example, clicking at the location of a button activates the script bound to that button.

These tools were very useful in constructing the SAM interface and making concrete the ideas which were generated in the initial specification and during the subsequent design meetings. A first version prototype was produced and then evaluated. Before discussing the results from the evaluations which took place, a description will be given of the features of the prototype.

8.10. The first-version prototype

The Build module had been the first module to be analysed and prototyped. This constituted the first version prototype. On completion of the Build module, and some work on the Tune module (to the point where there were listings of activities in menus, but without associated screens) it was decided to show the prototype to BSC and to

Mode of use

Much of the database work at an installation such as the one at BSC involves making amendments to an existing database, rather than building a complete database from scratch. There was a slight exception to this at BSC at the time as they were in the process of transferring from from TOTAL to SUPRA and this resulted in quite a significant redesign of the existing database. However, it was the Tune and Monitor Performance modules which were seen as being the most appropriate to BSC's more common needs.

Monitor Performance was an activity which occurs regularly (usually once a week at the BSC installation). Tuning a database in any significant way, however, is a less frequent activity. Significant problems of the sort that require substantial tuning are fairly uncommon. Instead BSC reported that the most common change made in respect to tuning, was to add extra "links" between data items in order to cope with changes in demands on the database. This had potential implications for the user interface to Tune, as some form of user support for occasional, as opposed to regular, system use might be required of SAM. It was noted that these implications should be borne in mind when designing help facilities for commands, function keys etc. and explanation facilities for conveying the problem diagnoses - a proposed feature of Tune. Such support facilities would serve as 'reminders' in the event of occasional system use.

Establishing the above description provided the design team with a clearer picture of the users and their environment, which would act as a basis for further developing the system. The description could be used to guide the development of general features, such as in prompting the implementation of levels of system access, and in designing help and explanation facilities.

As stated earlier in this section, a further objective of the evaluation with BSC was to obtain feedback on the suitability and relevance of the actual functionality and dialogue which was being represented in the existing prototype. In general, the response to both the existing functionality and user interface dialogue was favourable. It was agreed that such a system would be useful. At the more detailed user interface level, the feedback concentrated on the use of domain specific terminology. Some terms used in the prototype to describe design objects, such as *'access profiles'*, were known by other names by the BSC members. This reflected different terminology associated with the different database systems, as used by the expert during initial knowledge engineering, and by the installation at BSC. It was agreed that, in essence, these design objects appeared to be the same. However, such a finding raised two important considerations. Firstly, there was a requirement, at the very least, for some form of on-line thesaurus for cross checking the meaning of terms and their synonyms. Secondly, and more importantly, the mismatch expressed here highlighted that there were potential differences in design practice between the expert, and the users as represented by BSC. It was necessary to be aware of such a mismatch and to investigate possible differences throughout the development of SAM.

Related to the point just described, a further issue which emerged from the evaluation of the prototype concerned the construction and display of diagrams which were simulated. It was felt that there was a need to provide some level of help which described the diagramming conventions; that is the meaning of the various symbols (such as lozenges, boxes and types of arrow) used within the diagrams to represent particular data items and data relationships.

The results from the evaluation with BSC were documented as part of the evolving specification. To complement this evaluation, the prototype was also demonstrated to various database design experts from which further evaluative feedback was gained. The findings from this work are described in the following section.

8.12. Evaluation with database design experts

Three further evaluations of the prototype were carried out, this time with BIS consultants who all had experience in database design. The same demonstration and feedback recording procedure as had been employed during the evaluation with BSC was used in these subsequent evaluations. They involved the following three sets of experts:

- The expert consultant who had been interviewed during the initial knowledge elicitation phase. A half-day meeting was held. Due to his heavy workload a visit was made by the core development team to Manchester where the consultant was currently working 'on-site', on the design and implementation of a SUPRA database.

- Another consultant who had recently joined BIS and who had considerable knowledge of database design. Again a half-day meeting was held, this time at the University at Aston.

- The *Intellipse* project manager, and a senior consultant who was involved in the project but who had not been part of the core development. Both of these people had experience of database design but were not currently working in that specific area. A two-hour evaluation meeting was held at Aston.

Once again the opinion of the prototype was favourable, particularly from the expert who had been involved at the start of the project. He suggested that such a system would have real commercial potential, and indeed if it was already fully developed would be particularly useful for members on the current project he was working on. At a detailed

level the feedback from these evaluations centred, in the main, upon omissions or misrepresentations of particular items which were identified in the prototype. These included such things as the omission of parameters which would be required during the input of data, and incorrect arrow assignments on parts of the data flow diagrams. At the same time, comments were received on certain aspects of the dialogue requirements. For example, the need for further scrolling facilities was identified in one area of the prototype. In identifying all of these elements the value of the prototype as a medium of communication in knowledge acquisition was emphasised.

What also emerged from these evaluations was a certain level of conflict between the views held on database design practice across the three groups of experts consulted. Such a finding is not uncommon when more than one domain expert is consulted (see for example, Hart 1986). In this case the differences can be attributed, to a large extent, to differences in approach resulting from varying types of practical experience. The expert who had initially been consulted had up-to-date practical experience; this was in comparison with the others who possibly held a more academic view of design as they were either involved in presenting courses on the subject, or had not worked in that particular area for some time. Although no fundamental mismatches were identified in our work, there were some differences in opinions expressed. It was important in the future development of SAM to be mindful of such differences amongst experts. This was particularly so if SAM's development was to be continued beyond *Intellipse* where it was likely to involve other experts than the one that had been consulted in the initial knowledge elicitation phase.

8.13. End of project documentation

The evaluation comments which had been obtained were dealt with in two ways. Firstly amendments were made to the prototype so that both omissions and extra requirements that had been pointed out were dealt with. This lead to the production of a version 1.1. of the HyperCard prototype. Secondly, where appropriate the comments were documented as part of the final specification. This specification was one part of the final documentation. As the *Intellipse* was about to terminate it was not possible to continue work on the next stage of development. The next stage would have primarily involved developing the knowledge bases to support the activities which had been identified by the work so far. The objective of the final project documentation was to provide a basis for any future work that would be carried out on developing SAM. The overall contents of the documentation are described below:-

1. Version 1. and 1.1. of the HyperCard prototype (in software).
2. A specification of the proposed system including a description of the functions along with their required data inputs and outputs. This was drawn up by a member from BIS according, as far was possible, to their MODUS standards documentation.
3. A set of the screen layouts from the prototype. (These were screen 'dumps' taken from HyperCard; see Appendix 9.)
4. A set of documentation describing the user interface features along with explanations of why these features had been implemented. These were documented using the 'design decision' *pro formas* which had been developed as part of the set of methods (described in Chapter 6).
5. A report on the User interface issues (prepared by the author). This outlined, with explanations, what were considered to be the basic requirements and options for implementing the SAM user interface. It drew upon what had been established so far in the work, and again was to act as part of the basis for any further work on SAM.

Items 4, and 5 will now be described in more detail, in the following two sections.

8.14. Documenting the user interface design decisions

The SAM project had progressed to an appropriate point for highlighting the value of documenting the rationale behind the various user interface features in the existing prototype. The extent and method of evaluation was also recorded. The documentation forms that had been drawn up previously were used for this purpose (see instantiated examples of these in Figure 8.6.). In documenting the features in this way it provided a way of capturing what had been learnt so far in respect of the user interface. This information could then be utilised in future. In particular, it was possible to make explicit both those features implemented as a result of the information that had to be represented at a particular point (for example, the use of forms for entering information about each data relation), and those prototype features that had been determined by the tool-specific (i.e. HyperCard) facilities provided (such as the use of screen 'buttons' to indicate options).

8.15. Report on Recommendations and Options for the user interface

As requested by the project manager, the author produced a final report which described a set of basic requirements for the user interface, that is a basic set of facilities that it was proposed would be necessary to meet the identified functionality for SAM. Table 8.1. summarises the specific dialogue and closely related functional features which were identified in the report, along with the associated options, or alternatives, that were cited.

Application name : SAM Prototype V.1.	Development tools : Hardware : Apple Macintosh Plus Software : HyperCard
Interface element : FORM-filling	**Evaluation procedure :** Demonstration as part of prototype to potential end-users and experts
Reason for choice : The nature of information, i.e. several characteristics associated with an element, and these characteristics requiring values, made a FORM desirable	
Alternatives considered : _____	**Reasons against:**
Guidelines sought : **(references)** _____	

Application name : SAM Prototype V.1.	Development tools : Hardware : Apple Macintosh Plus Software : HyperCard
Interface element : OPTION "Buttons"	**Evaluation procedure :** Demonstration as part of prototype to potential end-users and experts
Reason for choice : 1. Object-orientated nature of prototyping tool allows such a facility 2. Limitation on screen size made buttons desirable (if limited number of options)	
Alternatives considered : Pop-up/Pull-down menus which incorporate each option	**Reasons against:** Possible over-sophistication for final delivery system
Guidelines sought : **(references)** _____	

Application name : SAM Prototype V.1.	Development tools : Hardware : Apple Macintosh Plus Software : HyperCard
Interface element : SCROLL BARS on diagrams	**Evaluation procedure :** Demonstration as part of prototype to potential end-users and experts
Reason for choice : Due to size of the Logical Data model, and Physical Design diagrams, scrolling is highly desirable.	
Alternatives considered : Paging through diagram	**Reasons against:** Scale and complexity of diagrams made paging impractical and scrolling preferable
Guidelines sought : **(references)** _____	

Figure 8.6. Examples of documented user interface design decisions from the SAM prototype

Feature	*Alternatives*	*Disadvantages of alternatives*
Interface features		
Bit-mapped facilities	Character-based facilities	Speed. Inappropriate for heavy graphics requirements of SAM. Bit addressable screens would facilitate many of the options outlined below.
Mouse input	Keyboard only	Manipulation required in constructing and navigating diagrams would be difficult.
Pop-up/Pull down menus	Fixed screen menus	Slow navigation. Inapropriate given non-hierarchical nature of options identified in SAM. Also given numerous amount of options which would put a limit on screen space.
Scrolling (particularly diagrams)	Page between screens	Poor feature for viewing and understanding what are often large, complex diagrams which require checking for errors or possible changes by the user.
Zooming facility (for viewing diagrams)	Single level of diagram magnification	Inability to view whole, or detailed view of the picture.
Functional features		
Logical Data models generated diagrammatically from data relation forms	User constructs diagram by hand	It is yet to be established which method is better. It is likely to be a between what is feasible in terms of implementation and what is most suitable for the user.
Access Profiles	User describes the profile non-graphically (e.g. textually)	'Conceptualising' the profile and its characteristics is likely to be less easy. Current practice is diagram based.

Table 8.1. SAM system's user interface requirements

The report also included some more general proposals on various aspects which the author believed it was important to consider in any future development of SAM. These general proposals concerned the following issues:

- The choice of future prototyping tools - It was pointed out that although HyperCard had been extremely useful in providing a quick solution for gathering aspects of user interface requirements it was not an appropriate tool for developing either the actual knowledge base, or in delivering the final user interface.

- The importance of establishing available experts - The initial expert used was intimating that due to increased workload, his future availability would be very restricted. Given the complexity of the proposed system, for any future work on SAM to be feasible it would require one, or more experts, who could be regularly available.

- The importance of continued involvement of users for evaluation. It was pointed out that although a quite detailed picture of potential users had been built up it would be important to maintain access to a set of users (such as BSC). Their availability for purposes of evaluation was as important as the availability of the experts. It was also suggested that if future work was aimed at developing a commercial product, then in order to gain a more representative view of potential users it would be highly desirable to obtain some level of feedback on SAM from people other than just the group at BSC.

8.16. Summing up the SAM project

In total, the end of project documentation laid the foundations for future work. As already mentioned earlier in the chapter, the work that had been carried out on SAM was actually used as the background to a proposal submitted to a new government research initiative. At the present time of writing it is not known whether this proposal will be accepted. What had been particularly positive about the work carried out was the increased role and concentration upon the user interface component of the system; something which had not been existent previously within *Intellipse* (see Chapter 4). Whilst it was not possible to show the final rewards for earlier consideration of user requirements, some more immediate benefits were identifiable:

- The user interface prototype which had been developed provided a 'concrete' means for beginning the process of user requirements capture, and as such

acted as a detailed concept demonstrator. It is unlikely that such a clear picture of user requirements and their complexity would have been established so early on, if the usual approach of prototyping the knowledge base had been adopted. At the same time it was important to be aware that construction of the knowledge base would itself be no trivial matter. However, by beginning with the process of user requirements capture and being led by these requirements it was less likely to lead to effort being put into constructing a KBS which, although internally consistent, might not have been consistent with external (user) requirements.

- The evaluation conducted with BSC was very useful in highlighting issues at what was quite an early stage in development. Those issues highlighted were unlikely to have been discovered until considerably later in development if an evaluation with users had not been carried out in this way.

- The user interface prototype also had an identifiable role to play in evaluation with the domain experts. Its role as a medium of communication between developer (knowledge engineer) and expert was highlighted. In terms of SAM, the facility for such communication would have been slowed down severely if there had been a reliance upon constructing a working knowledge base first.

8.17. An appraisal of the methods for early user interface consideration

The methods for early consideration of the user interface that had been developed previously (described in Chapter 6) were instrumental in guiding the emphasis and activities summarised in the above section. The analysis of functionality in terms of their required representation at the interface had been very useful in giving structure to the process of developing the prototype. This analysis was very useful in 'managing' the design meetings which took place between the core development team. The use of this approach emphasised several points. Most importantly it demonstrated that although it is not possible to specify from the outset exactly what the user interface will look like in the eventual system, concentrating on the interface early on provides guidance for the future design iterations. More specifically, it provides a form of checklist for the developer during evaluation. Also, in this project the analysis of representation requirements, together with the formulation of a description of potential classes of user and their environment, helped in reducing the likelihood that incorrect assumptions were made about what the user requirements actually were. In effect, it reduced the potential problems which might have occurred in a more designer-centric and expert-centric approach to the system development. Consequently, the overall approach to developing

SAM ensured that the complexity of implementing the proposed system was viewed not merely in terms of technical complexity, but also identified the user issues as an equally important element in determining the success or failure of actually implementing the SAM system. This represented a real change in emphasis from earlier work in *Intellipse*, and indeed also a change from the picture of present KBS development practice in general that had emerged from the earlier research work.

8.18. Concluding remarks

As had been described in the previous two sections, several lessons had been learned during the SAM project. As a result of these lessons it had been possible to show to the development team as a whole, that there was considerable benefit in adopting the kinds of user-centred approaches that were adopted. Such a demonstration, involving the development team as a whole, was important. This was because the essence of the methods employed, whilst having a basis in what could be considered established human factors practice, were not common, or indeed obvious to the developers who had been involved in this project. Overall it had been shown that these kinds of approaches have a definite place within a KBS development methodology. The following chapter will place the SAM project together with the work described in previous chapters, and will discuss the overall findings and their implications in terms of the wider context of human factors and KBS.

Chapter Nine

Discussion

Outline

This chapter draws together the various 'strands' which have so far been described in this thesis. These strands constitute the experiences and insights gained during the course of the research. The chapter begins by restating the original objectives of the work and reviews the approach which was adopted to address these objectives. The issue of human factors integration within KBS development was central to much of the work that was carried out. This issue is first discussed in terms of the general 'inhibitory' factors, or constraints upon successful integration, which exist in the commercial and industrial environments - the particular 'subjects' for the research. Following this discussion, the research experiences are further analysed in an attempt to derive some general lessons from them. These lessons are presented in two ways. Firstly a set of statements is presented; these outline what is in essence a prescription for human factors integration on a KBS development project. Secondly, a development life-cycle for user interface design within overall KBS development is proposed. This life-cycle proposes user interface design as a parallel, and complementary, activity to knowledge base design, rather than as an activity which 'happens' subsequently. Finally the chapter concludes by addressing the question of how to support such a life-cycle. This is done both by relating the experiences from this research, and by looking at the issue from a wider perspective.

9.1. Reviewing the research approach

The original objectives for the human factors component of the Alvey *Intellipse* project had been:

- To give guidance on MMI issues which arose within *Intellipse*
- To identify, more clearly, the human factors issues in KBS development

These objectives were discussed in more detail in the first chapter (section 1.7.). The research plan which was formulated and subsequently implemented to meet these objectives has now been described in the previous chapters. From the research work that was undertaken there emerged a detailed view of current KBS development practice and the role that human factors has within that development. The chapter will draw together the important issues that were identified during the course of the research and will present a picture, both of the 'present' and 'future', of human factors within KBS development. Before presenting this picture, it is important to review some of the important background elements to the research work. These elements, of which the central ones will be discussed below, each had some considerable effect upon the formulation and execution of the plan for research, and upon the focus and emphasis of the resultant findings.

The overall essence of the original objectives, and the subsequent work, was that both specific problems and general issues were to be addressed; specific problems such as the design of the user interface for modules in *Intellipse*, and general issues such as the role of human factors in KBS development. The same approach was also adopted in the other main case study which was carried out with the TSB, this project taking place outside the *Intellipse* project (cf. Chapter Seven). The TSB project had not been envisaged at the outset of the work, but helped to provide, along with the interviews conducted with other commercial and industrial developers (see Chapter Five), a further source for developing a representative view of KBS practice. In the research work reported in this thesis there was considerable emphasis throughout the work upon general and specific issues. This is highlighted in the initial objectives as stated above, and is also highlighted, very much, in the actual research approach which was adopted. A good example of the relationship between general and specific considerations in the research is shown in the development and use of the methods devised for user interface design (see Chapter Six). These methods were developed with two main objectives in mind. Firstly, they would provide the initial basis for actual support for the KBS developers in the current and future projects being undertaken. Secondly, they would provide a research vehicle for identifying the general issues pertaining to the uptake of human factors orientated approaches in the setting of commercial and industrial KBS development. These two objectives were related to one another, in that by identifying general issues it was possible to get feedback on the specific methods themselves as a precursor to indicating the form of future human factors support (i.e. better developed methods) for KBS development.

A second essential characteristic of the research concerns the actual setting, or environment in which the overall studies took place. The environments that the various KBS developments focussed upon in this work were primarily commercial, or commercially-orientated ones. As a result, various factors were at play which detracted from achieving the human factors 'ideal' in system development; factors such as the limited, largely non-existent, human factors base within a development team (in terms of either qualified personnel or general awareness of human factors issues); the commercially-based choice of system implementation tools; and the limited involvement of end-user. Each of these factors had an effect in determining the role of human factors in 'real-life' KBS development. This 'real-life' development is distinguishable from purely academic or research environments where business factors are not so overriding. If the setting had been that of the latter, 'laboratory-like' environment, then this would have undoubtedly led to a research emphasis (in terms of approach, and the focus and content of the research findings) which differed significantly from the one which

emerged. To illustrate this point, an important factor which was established during the research work carried out concerned the need to capture end-user requirements for a proposed KBS (see, in particular, Chapter Four). This was identified early on as an area for human factors support, and the subsequent research work concentrated upon this area. If however, the research had not been so concerned with development in the commercial setting, it is likely that in addressing the second of the initial project objectives, the research work and its findings would have focussed upon more 'theoretical' areas of KBS, such as knowledge elicitation. While human factors has a significant contribution to make in those theoretical areas of KBS, such a context for research did not exist in the *Intellipse* project which provided the impetus for all of the work carried out. Here it was important to serve the direct needs of the *Intellipse* project whilst, where possible, extracting more general issues.

A further important feature of the research concerns the emphasis upon "designer-centred" research. A philosophy behind the case study and interview approach was that it was aimed at assessing, and responding to, the (user interface) design support requirements of the designers *in vivo*. The need for such an approach was identified in part from a review of the literature on human factors and its practice, in both general interactive system design, and KBS design (see Chapters Two and Three). The need was further identified during the preliminary *Advisor* case study (see Chapter Four). The review of the literature indicated that there was a considerable gap between the level of human factors application in system design as proposed by human factors specialists, and the actual level of application in 'real-world' design. The *Advisor* study again highlighted, this time at first hand, the existence of this problematic gap. Together the two sources lead to the formulation of the subsequent research approach that was adopted. There is a growing awareness of the need for "designer-centred" human factors research, such research being seen not as excluding, but rather as complementary to, more 'basic', or 'pure' research. The awareness stems from an increasing acknowledgement that the successful application of human factors techniques will only be achieved fully when there is also an appreciation of the practical design environments to which these techniques are targeted. As was previously cited in the first chapter (section 1.9.), Hammond et al. (1983) stated that:-

> "...there is a need...to consider the interface between researcher and practitioner as well as that between system and user."

Further agreement for this perspective comes from the 'practice what you preach' principle being employed in the ESPRIT HUFIT project, mentioned previously in Chapter Two (section 2.11) and Chapter Six (section 6.3.). The user interface design

support tools being developed in that project are being specified, and it is proposed will be evaluated, through the involvement of the IT product designers - the potential end-users of the support tools (Galer and Russell 1987; Russell and Galer 1987).

The research which has been reported in this thesis had, as its basis, similar principles to those just reported from other areas within human factors. As has already been stated the research was conducted within an action research framework, implemented primarily through active involvement on 'live' KBS development projects which acted as case studies. There have been further, more recent calls, for the action research approach within the human factors community. In a report by Klein and Newman (1987), which formed part of the Alvey Human Interface Club's response to a proposed follow-up initiative which was to succeed Alvey, it was suggested that there was a role within future human interface work for:-

> "..tracer studies within an action research framework...where the *'accompanying researcher'* can pick-up content-related issues and feed in the results of research" (Klein and Newman 1987; p 6).

The research approach that was adopted in this work is linked closely with the call which came subsequently from Klein and Newman. Various factors can be seen as having determined the approach in this work. As was stated earlier the literature review and preliminary *Advisor* study had considerable influence in determining the research plan. Of equal importance was the collaborative nature of the work, involving both industrial and academic partners. Such collaboration leant itself to an action research framework, as it was possible to obtain a level of active involvement on live projects, where the researcher would both initiate and respond to user interface design activities throughout a project development. Whilst the case study conducted with the TSB, and the study begun with British Coal (see Chapter Six; section 6.9.), were not in the original proposal, once set up they were important and useful studies, providing a more representative picture of KBS development practice than would have been gained from work carried out within *Intellipse* alone.

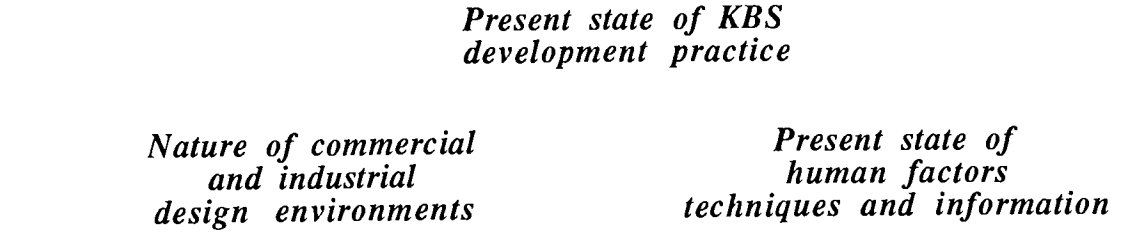
Having reviewed the research approach adopted in this work the remainder of the discussion will concentrate upon the actual research findings that emerged. As was mentioned earlier in this section, the research addressed both general and specific issues pertaining to user interface design within KBS development. In keeping with this distinction between general and specific issues the two levels will be dealt with individually. Firstly, the more general issues will be discussed. This discussion will draw from all of the research sources involved, that is the literature, the interviews, and

the case studies, in order to identify issues concerning the integration of human factors within KBS development. The essence of this general discussion is to highlight what are seen as the factors which are obstructing the practical integration of human factors within KBS development. An appreciation of these inhibitory factors is important in that they have implications for the successful integration of any human factors methods and tools. The second part of the following discussion centres on more specific issues, drawing largely upon the direct experiences gained from the case studies. What was learned from these studies is then presented in the form of a model for user interface design activity within the KBS development life-cycle.

9.2. Human factors integration in KBS development

At various stages throughout the course of the research numerous factors were identified which could be seen as having some influence upon the extent to which information and techniques from human factors were, and indeed will be, incorporated in the development of KBS. These influencing factors can broadly be classified under three headings, these are represented in Figure 9.1., below.

Figure 9.1. Three main groups of factors influencing uptake of human factors in KBS development



Many of the issues pertaining to each of the three headings listed in Figure 9.1. have been reported already during the course of the thesis. As they were identified at the various research stages (during the literature review, the interviews and the case studies) so they were discussed. In Chapter Two, which reviewed human factors within interactive systems design, issues were discussed which were seen as contributing to poor human factors integration in interactive system design in general. To recap on these issues, Figure 2.2. from that chapter is presented again in Figure 9.2.

The factors identified in Chapter Two and represented in Figure 9.2. can be seen as being applicable to the state of interactive computer system design in general. The literature review reported in Chapter Three, and the interviews held with KBS

developers which were reported in Chapter Five, identified further inhibitory factors which pertained to human factors integration in KBS in particular. Once again, to recap on these KBS specific factors which were identified at that time, Figure 9.3. is taken from Chapter Five.

Figure 9.2. Determinants of poor human factors integration in interactive system design (From Chapter Two; section 2.10)

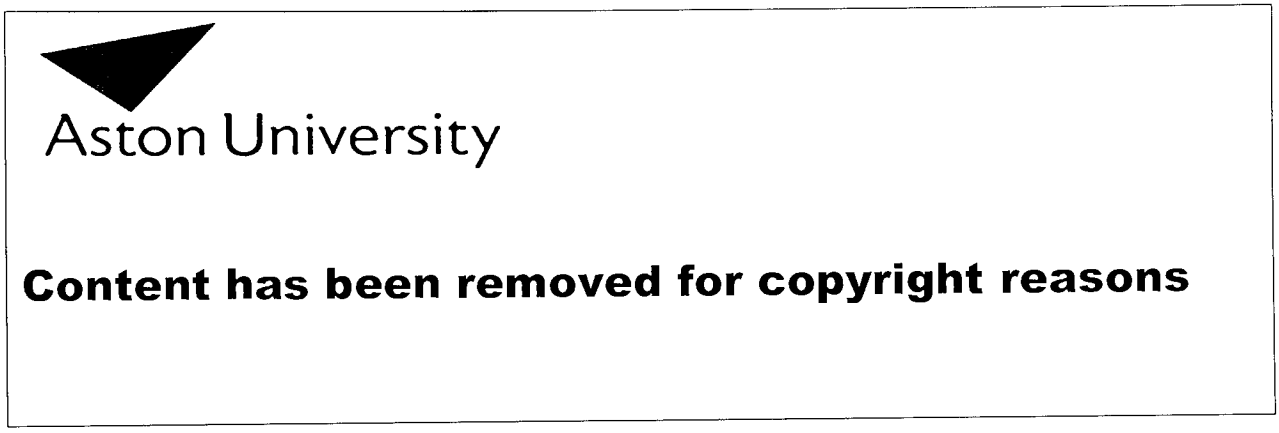


Figure 9.3. Features of KBS design practice contributing to poor user interface consideration (From Chapter Five; section 5.4.)

9.3. Focusing on factors within present KBS development

As discussed and represented in the previous section, several factors can be seen as contributing to the approach taken towards human factors, and within that user interface design, in present KBS development. The two case studies that followed on from the earlier work provided a clearer picture of the existing problems in this area. Drawing upon the overall research experience, this section will single out four, largely interdependent, characteristics of present KBS development which can be seen as significant 'hurdles' which stand in the way of improved human factors practice. These characteristics should not be seen as additional to, or exclusive of, the issues already raised. Many of the points which are about to be made have been implied earlier at some level, and so these characteristics should be viewed as encompassing what are central problematic features in the context of KBS.

The characteristics which are to be discussed in detail all reflect, to varying degrees, an 'immaturity' in the existing KBS technology; immaturity not only in terms of the present state of the methodologies and tools which are used to implement the systems, but also in terms of the objectives underlying the attempt to apply the technology in the first place. The use of the word 'immature' in respect to KBS technology implies that at some point changes will occur in the technology - that is it will become better developed, and hence more mature. In consideration of this, the following discussion will not only describe the present state of KBS development and its effects upon existing user interface design practice, but will also discuss the possible consequences of further maturation upon that design activity.

The essence of the four characteristics which are to be discussed in detail, can best be introduced in the following statements:-

- *Those people involved in commercial and industrial KBS development do not automatically place human factors and user interface design as high priorities alongside other important, 'competing' design activities.*
- *KBS work has largely been concerned with, or has only reached the stage of, producing 'concept demonstrators' and prototypes rather than 'working' systems.*
- *KBS work has been 'solution-driven' rather than 'problem-driven'.*
- *KBS and user interface design both rely heavily upon iterative development. This makes control of the development process difficult.*

These will now be elaborated upon in turn in the following four sections.

9.4. User interface design - an obvious priority?

Throughout the thesis there have been references made, either implicitly or explicitly, to the general lack of awareness of human factors amongst those people who are carrying out KBS development. These system developers, along with computer system developers generally, can be thought of as the *"owners of the design process"* (Underwood 1987), as it is these people who implement and manage the overall approach to design, and the development process.

Whilst those in human factors propose various user-centred approaches it is not these same people who are commonly involved in, or can influence significantly, the overall development process. It was highlighted during the interviews that the technical background of most involved in KBS development was not one of human factors, and that human factors expertise and guidance was rarely, if ever, called in to a project, at least until much later on in development (cf. Chapter Five; section 5.3., Table 5.6.). It was reported that in both the cases of the TSB, and BIS, the perspective was one of KBS being an extension of, and not significantly distinct from, traditional DP system development. An implication of this perspective is that previous design practice, will be carried over, at least in the first instance, as the basis for the KBS development approach. This is problematic in that it obscures the view held on the importance of user interface design. The traditional systems, around which previous design practice evolved, do not generally require the levels of user-system interaction, or the kind of user consideration, that KBS potentially do (cf. Chapter Eight, section 8.3.).

Evidence of where user interface design is obscured at present, manifests itself in several ways. Most significant is probably the tendency for formative evaluation with just experts rather than potential end-users. This was identified as a problem throughout the research; in the literature (cf. Chapter Three; section 3.11.), the interviews (Chapter Five; section 5.3., Table 5.3), and early on in the IBS case study (Chapter Seven; section 7.10.). This feature of present development practice stems from a potentially misplaced concern to build a substantial part of the knowledge-base for a proposed application, prior to establishing what the user's functional and dialogue requirements are in that proposed application domain.

A further example of where the importance and complexity of user interface design can be obscured is in the process whereby a tool (expert system shell) is chosen in order to gain familiarity with the KBS technology. As it was pointed out in Chapter Three (section 5.3.), whilst this is an apparently legitimate rationale in the very short-term, the preoccupation with tools in this way, especially given their existing limitations, can

divert the developers from becoming familiar with all aspects and problems in KBS development; problems which include user requirements capture and interface design.

The situation just described will only change when an awareness of the importance of user interface design is engendered from within, by the KBS developers themselves, as well as from outside from human factors specialists. This is likely to happen when work on KBS is conducted in a less speculative way and where it is no longer commercially justifiable to merely explore the technology under the auspices of speculative research. This transition has already begun, and as more experiences accrue of development problems, or failures, due to inadequate consideration of the user interface, so there should be an increased receptiveness to, and awareness of, human factors. Evidence for this mounting awareness does not simply come from the anecdotal reports of development experiences (cf.Chapter Three; section 3.9.), but is also shown in the increasingly sophisticated user interface facilities, alongside more developed knowledge processing facilities, which are being built into KBS development tools (i.e. the shells and development environments). However, as has been stressed throughout this thesis, user interface design should not just include the physical or dialogue elements of the system but should also covers users' task and functional requirements. Inadequate consideration of such fundamental requirements at the outset of a project cannot be compensated for later, by the design of a more sophisticated physical interface (see for example, Chapter Four; section 4.7.). As a result, improvements in the approach to user interface design must be integrated within the overall process of KBS development rather than as a (belated) appendage to one particular stage in that development.

9.5. The predominance of prototypes

A second feature of the current state of KBS development concerns the predominance of prototype, rather than complete systems. This feature is closely linked to the issues raised in the previous section, as a consequence of both sets of factors is that the importance of human factors in KBS can only be discussed, to a significant extent, in terms of potential importance rather than well-established, and agreed, importance. Indeed there is a analogous emphasis on 'potential' rather than 'established', in terms of KBS technology as a whole. The wider issue of 'unproven' KBS potential will not be discussed here however, as it is the aim to concentrate on the issue of human factors. What is important here, as it has been throughout the thesis, is the role of human factors, and user interface design, as it is being addressed by those who are involved in the attempts at the practical implementation of KBS.

As has already been shown, much of the KBS work up to now has produced either concept demonstrators, or more developed prototypes, rather than delivered working systems that are in regular use. Such a trend was reported in the literature review in Chapter Three (cf. section 3.2.) and this was further substantiated by the reports during the interviews held with KBS developers (cf. Chapter Five; section 5.3.). The projects that were the subject of the case studies in this thesis were also concerned with the creation of prototype systems, which were seen as stepping stones towards the delivery of complete systems.

So what effect does all of this have upon user interface design practice. In fact there are a number of possible effects, all of which can be seen to manifest themselves in the present paucity of early user requirements capture and user-centred evaluation. The most significant effect, however, and one which can be seen as influencing those user-centred activities just referred to, concerns possible risky assumptions made by developers about the validity of prototypes as indicators for future system feasibility and acceptability. Associated with the existence of prototype systems there is a danger in underestimating the importance and influence upon eventual system success that early user interface consideration can have. At the same time the activity of user interface design can be 'forgotten' amidst the concern to create some form of working prototype. There is a danger in assuming a smooth transition in system development, whereby it is considered that a working system can be delivered as a simple extension of an existing prototype. That is the system can be delivered, incrementally, by iterating through fundamentally the same development phases which were followed in order to create a prototype. Such iteration might require some additional phases (such as user interface design), but basically the system 'specification' has been set quite rigidly once a prototype has been built. It can be seen that such a view can maintain user interface design as a relatively low priority, if it was not considered in detail during the construction of the original prototype. If user requirements are not addressed during the construction of the prototype, then it will not be until much later in development that such requirements are addressed, when problems might occur in terms of system acceptability by the users. Such a situation was seen in the *Advisor* study where there was a situation of 'a system in search of a user' (cf. Chapter Four; section 4.8.); there was a similar potential problem in the development approach preferred by the project sponsor that was described in the IBS case study (cf Chapter Seven; section 7.8.).

As long as it is seen commercially viable to go only as far as producing prototypes then it is likely that the importance of user interface design will continued to be obscured for the reasons just described. If prototypes can be 'written-off' as exploratory research then

real problems of user acceptability might never be encountered. However, as was similarly discussed in the previous section, if KBS are to grow as a proven technology then there will have to come a point where prototypes can no longer be justified in this way. With the advent of such maturation, the assumptions concerning the relatively straightforward, 'evolutionary', nature of existing prototyping will be tested. An outcome of this will be a greater awareness of user interface design as an important activity in the overall development process.

9.6. 'Solution-driven' KBS development

It was discussed in the previous section how KBS is seen very much in terms of having great potential as a technology, and that as yet the limit of such potential has far from been reached. A consequence of this is that much of the present KBS work has been 'solution driven' rather than 'problem driven', with KBS technology playing the role of the solution whilst problems (or domain applications) are sought for it to be applied to. This characterisation of present KBS work has been captured by others, for example, Bader (1988):-

> "Many KBS are the result of a process which can be characterised as a *specific technological solution looking around for a problem to solve.* " (Bader 1988; p 191)

Rodger and Edwards (1989) suggest a 'spectrum' of current approaches to the application of KBS which has *exploratory research* at one end of this spectrum, the *solution-driven* approach in the middle, and the *problem-driven* at the other end:-

> "As exploratory developments, their (*KBS*) applicability occurs through anticipated "spin-offs". The middle ground consists of overt searching for expert system applications; the solution-driven approach. The other extreme is the use of expert systems to tackle real-world problems; the problem-driven approach."

It could be argued that there is an element of *post hoc* rationalisation in this classification, particularly in respect to the apparent structured rationale behind the present application of KBS (connoted for instance by the phrase 'anticipated spin-offs'). However, it is a useful division of present KBS work and methodology. The authors go on to state that much of the present work, having moved from the exploratory stage, now lies within the solution-driven band, but that future growth of the technology will rely upon a more problem-driven approach. Rather than searching for a problem as is the case with the solution-driven path, the problem-driven approach would begin with the identification of a problem, and having established the appropriacy of automated

support, would then assess the extent to which that support most suitably involved the use of KBS, more conventional systems, or a mixture of both.

Examples of the solution-driven approach were also found during the course of the research work in this thesis. It was reported on several occasions during the interviews that a role of prototyping was to demonstrate the basic concepts behind KBS to management. This was as a precursor to identifying with management (the potential 'clients') any possible application domains which might exist in their department, or organisation. Much of the early work of the KBS team at the TSB, for example, had involved seminar presentations to business managers, and indeed this is how the teams' involvement in the IBS project had subsequently come about.

Once again the important question in all of this is - 'What is the effect upon the integration of human factors and user interface design activity?'. Overall, it means that along with an identification of potential applications, users also have to be sought. It is not the case in the more solution-driven state that users present themselves to the developers requesting a KBS solution to a problem. As a result users are not well-defined, and a process requiring such definition is needed. However, as has been seen during this research and for some of the reasons discussed recently in this chapter, identifying and defining end-users is often neglected when placed amongst other design activities.

A related problem exists in terms of the developer-client relationship where the clients are largely management. Whilst managers are the starting point for choosing and initiating a project these people are not always the actual end-users (this situation is exemplified in the IBS project) and so are likely to be somewhat removed from the particular chosen application domain, and hence from the actual tasks involved in that domain. This puts a heavy requirement on end-user involvement during specification and evaluation, but once again this involvement doesn't materialise. The likelihood of end-user involvement is further reduced in this problem-searching scenario by the fact that managers (those funding the project) are being asked to provide, or provide for, both experts' time and end-users' time in a KBS development. Such provision is obviously costly, and is being requested against the backcloth of an unproven technology. This inevitably increases the perception of KBS as a risky investment, and this in turn can be seen as a further reason for why end-users are not involved in development as much as is prescribed by human factors specialists.

If there is a transition towards problem-driven KBS development, as it has been argued here there must be in order for the technology to mature, then the situation which exists in terms of levels of user involvement will also change. In a problem-driven situation end-users should be more clearly defined, and the process which exists at present, whereby users must be explicitly sought by developers, should be reduced. Involving and making active reference to users should be more justifiable (in terms of cost incurred) when it is an agreed problem which initiates a KBS development, rather than merely the KBS technology itself.

9.7. Consequences of iteration in KBS development and user interface design

The final, general, feature of KBS development to be discussed in this part of the chapter concerns the level of iteration which exists both in the process of KBS construction and in user interface design. This makes management and control of a project difficult when compared with conventional life-cycles which rely upon discrete, phased development. It could be argued that unlike the previous features discussed (which relate more to the present state of KBS project management) iteration is a feature which is bound up with the internal nature of KBS themselves, and as such will not change in the manner predicted for the other features. Whilst this is to a large extent true, it is proposed that iteration is an important aspect which, as will be argued in this section, will be open to some change as a result of changes in project management practice.

It was shown in Chapter Two (cf. sections 2.5., 2.6.) that user interface design is itself an iterative, evolutionary process, involving a certain amount of specification followed by prototyping, evaluation, and then amendments to the earlier specification. In the description of fundamental stages in KBS development (described in Chapter Three) it was shown that iteration was also a prevalent feature which occurred, in the main, between the stages of knowledge acquisition and prototype construction. This means that in total, there exists a considerable amount of iteration required to produce a KBS, both in terms of knowledge base construction and user interface construction. Iteration makes the task of project management, or control, a difficult one. In particular, it is hard to estimate the time required for a particular development (the number of iterations are unknown at the outset) and it is not easy to generate project documentation in a discrete form; that is a form which details phases of work that can be referred to, and validated.

In Chapter Two (section 2.5.) reference was made to Gould (1987) who identified three requirements for iterative design for usability; in summary these were *identification* of

required changes, followed by *ability* and *willingness* to make those changes. The last of those requirements, the willingness to make changes, is particularly important here. With the amount of iteration that is associated with the creation of the knowledge base, the requirements for iteration in the construction of the user interface just adds to the complexity from the perspective of project management. The result of this is that the user interface activity is not conducted in the way that human factors specialists would prescribe, but is dealt with much later in KBS development as an appendage, or afterthought, to the knowledge base. On the subject of the present approach to rapid prototyping in KBS development Klein and Newman (1987) state that:-

> "...the user interface is normally ignored until the last design iteration, or after it." (Klein and Newman 1987; p 7)

Many of the issues already raised in previous sections (such as lack of awareness, and user interface design activity as a low priority) come into play alongside this issue of iteration, to create the problem that is reported in the above extract.

The extent to which the problem of iteration will change in the future is not easy to predict. As has already been suggested, the nature of KBS development is that it is iterative. However, it is likely that there will be a reduction in the present level of iteration as there is a transition away from the more *ad hoc* approach to a more structured development approach. This transition will occur as there is an increased emphasis within the business sector upon better project estimation and overall project management. This transition has already begun with the emergence of proposals for more structured, phased development approaches, in the form of KBS methodologies. For example, both Hayward (1987) and Bader et al. (1988) propose a more structured, less iterative approach to development, rejecting the present *ad hoc* approaches which they say are commercially unviable. A new project, called GEMINI, recently launched by the UK government's Central Computer and Communications Agency (CCTA), is an industrial collaborative initiative with the objective of laying *"the foundations for a publicly available expert system development methodology to complement and integrate with SSADM."* (Montgomery 1988). As is indicated, such a methodology is seen as being part of an existing structured methodology (SSADM) for the development of conventional computer systems. A similar view of a need for integration, in order to provide a development framework for KBS, was seen in the approach being adopted by BIS in the SAM case study (cf. Chapter Eight; section 8.2.).

The move towards a better development framework should influence the approach taken to user interface design, if there is sufficient appreciation and consideration of the

importance of that design activity. It has been demonstrated how difficult it is to try and integrate human factors principles in a *post hoc* way to an already existing methodology (Damodoran 1988). This stresses the need for human factors considerations to be integrated from the outset of any new methodology like those being proposed for KBS. Klein and Newman (1987) in the same report quoted earlier in this section, emphasise the requirement for a KBS methodology which suitably addresses the user interface design activity:-

> "IKBS design should incorporate user-interface design in a joint prototyping process. This may require extra effort to work through possible differences within the (*design*) team." (Klein and Newman 1987; p 8)

This once again points out the need for a complementary development life-cycle, that is one which addresses the creation of both knowledge base and user interface through prototyping, rather than having the user interface added on following evolution of the knowledge base.

The notion of such a life-cycle is further developed in the following sections in this chapter. Having considered some of the major general findings of the research work the next sections will deal with more specific lessons which were learned during the research. These are primarily lessons which were learned through the case studies. In the following sections the lessons are brought together to propose a basis for a development approach that addresses some of the problems identified during the research.

9.8. Comparing the experiences of applying the user interface design methods in the IBS and SAM studies

The two case studies had provided an opportunity for applying the methods that had been developed during the middle phase of the research (cf Chapter Six). The effectiveness of the methods in each study has already been discussed independently of one another (in Chapter Seven, on the IBS project; and in Chapter Eight, on the SAM project). It is worth assessing, however, the comparative effectiveness of these methods in the two studies, as such a comparison brings to light some further interesting issues.

In the IBS study it was recorded how, although the principles behind the methods had been used to control the process of 'brainstorming' during design meetings (cf. Chapter Seven; section 7.9.), it had been difficult to apply the methods directly during the course of the project. This was seen as being as a result of several features of the development set-up (cf. Chapter Seven; section 7.14.). In particular, the absence of any detailed

specification for the system, and the project sponsor's reluctance to consult potential end-users of the system, were identified as two central constraint features.

The SAM project however, was different. Here, the methods provided a more identifiable basis of support in the development of the prototype system (cf. Chapter Eight; sections 8.16, 8.17.). The user interface prototype, which emerged from analysing the early specification in terms of both its potential functional (task), and dialogue requirements, became the central focus of attention during the project development. It provided a means for evaluation with both experts (in database design) and potential end-users (BSC). As was noted at the time, the approach to development was markedly different to previous work carried out within the overall *Intellipse* project, and enabled the team to form a much better picture of users and their requirements of SAM than had been the case in past developments. This was shown in the set of requirements and recommendations for future work on SAM that were produced at the end of the project (cf. Chapter Eight; section 8.15).

There were undoubtedly several factors at play which resulted in the differences in the two case studies as just described. One factor in the two projects that can be seen to have had greatest significance, concerns differences in the level of control that the author had in the two studies. To begin with, the project set-up was markedly different across the two studies. For instance the IBS project involved a specific 'client' (the project sponsor) who had tight control over the development approach taken, particularly in respect to access to end-users. In the SAM project, although overall project control was with the project manager from BIS, such control was not as dominant as in the relationship with the sponsor at TSB. At the same time the collaborative set-up in the Alvey *Intellipse* project meant that the author was able to gain more active involvement, and indeed control, during SAM than in the IBS project. In that later project the situation was one where second to the project sponsor, the TSB members of the team in Manchester were those who held most project control. In the IBS project the author's role in the project, despite involving a considerable amount of involvement in terms of giving advice and guidance, was more one of a monitoring role. This is in comparison with the SAM project where the author was able to exercise considerable, immediate, project control, in the sense that he was able to influence the overall development approach taken. This influence was shown in the decision to prototype the user interface first, rather than the knowledge base.

It is worth mentioning two further factors which supported the development approach taken in SAM and which enabled the methods to be applied more effectively. These were:

- The use of HyperCard as a rapid prototyping tool

and

- The existence of BSC as a collaborator whose main project role was as end-users in evaluation

The choice of HyperCard as a prototyping tool was tied up with the decision to prototype the user interface first, as discussed in the previous paragraph. It was highly improbable that this tool would have been chosen if the author had not been able to influence that initial high-level decision. In effect HyperCard provided support for the approach taken in that it allowed the team to construct a prototype which demonstrated the elements of both functionality, and dialogue, quickly in order that evaluation could take place and feedback could be obtained. General 'teething' problems experienced with the use of KEE in the IBS project (cf. Chapter Seven; section 7.11.), combined with the way that the prototype was being developed - it attempted to cover aspects of both the user interface and the knowledge base (cf. Chapter Seven; section 7.14.) - meant that progress was not as rapid as in SAM.

The second feature referred to above, concerns the existence of an end-user base. The presence of BSC, collaborators throughout the *Intellipse* project and in SAM, meant that there was an identifiable set of potential end-users from whom a description of the user environment could be gleaned, and with whom evaluation of the prototype could be conducted. Such a desirable situation did not exist within the environment and development approach which existed in the IBS project, and as a result such detailed evaluation could not be carried out in that project.

9.9. Deriving some general lessons from the research experience

The following two sections attempt to derive some general lessons from the experiences just described. The first section will present some of these lessons as a set of guidelines, or a prescription, for the integration and role of human factors (as related to the specific task of user interface design, within a KBS development project. Successful integration and human factors consideration will not simply emerge 'from thin air'. On the contrary, the effective integration of human factors will depend upon the administration and management of human factors issues during a project. This may be carried out by a project manger, or as is more likely, by a project team member who is a human factors specialist, or has human factors 'sensitivities'. It is important to reiterate that, as was

found throughout the research, in the commercial and industrial environments it cannot be expected that human factors specialists will always be at hand in a project. This may be due either, to the scarcity, or more commonly, non-existence of such people within an organisation; or to a project management 'philosophy' which does not regard them as crucial in a KBS project. It is in appreciation of this, that the notion of a 'member with human factors sensitivities' is introduced. The second section will present lessons from the research in the form of a user interface design development life-cycle which runs in parallel with other KBS development activities (mainly the construction of the knowledge base). Although the life-cycle to be presented draws significantly upon the experiences of applying user interface design methods within the SAM project, it also takes into consideration other issues which were raised throughout the research.

9.10. Guidelines for human factors integration in a KBS development project

The following statements, or postulates, and their accompanying rationale, outline some important characteristics for the effective integration of human factors. These statements are related to the specific experience gained during the course of the research. To reiterate upon a point made in the previous section, these statements refer to a member, or set of members, who has responsibility for the user interface design activity within a project. For convenience these team members will be identified under the generic term of 'user interface designer', or simply 'designer'.

The user interface designer is involved in development from the outset of a project

Such involvement is important for two major reasons:-

1. To enable the designer to lay down an overall approach to user interface design that runs alongside the other design activities.
   (The basis for such an approach will be described in detail in the following section)
2. To allow the designer to become familiar, at the same time as the other team members, with a) the proposed objectives of the project, and b) the application domain (this is discussed further, below).

The user interface designer has a certain level of knowledge about the application domain which the system is addressing

This is related to points made in the previous statement in that belated involvement in a project will mean that the designer will miss out on the inevitable learning process which other members of the team will have obtained through earlier involvement. This puts the designer at a disadvantage in that it is important

for him, or her, to have some knowledge of the application domain. This does not however mean that they have to be an expert within that domain. Knowledge of the application domain is useful as it enables the designer to have greater control on the evaluation process by being able to elicit feedback, much more effectively, from those evaluating the system. In essence it predisposes the designer to ask more 'intelligent' questions during evaluation, thereby increasing the likelihood of identifying changes that are required in a prototype system.

The user interface designer has a certain level of knowledge about the development tools being used

Although it is often seen that the 'logical' specification for a system should be divorced from considerations of its 'physical' implementation such a situation is hard to create in reality. This is particularly so in KBS development where specification is so bound up with prototyping which involves the use of some form of ('physical') software tool. As a result it is important, from a pragmatic viewpoint, for the user interface designer to have some knowledge of the facilities and limitations of the prototyping tool to be used. Such knowledge will obviously be quite extensive if the designer will themself be carrying out some coding of the prototype (as was the case in the SAM study). However, even if the designer is only specifying or making recommendations for the user interface, a knowledge of the facilities and limitations of the tools is important. In the IBS study, for example, a knowledge of the KEE tool was extremely useful during design meetings with the TSB team. As is the case in developing particular domain knowledge, familiarity with the tools being used enables greater communication between user interface designer and other team members. In the IBS study just mentioned, such familiarity enabled the author to communicate his ideas about possible implementation features. It also predisposed him to respond to, and appreciate the rationale behind, features being proposed by other team members.

Another consequence of the user interface designer possessing knowledge concerning both domain, and development tool, is an increase in credibility amongst the development team. If the designer is seen to be able to communicate with fellow team members, as well as experts and end-users, then their contribution will be viewed more seriously, and be seen as less tangential, by others involved in the project. Such knowledge, however, depends upon firstly a willingness on the part of the user interface designer to appreciate these other factors, and secondly the ability to be able to build up that knowledge. The ability to build up such knowledge is often hampered by the

situation which occurs commonly in a project, where a user interface designer is called in at a late stage in a project.

9.11. A user interface design life-cycle within KBS development

A development life-cycle for KBS is shown in Figure 9.4., overleaf. What is proposed in that life-cycle is an incorporation of user-interface design as a parallel activity in overall KBS development. Such a proposal addresses, quite neatly, the requirement for the "joint prototyping process" called for by Klein and Newman (cf. section 9.7.). The life-cycle draws directly upon the experiences gained from involvement on KBS development projects, as carried out through the case studies in this research. Overall experience was enhanced significantly by the fact that early involvement was secured in two projects. Consequently, it is now possible to say something about the position of user interface design as a part of the overall development process, rather than as an activity which sits outside other stages in development.

The SAM project (cf. Chapter Eight) in particular, showed how it is possible to conduct user interface design in parallel with other development activities. This was facilitated, in the main, by the construction of an initial user interface prototype which it was possible to use in two ways:-

1. As an early demonstrator system for evaluating functional requirements with:-
   a) Expert(s) - acting as a vehicle for evaluating information gained from initial knowledge elicitation. This acts as a first, 'communication', stage between developer and expert(s) in the iterative development of the knowledge base

   b) Users - acting as a vehicle for capturing user's functional requirements; gaining feedback on the potential utility of the proposed system

2. As an early demonstrator for evaluating the user-system dialogue
   - to assess (at a high-level) the interface requirements, such as dialogue style(s) and graphics requirements

Recent work reported by Candy and Edmonds (1988), and Buckley et al. (1988), has also been conducted using a more user-centred approach during KBS development which utilises a user interface prototype in a similar way to that used during the SAM project. In a project involving all of the above authors - a project to develop a KBS for an office environment - it was reported that a prototype was constructed for the following evaluative purposes:-

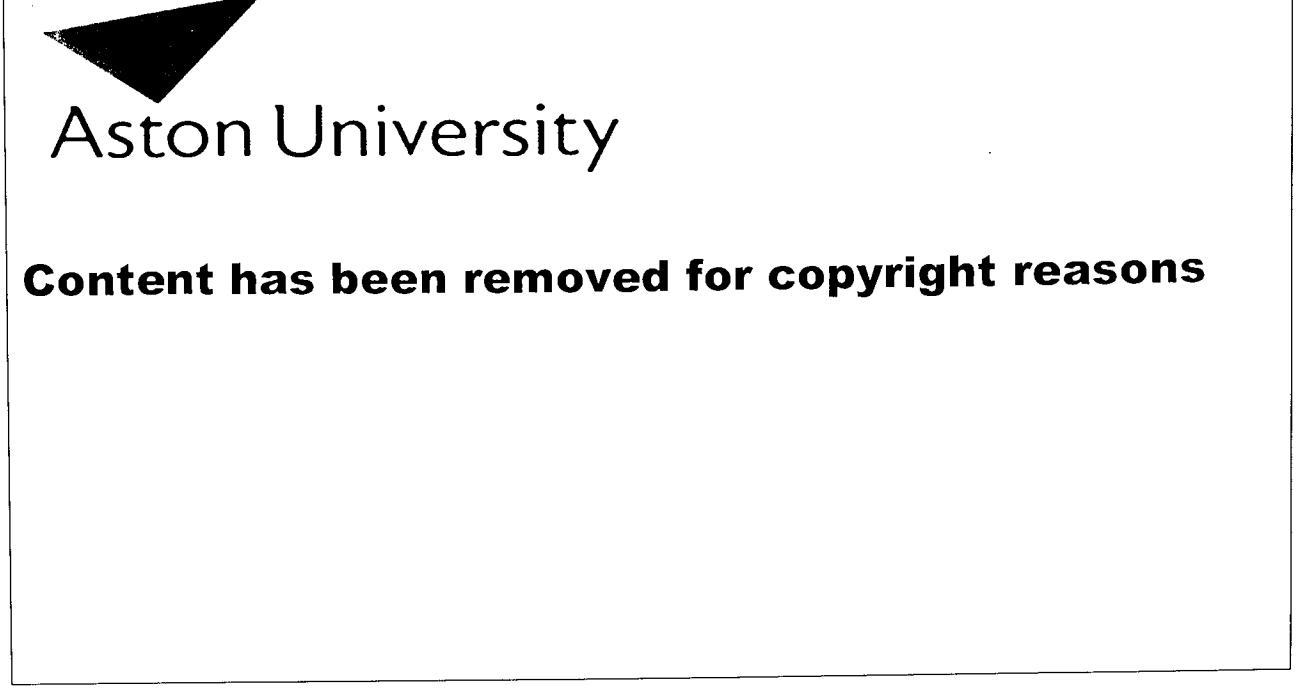


Figure 9.4. Illustration of a parallel user interface design life-cycle for KBS development

" i) To evaluate the prototype system in terms of its suitability for the task on hand from the user perspective.
ii) To provide feedback on the design and operation of the system or the next stage of design and implementation work." (Candy and Edmonds 1988)

These objectives lend corroboration to the similar dual-purpose aims underlying the SAM prototype development, described in the previous paragraph. The life-cycle described in Figure 9.4. (previous page) further elaborates on these objectives in terms of the overall process of evaluation and development process in KBS.Concentrating primarily upon the user interface design activity the life-cycle shown in Figure 9.4. will now be described, stage by stage, in more depth:-

---

Identify Domain and Specify Objectives
This refers to the identification of a domain which is appropriate to KBS technology, and to the specification of objectives for the project. Primarily, project objectives concern some improvement in an existing system (manual or automated) to which the KBS is to be applied. However, objectives might also be couched in terms of the attainment of some level of development; i.e. the production of either a concept demonstrator, working prototype, or full working system.

Identify Users
In parallel with the identification of one or more experts, or other knowledge sources (such as manuals), so end-users for the proposed application should be identified. Once identified such users can be used as a 'reference point' for future user interface design work, that is for compiling a description of the user environment (see below), and for subsequent prototype evaluations.

Describe User Environment
A description of the environment of the users (identified above) is required. This description provides necessary information for producing the initial user interface prototype (termed "Concept / Functions Demonstrator" in the life-cycle) and its subsequent, more detailed, versions. The information helps to map the proposed functionality, as identified in the project objectives and the 'first-pass' knowledge elicitation with the domain expert(s), on to the user's requirements. Any conflicts, or apparent contradictions between the expert and user's 'view' of the proposed application might first be identified at this stage. At the same time the description of the user environment helps to identify issues such as the existence of different potential classes of user, and their associated (possibly varying) functional and dialogue requirements.

This activity was contained in the methods developed during the course of this research work (cf. Chapter Six; section 6.6.).

'First-pass' Specification
A 'first-pass' specification for the system can be drawn-up from the description of the user environment and the information gleaned from the initial, 'first-pass' knowledge elicitation sessions conducted with the expert(s). This specification contains a description of the functionality for the system, and the dialogue requirements as identified so far. The dialogue requirements can be identified through an analysis of the functions in terms of their associated inputs and output representations (cf. Chapter Six; section 6.6.), and from the existing knowledge of the user environment (see above).

Build Concept / Functions Demonstrator
From the 'first-pass' specification (see above) an early prototype can be developed which illustrates the system in terms of its functional and dialogue requirements as they are so far understood (by the developers) from the existing development work. This demonstrator can be evaluated with both users and expert(s). With the users the system is evaluated primarily to clarify, and further establish user requirements. With the expert(s) the demonstrator is used primarily to clarify what are understood to be the functional requirements for the system.

It is important to stress that the dialogue design contained in the demonstrator represents an initial design, one which will be amended during future iterations of the evaluation process. However, the process by which this initial dialogue design has emerged (that is through the various considerations of user interface requirements in earlier stages) means that the initial design is more likely to be closer to the final requirements than if it had been produced without earlier reference to users. The outcome of this is a reduction in the total number of design iterations required in the overall system development.

Derive Functional Specification
Resulting from the previous stages, a specification of the functionality of the proposed system can now be derived. This specification, although likely to be open to some degree of amendment from future prototype evaluation with experts and users, is now fairly well established. It can now be used to carry out detailed dialogue design in consultation with users; and as the basis for constructing the knowledge base (establishing the knowledge in terms of rules, facts etc.) in consultation with the expert(s).

Detailed Interface Design
Whilst the knowledge base is being constructed through the creation and evaluation of prototype knowledge bases, so through the same iterative process, the user-system dialogue can be specified and implemented. The two processes cannot be carried out totally independently of one another. The nature of the detailed knowledge which is elicited during detailed knowledge engineering at this stage will have some effect upon the dialogue design (for example, in terms of the detailed content of explanations), and as a result the processes should be carried out in parallel. What is important though, and what is stressed in the life-cycle illustration in Figure 9.4., is to continue evaluation of the user-system interface with users alongside the expert-centred work. This is to ensure that the knowledge being elicited from the expert continues to be 'sound', both in its comprehensibility and its utility, in terms of users and their tasks.

The essence of the life-cycle is that it illustrates user interface design as an activity that can be addressed early on in KBS development, and that through the construction of a user interface prototype, it is possible to support the overall development process. The adoption of such an approach to development was found to be of significant use, for example, in the SAM project (cf Chapter Eight).

9.12. Support requirements for the proposed development life-cycle

In the description earlier in the thesis of a 'Three stage process for user interface design' proposed by Williges et al. (1987) (cf. Chapter Two; section 2.6.), Williges was reported as saying of that model:-

> "Such a user-centred approach to interface design requires that the designer, or design team, use appropriate tools to aid in the design process." (Williges 1987)

The same most certainly also holds for the life-cycle just described in the previous section. Successful adherence to the activities described in the life-cycle will depend upon firstly the existence, and subsequently the application by the KBS developers, of suitable and complementary support tools. In the context of the life-cycle described for KBS three main classes of support tool can be identified:-

1. Methods and techniques - for the capture and analysis of user requirements.

2. Documentation - of the various activities or stages during a project development. Documentation is particularly important in order to manage the high-level of iteration, as occurs in KBS development, where several prototype versions of a system are produced and evaluated.

3. Prototyping tools - for the development of the 'Concept /Functions Demonstrator', and for the development of subsequent, more detailed, prototypes. The prototyping tool which is used to create the demonstrator may be different from the tools used subsequently for prototyping. It is feasible in the demonstrator to use a rapid prototyping tool which illustrates, as a 'first-pass', the proposed functionality and user-system dialogue without involving a 'live' knowledge base. This is not, however, feasible for subsequent user interface prototypes which are developed in parallel with the evolving knowledge base . The more detailed user interface prototyping must involve the use of a tool which, if not part of the same development environment being used for knowledge base construction, must be able to interface 'architecturally' (that is in software and hardware terms), at some point to the knowledge base prototyping and subsequent delivery tool(s).

The methods which were developed during this research (cf. Chapter Six) went some way in supporting the design activities suggested by the life-cycle. The analysis of the

functional specification into its inputs and outputs, and the description of the user environment (as described in Chapter Six; section six) can be seen as a basis for the kinds of tools required to support user requirements capture at an early stage in development. They proved to be useful in the two later case studies in this research, providing a way of generating a specification of potential user interface requirements which could be then be utilised as the basis for prototyping and evaluation. Similarly, the documentation of design decisions (cf. Chapter Six; section 6.7.) can be seen as a method to aid in documenting and managing the prototyping iterations within a single project. In the SAM study they became an important part of the overall documentation of that project. Finally, the HyperCard and HyperTalk tools used in that same project were very useful in providing a rapid prototyping tool for producing the first version prototype, or concept demonstrator.

9.13. The issue of automated support

A question exists over the form that human factors support tools should take. In Chapter Six (sections 6.2, and 6.3.) there was some debate over whether to try and build either automated or manual methods of support within this research project. The decision was to take the 'manual route'. This decision was based upon an assessment of specific characteristics in this particular project, and upon more global issues concerning both present human factors and user interface design practice, and the present state of KBS as a technology. The decision to develop manual support methods was on the whole vindicated. The methods that were developed provided a basis for early, and active, involvement on KBS projects, which would not have been possible if time had been spent on developing automated support. As a result it was possible to get much further in assessing present KBS development practice, and the way that the approach being taken to user interface design might be improved in that overall process. The life-cycle outlined earlier, is one example of what resulted from the experiences gleaned from involvement on the KBS projects.

Although a decision was made to develop manual methods within this research project, in the longer term the successful integration of human factors support will be dependent upon the development of automated support tools. The trend within software engineering as a whole is towards automated support (Sartwin 1987; Jones 1986) and human factors will undoubtedly have to follow that trend. However, before this can happen two major issues have to be tackled. Firstly, there has to be a sounder understanding of user interface design as an activity. This means that better techniques will have to be developed, either within human factors or outside that community. In essence, the development of automated support will depend upon better manual support

methods as precursors to that automation process. In Chapter Two in particular, it was shown that although tools such as task analysis, and design guidelines, already exist, their application in 'real-world' design is scant. This is partly as a result of the immaturity of user interface design as a design activity. It also stems from an inadequate appreciation of 'real-world' design practice and the constraints within commercial and industrial environments which are involved in that 'real-world' design. This leads on to the second major issue that needs to be tackled before automated support tools can be developed and used effectively. Such tools will, in the end, only be integrated successfully if there is an awareness of user interface design as an important activity, by those responsible for system development,. It was seen during the course of this research that, for several reasons, such an awareness is far from being widespread. The successful integration of human factors tools into the development process depends very much upon the awareness of user-centred design as an integral design principle. Once established as an important, and in particular 'workable', principle, then any tools in support of that principle are likely to be more acceptable to, and utilised by, system developers.

9.14. Concluding remarks

This chapter has brought together the experiences gained during the research and attempted to derive from that experience a comprehensive view of the present and future for user interface design within KBS development. The experiences have been crystallised into a number of statements about user interface design activity, and into a development life-cycle which places user interface as a parallel, and indeed largely complementary, activity to knowledge base design. The thesis will be now be drawn to a conclusion in the final chapter by summarising the research work and its findings, along with proposals for future research directions in this area.

Chapter Ten

Conclusions
and
Recommendations for Future Research

Outline
As a conclusion, this chapter begins by once again considering the area which the research work focussed upon. It then presents a final summary of the work and the findings that emanated from it. Potential directions for future work in this area are then discussed.

10.1. A final overview of the research area

The research described in this thesis was concerned with user interface design within the development of knowledge-based systems. KBS represent a new and evolving class of computer systems which have grown from work carried out in the field of artificial intelligence. The potential application of KBS in numerous domains, as aids to problem-solving and decision support, is seen as considerable. The commercial and industrial sector is one sector, in particular, which views KBS as having both considerable technological and business potential. It was KBS development work that is taking place in this sector which provided the focus for the thesis work.

The activity of user interface design can be seen as a very important activity within KBS development. Indeed it can be argued that user interface design, at the very least, requires an equal amount of consideration and development effort as is shown to other design activities (such as the construction of the underlying knowledge bases for a system). KBS are an example of interactive systems which, due to the type of information (knowledge) they attempt to capture and transmit, raise several important 'end-user' issues. In total these issues indicate that early, and detailed, consideration of users' requirements of a KBS is crucial if the eventual utility and effectiveness of that KBS is to be ensured.

The importance of user interface design consideration, as just expressed in the previous paragraph, is a view held strongly by human factors specialists. However, through the course of this research it became apparent that there was a considerable gap between human factors design principles, and the application of these principles in the development of KBS, as practised by the majority of commercial and industrial developers. As was gleaned from the early review of the human factors literature, such

a gap, or 'mismatch', between theory and practice is one which not only exists in the context of KBS but is also quite pervasive within interactive systems design generally. Focusing upon KBS, this research set out to identify some of the factors which caused the mismatch; factors which stood in the way of human factors integration, and an improved level of user interface design practice, within KBS development.

10.2. 'Designer-centred' research

In the belief that one of the reasons behind the existing state of poor integration was an inadequate appreciation, by those in human factors, of the design environments, the work monitored KBS development through active involvement on 'live' projects. This belief is now becoming more widespread within the human factors community, with increasing calls for an 'action research' approach to work carried out on the formulation of human factors support. The work reported in this thesis was in essence, 'designer-centred', adopting an action research approach. This approach was engendered through the following particular activities:-

- Involvement in the collaborative Alvey *Intellipse* project. This project provided the initial, and continued basis, and support for the research work; providing insight, through case studies, into a number of KBS developments which took place as part of *Intellipse*.

- Involvement in a further KBS development project, with the Trustees Savings Bank. This project, which was set up independently from the original proposal for work within the *Intellipse* project, acted as a further case study. In this way a clearer, and more representative picture of present KBS and user interface design activity could be constructed.

- Contact with a number of other organisations involved in KBS development. This contact was, on the whole, constituted by a series of interviews with developers from the various organisations. Like the more detailed case studies, the objective of these interviews was to build a picture of current KBS development practice in general, and the approach being taken to user interface design in particular.

In all of the case study projects, involvement was not conducted through simply monitoring or observing, at a passive level, the events which occurred during a project. Instead, as had been the original objective set out in the *Intellipse* project proposals, the emphasis was on the author's (pro-) active involvement. The basis for this involvement

was reciprocity, whereby the author was able to gather information and experience for his research, whilst in return providing support to the development teams on their specific projects. This latter role meant that he had a specific task, or set of tasks, to perform in a project.

The main vehicle for ensuring involvement in the two later case studies (IBS; see Chapter Seven; SAM; see Chapter Eight) was a set of methods which were developed to enable the development teams to conduct an analysis of potential user requirements at an early stage in the project. These methods provided the foundations for improving user interface design practice. As they had been developed with an appreciation of the facets of existing KBS development approaches (for example, the need to deal with the high-level of iteration), they fitted in more easily with those existing approaches. Overall they contributed greatly to a better appreciation of what the user issues, and user requirements of a particular system were.

Whilst the use of the methods contributed to the actual case study projects, their application was not, as has been reported earlier in the thesis, always straightforward. It was reported how, for example, in the IBS project it was not easy to obtain a description, or knowledge, of the proposed user environment. This resulted from the project sponsor's reluctance to involve, or refer to specific end-users. Monitoring the response to applying the various methods was itself both illuminating and valuable, in terms of being able to address the research objectives. Monitoring in this way enabled the various methods to be evaluated, and it was through experience of decisions like the one just mentioned in the context of IBS, that enabled the author to draw up a picture (presented in Chapter Nine) of present development practice and its effect upon user interface design activity.

10.3. A final summary of the research findings and experience

The picture of KBS development practice and user interface design activity which emerged, along with the experiences gained in general throughout the research, led to the proposals for ways in which improved user interface design, and human factors integration, might be achieved. These were reported in the previous chapter (see in particular Chapter Nine; sections 9.10, and 9.11.). In summary, however, the key findings, and proposals, which emerged from the research work are restated below:-

General findings

- The present lack of human factors application in user interface design in commercial and industrial KBS development can be seen as having three main root causes. These fall into the following categories:-
    - The present state of human factors tools (that is guidelines, methods and methodologies etc).
    - The 'immaturity' of KBS as a technology; and the motives and objectives behind present KBS projects.
    - Constraints inherent in commercial and industrial design environments (constraints on resources such as project development time; or lack of human factors trained personnel which brings with it a lack of awareness of user interface design as an important design issue).

  Changes must occur in aspects of all three categories if the identified gap between human factors theory, and human factors practice, is to be significantly reduced, and improved integration is to be achieved.

Specific findings and proposals

- User requirements capture was identified as a fundamental problem in present KBS development practice. This problem manifested itself through inadequate definition of who potential users of systems would be, in terms of important elements such as users' present tasks, their future needs, and their existing domain knowledge. Problems of inadequate early definition of users, was subsequently compounded by a lack of evaluation with end-users. Instead, evaluation which did take place tended to focus upon evaluation of prototype systems with experts as part of the knowledge acquisition phase being conducted to construct the knowledge base(s).

- In response to the above problems concerning requirements capture, it was demonstrated during the two later case studies that it was possible to generate potential user interface requirements early on in a project. Once identified, these potential requirements could form the basis for subsequent iterative evaluation, conducted, in the main, through prototyping. There are several advantages of this early consideration given to the user interface:-

    - It creates an early awareness within the development team of the user interface as 'an issue' - an awareness which appeared to having been missing previously.

- An early analysis of potential requirements means that the development team have a better idea of the basic facilities which any implementation tool(s) would need in order to provide, at the very least, a 'minimum' system. This is likely to reduce problems which can occur later on when, after some development effort, a chosen tool is found to be impractical, and previous effort is therefore wasted.

- The 'first-pass' prototype, or concept demonstrator, was more accurate in terms of capturing user's requirements. This would mean that the number of subsequent iterations needed to capture user requirements, and hence build a system, should be reduced.

- In support of user interface design as an early, and parallel activity alongside knowledge base design, the basis for a new development life-cycle has been proposed (cf. Chapter Nine; section 9.11). This life-cycle draws, in particular, upon the experiences from the SAM case study project (cf. Chapter Eight). It has at its centre the construction of a (knowledge base independent) concept demonstrator, or early prototype, which, through evaluation with potential end-users, can be used early on as a method for capturing user requirements at both the functional (task) level, and the user-system dialogue level.

10.4. Summing-up and Recommendations for future work

The overall approach taken in this research has provided a good insight into KBS development practice, and the factors which create problems concerning user interface design within that practice. This insight has enabled the author to work constructively within KBS teams on 'real-life' development projects. There has been a considerable amount of 'cross-fertilisation' of ideas and perspectives between both author and development teams and this has meant that both parties have learned valuable lessons from each other. Consequently, from the author's point of view and overall objectives, it has been possible to derive important lessons concerning the directions that human factors must take in order to achieve better integration of its methods within KBS development, and indeed, more globally, within interactive systems as a whole.

As has emerged from this research there is much work to be done before human factors techniques and perspectives become better integrated. The work reported in this thesis is one attempt at providing the basis for such integration. Inevitably, given the numerous problematic issues involved and tasks to be faced, this work can only scratch the

surface. In consideration of that fact, and in order to bring this thesis to a close, some areas of future work, that have been indicated by the research, are proposed below:-

1. Continued improvement and expansion of the methods constructed in this work to support developers in the early, and continued, consideration of user interface requirements and design. The methods might be enhanced in the following ways:

    - by addressing support for more detailed user interface design aspects, in particular detailed dialogue design, which will occur after further design and evaluation iterations in the life-cycle. Such support might address the use of notations such as flow chart diagrams for describing dialogue sequences.

    - by supporting evaluation through the provision of guidance for carrying out end-user evaluations at different levels of detail and depth (perhaps support, at one end of the scale, for informal evaluations and at the other end for a more detailed, 'keystroke analysis' of user behaviour). Guidance in this way should also indicate the applicability of different evaluation approaches and their advantages and disadvantages, in terms of what such an evaluation is likely to tell the developer, and what it is likely to miss out. The objective of this guidance is to give developers the opportunity of selecting a particular evaluation approach according to either the stage in development, or to project constraints and resources.

2. Further work on, and also validation of (on further KBS development projects), the life-cycle (proposed in Chapter Nine) for user interface design within KBS. In particular, such work should consider the later stages of a system development which are closer to actual system delivery (this will be discussed again later on). The management of the parallel activities of detailed interface design, and detailed knowledge engineering, in terms of how they feed into one another should be addressed.

3. Establish further, the utility of the design decision documentation for the purposes for which they were designed; that is the utility of this documentation in terms of :
    - support within a single project, for managing and documenting design decisions and the evaluative feedback to these decisions
    - a basis for providing in-house, 'organisation-specific', user interface guidelines for use in other development projects

4. Following on from the above areas to be addressed in future work, a longer term objective should also be to provide automated support where appropriate, for the host of methods, and documentation, developed. This should provide support for activities throughout the development life-cycle and might be integrated under an umbrella 'project support environment' along similar lines to existing software engineering CASE and IPSE tools (cf Chapter Six; section 6.3.). An example of this kind of human factors support environment is exemplified, in part, by the HUFIT project work which has been referred to during this thesis.

As has already been stated, the research experience reported in this thesis dealt primarily with prototype systems - an indication of the predominant 'state of the art' in KBS development at present. Further study should address the issues which arise at later development stages, such as system delivery and maintenance. Indeed, all of the areas identified above, heavily suggest that further, longitudinal study, of KBS development projects is required. A central theme of this research has been the 'designer-centred', approach to the formulation and delivery of human factors support. In terms of the high-level of commercial and industrial collaboration which has been seen in this particular research, such an approach is not easily implemented. Whilst mindful of this problem, it is proposed that the successful integration of human factors will, in the end, be dependent upon the kind of action research approach that was adopted here. Without a thorough appreciation of 'real-life' development practice then the human factors specialist who is concerned with the utilisation of his, or her, design principles and support tools, is liable to make the same mistake as the system developer who does not validate assumptions concerning end-user requirements.

REFERENCES

Alvey Annual Report (1984) — *Alvey Programme, Annual Report 1984.* Alvey Directorate. Publ. IEE.

Alvey Annual Report (1985) — *Alvey Programme, Annual Report 1985.* Alvey Directorate. Publ. IEE.

Alvey Software Engineering Course (1986) — *A Course on User Interface Design.* Presented by William Newman and Karmen Guevara, Beta Chi Design. Cosener's House, Abingdon, 17-19 November 1986.

Apple Computer Inc. (1987) — *HyperCard User's Guide* . California, USA.

Atwood, M.E. (1984) — *A Report on the Vail Workshop on Human Factors in Computer Systems.* IEEE Computer Graphics and Applications, 4, 12. pp 48-66.

Bader, J.L. (1988) — *Knowledge-Based Systems and Software Engineering.* PhD. Thesis. University of Aston in Birmingham.

Bader, J., Edwards, J.E., Harris-Jones, C., Hannaford, D. (1988) — *Practical Engineering of Knowledge-Based Systems.* Information and Software Technology, 30, June 1988. pp 266-277.

Barfield, W. (1986) — *Expert-novice differences for software: implications for problem-solving and knowledge acquisition.* Behaviour and Information Technology, 5, 1. pp 15-29.

Barber, E.O. (1984) — *Expert Systems Survey.* Working Paper, W. 122. Department of Computer Science, University of Exeter.

Barr, A., Feignebaum, E.A. (1981) — *The Handbook of Artificial Intelligence.* Volume One. Pitman.

Barrow, C.W.M. (1985) — In *Alvey Programme, Annual Report 1985.* Section 3. Alvey Directorate. Publ. IEE.

Bell, M.Z. (1985) — *Why Expert Systems Fail.* Journal of Operational Research Society, 7. pp 613-619.

Bellotti, V. (1988) — *Implications of Current Design Practice for the Use of HCI Techniques.* In 'People and Computers IV. Proceedings of the Fourth Conference of the British Computer Society Human Computer Interaction Specialist Group. Jones, D.M., Winder, R. (eds). University of Manchester, England. 5-9 September 1988. Cambridge University Press.

Benchimol, G. (1987)

Developing Expert Systems for Business. English Translation, North Oxford Academic Publishers Ltd.

Berry, D.C., Broadbent, D.E. (1987)

Expert Systems and the man-machine interface. Part Two: The user interface. Expert Systems, 4, 1. February 1987. pp 18-27.

BIS (1984)

Proposal to the Alvey Directorate for Development of IKBS Software Tools. BIS Applied Systems Ltd., Birmingham, November 1984.

BIS (1986)

The BIS / IPSE. Product Brochure. BIS Applied Systems Ltd. London.

BIS (1987a)

Structured Systems Analysis. Course Manual. BIS Applied Systems Ltd. 1987.

BIS (1987b)

BIS / Estimator. Product Brochure. BIS Applied Systems Ltd. London.

Boehm, B.W. (1976)

Software Engineering. IEEE Transactions on Computers, C-25, 12, December 1976, pp 1226-1241.

Brachman, R.J., Amarel, S., Engelman, C., Engelmore, R.S., Feigenbaum, E.A., Wilkins, D.E. (1983)

What are Expert Systems? Chapter Two in 'Building Expert Systems'. Hayes-Roth, F., Waterman, D.A., Lenat, D.B.(eds). Addison-Wesley.

Bright, C.K., Inman, A., Stammers, R.B. (1989)

Human Factors in Expert Systems Design: Can lessons in the promotion of methods be learned from commercial DP? To appear in Interacting with Computers, 1, 2. July 1989.

Bright, C.K., Stammers, R.B. (1988)

The Design of the User Interface: What do Expert System designers need to know? In Proceedings of Human and Organisational Issues of Expert Systems. Berry, D., Hart, A. (eds). Joint ICL and Ergonomics Society Conference, 4-6th May 1988. Stratford-upon-Avon, England.

Buchannan, B.G. (1986)

Expert Systems: working systems and the research literature. Expert Systems, 3, 1. pp 32-51. January 1986.

Buchannan, B.G., Barstow, D., Bechtel, R., Bennett, J., Clancey,W., Kulikowski, C., Mitchell, T., Waterman, D.A. (1983)

Constructing an Expert System. Chapter Five in 'Building Expert Systems'. Hayes-Roth, F., Waterman, D.A., Lenat, D.B.(eds). Addison-Wesley.

Buckley, M., Candy, L., Edmonds, E.A. (1988)

Determining requriements and prototyping the user interface module. In Proceedings of Human and Organisational Issues of Expert Systems. Berry, D., Hart, A. (eds). Joint ICL and Ergonomics Society Conference, 4-6th May 1988. Stratford-upon-Avon, England.

Burton, A.M., Shadbolt, N.R., Hedgecock, A.P., Rugg, G. (1987)

A Formal Evaluation of Knowledge Elicitation Techniques for Expert Systems: domain 1. In Research and Development in Expert Systems IV. Proceedings of Expert Systems '87, Seventh Annual Technical Conference of the BCS Specialist Group. Brighton. Moralee, D.S. (ed.). Cambridge University Press.

Candy, L. (1987)

The Role of the User Interface Team in System Design. Presentation given at Alvey Special Interest Group Workshop on 'The Role of Human Factors in the Assessment and Evaluation Process'. Loughborough University, 16th December, 1987.

Candy, L. (1988)

Design Strategies for Expert Systems: A case study. In Proceedings of Human and Organisational Issues of Expert Systems. Berry, D., Hart, A. (eds). Joint ICL and Ergonomics Society Conference, 4-6th May 1988. Stratford-upon-Avon, England.

Candy, L., Edmonds, E.A. (1988)

Expert System Development for an Office Environment: Users, Evaluation and the Design Process. In Proceedings of Conference on Man-Machine Systems. June 1988.

Canfield-Smith, D., Irby, C., Kimball, R., Verplank, B., Harslem, E. (1982)

Designing the Star User Interface. BYTE. pp 242-274. April 1982.

Card, S.K., Moran, T.P., Newell, A. (1983)

The Psychology of Human-Computer Interaction. Lawrence Erlbaum Associates, Inc.

CCTA (1986)

Expert Systems: Some guidelines. HM Treasury (Central Computer and Telecommunications Agency). UK.

Christie, B., Gardiner, M.M. (1987)

Future Directions. Chapter Ten in 'Applying Cognitive Psychology to User Interface Design'. Gardiner, M.M., Christie, B. (eds). John Wiley & Sons.

Clarke, S.L.H., Tainsh, M.A. (1987)

Director's Report for Human Interface. Research Report:ALV/HI/005/87

Cochran, A.J. (1981)

Vocational PhD's: Aston's IHD Scheme. University of Aston in Birmingham, May 1981.

Cockton, G. (1987) *A New Model for Separable Interactive Systems.* In Proceedings of 'INTERACT'87', the Second IFIP Conference on Human-Computer Interaction. Shackel, B., Bullinger, H-J. (eds). Stuttgart, FRG. September 1-4, 1987. Publ. Elsevier Science.

Connell, C. (1987) *The Application of Expert Systems in UK Industry.* Report funded under Economic and Social Research Council, project No. F24250020.

Coombs, M.J., Alty, J.L. (1980) *Face-to Face guidance of university computer users - II: Characterising advisory interactions.* International Journal of Man-Machine Studies, 12. pp 407-429.

Coombs, M.J., Alty J.L. (1984) *Expert Systems: An alternative paradigm.* International Journal of Man-Machine Studies, 20. pp 21-43.

Cornes, R. (1986) *The Prototyping Principle.* Guardian, January 22nd 1986.

Cornes, R. (1987) *Structure and the Fourth Generation.* Guardian, December 10th, 1987.

Cullen, J., Bryman, A. (1988) *The Knowledge Acquisition Bottleneck: Time for Reassessment?* Expert Systems, 5, 3. pp 216-225.

d'Agapeyeff, A. (1984) *A Short Survey of Expert Systems in UK Business.* R&D Management, 15, 2. pp 89-99.

d'Agapeyeff, A. (1987) *Report to The Alvey Directorate on The Second Short Survey of Expert Systems in U.K. Business.* Publ. IEE on behalf of The Alvey Directorate, London, August 1987.

Damodaran, L. (1988) *Integrating Human Factors Principles into Structured Design Methodologies. A Case Study in the UK Civil Service.* In 'Information Technology for Organisational Systems'. Bullinger, H-J.et al. (eds). Elsevier Science Publishers.

Durham, T. (1987) *Moving Experts Forward One Step at a Time.* Computing, September 17th, 1987. pp 20-21.

Eason, K. (1984) *Towards the experimental study of usability.* Behaviour and Information Technology, 3, 2, pp 133-143.

Eason, K.D., Harker, S.D.P. (1987) *A User Centred Approach to the Design of a Knowledge Based System.* In Proceedings of 'INTERACT'87', the Second IFIP Conference on Human-Computer Interaction. Shackel, B., Bullinger, H-J. (eds). Stuttgart, FRG. September 1-4, 1987. Publ. Elsevier Science.

Edmonds, E. (1982) *The man-computer interface: a note on concepts and design.* International Journal of Man-Machine Studies. 16. pp 231-236.

Eike, D.R., Fleger, S.A., Phillips, E.R. (1986) *User Interface Design Guidelines for Expert Troubleshooting Systems.* In Proceedings of the Human Factors Society 30th Annual Meeting. p1024-1028. Dayton. Ohio.

Engel, S.E., Granda, R.E. (1985) *Guidelines for Man/Display Interfaces.* IBM Technical Report TR 00.2720. Poughkeepsie, New York.

Expertech Ltd. (1987) *Xi Plus.* Product Brochure.

Fish, A.N. (1988) *Expertise in the Human-Computer System: Dispensing with the Expert System Metaphor.* In Proceedings of Human and Organisational Issues of Expert Systems. Berry, D., Hart, A. (eds). Joint ICL and Ergonomics Society Conference, 4-6th May 1988. Stratford-upon-Avon, England.

Ford, L. (1986) *Artificial Intelligence and Software Engineering: A Tutorial Introduction to their Relationship.* Artificial Intelligence Review, 1, 1. pp 255-273.

Galer, M., Russell, A.J. (1987) *The Presentation of Human Factors to Designers of I.T. Products.* In Proceedings of 'INTERACT'87', the Second IFIP Conference on Human-Computer Interaction. Shackel, B., Bullinger, H-J. (eds). pp 11-16. Stuttgart, FRG. September 1-4, 1987. Elsevier Science.

Garg-Janardan, C., Salvendy, G. (1988) *The contributions of cognitive engineering to the design and use of expert systems.* Behaviour and Information Technology, 7, 3. pp 323-342.

Gaschnig, J., Klahr, P., Polpe, H., Shortliffe, E., Terry, A. (1983) *Evaluation of Expert Systems: Issues and Case Studies.* Chapter Eight in 'Building Expert Systems'. Hayes-Roth, F.,Waterman, D.A., Lenat, D.B.(eds). Addison-Wesley.

GEC (1987) *GENOS: Integrated Project Support Environment.* Product Brochure. GEC Software Ltd. London.

Gillett, P.R. (1987) *Victory from the Jaws of Defeat.* Report on the ALFEX club at Alvey Knowledge Based Systems Club, Open Meeting. Alvey Conference. UMIST. July 1987.

Goodman, D. (1987) *The Complete HYPERCARD Handbook.* Bantam Books.

Goodwin, N.C. (1987) *Functionality and Usability.* Communications of the ACM, 30, 3. March 1987.

Gould, J.D. (1987) *How to Design Usable Systems.* In Proceedings of 'INTERACT'87', the Second IFIP Conference on Human-Computer Interaction. Shackel, B., Bullinger, H-J. (eds). Stuttgart, FRG. September 1-4, 1987. Publ. Elsevier Science.

Gould, J.D., Lewis, C. (1985) *Designing for Usability: Key Principles and What Designers Think.* Communications of the ACM, 28. pp 300-311.

Green, M. (1985) *Report on dialogue specification tools.* In 'User Interface Management Systems'. G.E. Pfaff (ed.). Springer Verlag.

Hammond, N., Gardiner, M.M., Christie, B., Marshall, C. (1987) *The role of cognitive psychology in user-interface design.* Chapter Two in 'Applying Cognitive Psychology to User Interface Design'. Gardiner, M.M., Christie, B. (eds). John Wiley & Sons.

Hammond, N., Jorgensen, A., Maclean, A., Barnard,P., Long, J. (1983) *Design Practice and Interface Usability: Evidence from interviews and designers.* IBM Hursley Human Factors Report HF082, Hursley Park, Winchester U.K.

Harrington, R.J. (1986) *KES- An Expert System Development Tool.* Conference on 'Expert Systems: Available Hardware and Software Conference'. Mol, 18-19 June 1986.

Hart, A. (1986) *Knowledge Acquisition for Expert Systems.* Kogan Page.

Hayes-Roth, F. (1984) *The Knowledge-Based Expert System: A Tutorial.* Computer, September 1984. IEEE.

Hayes-Roth, F., Waterman, D.A., Lenat, D.B. (1983) *An Overview of Expert Systems.* Chapter One in 'Building Expert Systems'. Hayes-Roth, F., Waterman, D.A., Lenat, D.B.(eds). Addison-Wesley.

Hayward, S. (1987) *A Structured Development Methodology for Expert Systems.* In Proceedings of KBS'86 Conference. Online Publishers.

Hekmatpour, S., Ince, D.C. (1986) *Rapid software prototyping.* In Oxford Surveys in Information Technology. Vol.3. pp 37-76. Oxford University Press.

Ince, D. (1988)

Software Prototyping and Artificial Intelligence Based Software Tools. In Research and Development in Expert Systems V. Proceedings of Expert Systems 88, the Eighth Annual Technical Conference of the BCS Specialist Group on Expert Systems. Kelly, B., Rector, A. (eds). Brighton, 12-15 December. Cambridge University Press.

Inference Corporation (1986)

Inference ART: Automated Reasoning Tool. Product Brochure. Ferranti plc. 1986.

Intellicorp, Inc. (1987)

KEE: The Knowledge Engineering Environment. Product Brochure. Intellicorp, Inc.

Intelligent Environments (1986)

Crystal. Product Brochure.

Jackson, P., Lefrere, P. (1984)

On the application of rule-based techniques to the design of advice-giving systems. International Journal of Man-Machine Studies, 20, pp 63-86.

Johnson, P., Diaper, D., Long, J. (1984)

Tasks, Skills and Knowledge: Task Analysis for Knowledge Based Descriptions. In Proceedings of 'INTERACT' 84', the First IFIP Conference on Human-Computer Interaction. London, England September 1-4, 1984. Publ. Elsevier Science.

Johnson-Laird, P.N. (1985)

Mental Models. Chapter Four in 'Issues in Cognitive Modeling'. Aitkenhead, A.M., Slack, J.M. (eds). Lawrence Erlbaum Associates.

Jones, R. (1986)

An Automation Revolution Hits System Design. Computing, July 17th, 1986. pp 16-17.

Jones, R. (1987)

Pearl is assured of wisdom with ipse. Computing, March 5th 1987.

Kidd, A.L. (1983)

Human Factors in Expert Systems. In Proceedings of The Ergonomics Society's Conference 1983. Coombes, K. (ed). Taylor & Francis.

Kidd, A. (1984)

Human Factors Problems in the Design and Use of Expert Systems. Chapter Fourteen in 'Fundamentals of Human-Computer Interaction', Monk, A. (ed). Academic Press.

Kidd, A.L. (1985)

What Do Users Ask? - Some Thoughts On Diagnostic Advice. In Proceedings of Expert Systems'85, Fifth Technical Conference of the BCS Specialist Group on Expert Systems. Merry, M. (ed.). 17-19 December. University of Warwick. Cambridge University Press.

Kidd, A.L., Cooper, M.B. (1985)

Man-machine interface issues in the construction and use of an expert system. International Journal of Man-machine Studies, 22. pp 91-102.

Kielty, J. (1987)

Selecting an Application. Tutorial Notes from seminar presented at Seventh Annual Technical Conference of the BCS Specialist Group. 14-17 December 1987. Brighton.

Kieras, D., Polson, P. G. (1985)

An Approach to the Formal Analysis of User Complexity. International Journal of Man-Machine Studies, 22, pp 365-394.

Klein, L., Newman, W. (1987)

A Strategy for Integrating Human-Computer Interface Considerations into Alvey-2 Application Projects. Report to the Alvey Human Interface Club. Open Meeting, London, 13th October 1987.

Lehner, P.E., Zirk, D.A. (1987)

Cognitive Factors in User / Expert-System Interaction. Human Factors, 29, 1. pp 97-109.

Leitch, T., Dulieu, M.R.W. (1987)

RESCU Revisited. A Review of a Practical Real-time Expert System. In Research and Development in Expert Systems IV. Proceedings of Expert Systems '87, Seventh Annual Technical Conference of the BCS Specialist Group. Moralee, D.S. (Ed.). 14-17 December 1987. Brighton. Cambridge University Press.

Lenorovitz, D.R., Reaux, R.A. (1986)

Integrating Human Factors Guidance Information with the USI Design / Rapid Prototyping Process. In Proceedings of the IEEE 1986 National Aerospace and Electronics Conference NAECON 1986. Volume Three. 19-23 May. Dayton, Ohio, USA. IEEE New York.

Madni, A.M. (1988)

The Role of Human Factors in Expert Systems Design and Acceptance. Human Factors, 30, 4. pp 395-414.

Maguire, M. (1982)

An evaluation of published recomendations on the design of man-computer dialogues. International Journal of Man-Machine Studies, 16, pp 237-261.

Mantei, M.M., Teorey, T.J. (1988)

Cost / Benefit analysis for incorporating human factors in the software life-cycle. Communications of the ACM, 30, 4, pp 428-439.

Marshall, C., Nelson, C., Gardiner, M.M. (1987)

Design Guidelines. Chapter Eight in 'Applying Cognitive Psychology to User Interface Design'. Gardiner, M.M., Christie, B. (eds). John Wiley & Sons.

McDermott, J. (1982)

R1: A Rule-based Configurer of Computer Systems. Artifical Intelligence, 19. pp 39-88. September 1982.

Milner, N. P. (1988)

A Review of Human Performance and Preferences with Different Input Devices to Computer Systems. In 'People and Computers IV. Proceedings of the Fourth Conference of the British Computer Society Human Computer Interaction Specialist Group. Jones, D.M., Winder, R. (eds). University of Manchester, England. 5-9 September 1988. Cambridge University Press.

Minsky, M. (1975)

A Framework for Representing Knowledge. Chapter Six in 'The Psychology of Computer Vision'. Winston, P.H. (ed). McGraw-Hill.

Montgomery, A. (1988)

GEMINI - Government Expert Systems Methodology Initiative. In Research and Development in Expert Systems V. Proceedings of Expert Systems 88, the Eighth Annual Technical Conference of the BCS Specialist Group on Expert Systems. Kelly, B., Rector, A. (eds). Brighton, 12-15 December. Cambridge University Press.

Moran, T.P. (1981 a)

An Applied Psychology of the User. In 'Special Issue: The Psychology of Human-Computer Interaction.' Computing Surveys, 13, 1, March 1981. ACM.

Moran, T.P. (1981b)

The Command Language Grammar: a representation for the user interface of interactive computer systems. International Journal of Man-Machine Studies, 15, pp 3-50.

Moran, T.P. (1983)

Getting into a System: External-Internal Task Mapping Analysis. In Proceedings of CHI'83-Human Factors in Computing Systems, ACM, New York.

Morris, A.J. (1986)

Why People aren't using Expert Systems. In Proceedings of Second International Expert Systems Conference. September 1986, London. Learned Information.

Morris, A. (1987)

Expert systems - Interface Insight. In People and Computers III, Proceedings of the Third Conference of BCS Human-Computer Interaction Specialist Group. Diaper, D., Winder, R. (Eds). September 7-11, University of Exeter. Cambridge University Press.

Mumford, E. (1983)

Designing Human Systems for New Technology: The ETHICS method. Manchester Business School.

Mumford, E. (1988)

Participative Design for Expert Systems. In Proceedings of Fourth International Expert Systems Conference.7-9 June 1988, London. Learned Information.

Newman, W. (1988)

The representation of user interface style. In 'People and Computers IV. Proceedings of the Fourth Conference of the British Computer Society Human Computer Interaction Specialist Group. Jones, D.M., Winder, R. (eds). University of Manchester, England. 5-9 September 1988. Cambridge University Press.

Norman, D.A. (1986)

Cognitive Engineering. Chapter Three in 'User Centered System Design. New Perspectives on Human-Computer Interaction.' Norman, D.A., Draper, S.W. (eds). Lawrence Erlbaum Associates.

Ovum Ltd with Segal Quince Wicksteed (1988)

Expert Systems in Britain. An overview of the UK Expert Systems Industry based on a report to the Department of Trade and Industry. British Crown Copyright.

Partridge, D. (1988)

To add AI, or not to add AI? In Research and Development in Expert Systems V. Proceedings of Expert Systems 88, the Eighth Annual Technical Conference of the BCS Specialist Group on Expert Systems. Kelly, B., Rector, A. (eds). Brighton, 12-15 December. Cambridge University Press.

Payne, S.J. (1984)

Task-Action Grammars. In Proceedings of 'INTERACT' 84', the First IFIP Conference on Human-Computer Interaction. London, England September 1-4, 1984. Publ. Elsevier Science.

Pollack, M.E., Hirschberg, J., Webber, B. (1982)

User Participation in the Reasoning Processes of Expert Systems. In Proceedings of AAAI-82, The National Conference of A.I.

Rector, A.L., Newton, P.D., Marsden, P.H. (1985)

What kind of system does an expert need? In 'People and Computers: Designing the Interface'. Proceedings of the First Conference of the British Computer Society Human-Computer Interaction Specialist Group. Johnson, P., Cook, S. (ed). University of East Anglia. Cambridge University Press.

Rich, E. (1983)

Artificial Intelligence. McGraw-Hill.

Robertson, D. (1987)

Closing the Gap. Report to the Alvey Human Interface Club. Open Meeting, London, 13th October 1987.

Rodger, M.A., Edwards, J.S. (1989)

*A Problem-Driven Approach to Expert System Development.*Submitted to Journal of Operational Research Society.

Rouse, W.B. (1987)

Much Ado About Data. Human Factors Society Bulletin, 30, 9. pp 1-3.

Russell, A.J., Galer, M.D. (1987)

Designing Human Factors Aids for Designers. In Cognitive Engineering in the Design of Human-Computer Interaction and Expert Systems, Proceedings of the Second International Conference on Human-Computer Interaction, Salvendy, G. (ed). Volume 2. Honolulu, Hawaii. 10-14 August. Elsevier.

Sartwin, D. (1987)

IPSE: The Office of the Educated Professional. Computer Bulletin, 3, 1. March 1987.

Schank, C., Abelson, R.P. (1977)

Scripts, Plans, Goals and Understanding. Erlbaum, Hillsdale, N.J.

Shackel, B. (1986)

Ergonomics in Design for Usability. In 'People and Computers: Designing for Usability' Proceedings of the Second Conference of the British Computer Society Human Computer Interaction Specialist Group. Harrison, M.D., Monk, A.F. (eds). University of York, England. 23-26 September 1986. Cambridge University Press.

Shneiderman, B. (1980)

Software Psychology. Winthrop Publishers Inc., Massachussetts.

Shneiderman, B. (1986)

Designing the User Interface. Addison Wesley.

Shortliffe, E.H. (1976)

Computer-based medical consultation: MYCIN. New York, American Elsevier.

Shpilberg, D., Graham, L.E., Schatz, H. (1986)

ExperTAX: An Expert System for Corporate Tax Planning. In Proceedings of Second International Expert Systems Conference. September 1986, London. Learned Information.

Smith, S.L. (1986)

Standards versus guidelines for designing user interface software. Behaviour and Information Technology, 5, 1. pp 47-61.

Smith, S.L., Mosier, J.N. (1984)

The user interface to computer-based information systems: a survey of current design practice. Behaviour and Information Technology, 3, 3. pp 195-203.

Smith, S.L., Mosier, J.N. (1986)

Guidelines for Designing User Interface Software. MITRE Corporation Report No. 10090. Bedford Massachussetts, USA.

Trimble, G. (1988)

Practical Issues in Expert System Development. Talk given at BCS HCI Specialist Group Seminar on 'Knowledge Elicitation'. February 1988. London.

Underwood, M.J. (1987)

AHIC Response to the IT'86 (Bide) Report: Organisation. Report to the Alvey Human Interface Club. Open Meeting 13th October. pp 15-28. London.

Waterman, D.A. (1986)

How do expert systems differ from conventional programs? Expert Systems, 3, 1. pp 16-22. January 1986.

Williges, R.C. (1987)

The use of models in human-computer interface design. The Ergonomics Society Lecture. Ergonomics, Vol 30, 3, pp 491-503. March 1987. Taylor & Francis.

Williges, R.C., Williges, B.H., Elkerton, J. (1987)

Software Interface Design. In 'Handbook of Human Factors'. Salvendy, G. (ed). pp 1416-1449. Wiley, New York.

Wilson, M. (1988)

What user models aren't, to designers. Seminar notes from talk given at BCS HCI Specialist Group Seminar on 'User Models'. November 1988. London.

Wilson, M.D., Barnard, P.J., Maclean, A. (1986)

Task Analyses in Human-Computer Interaction. IBM Hursley Human Factors Report HF122, Hursley Park, Winchester U.K.

Winograd, T. (1985)

What Does It mean to Understand Language? Chapter Eight in 'Issues in Cognitive Modeling'. Aitkenhead, A.M., Slack, J.M. (eds). Lawrence Erlbaum Associates.

Winograd, T., Flores, F. (1987)

Understanding Computers and Cognition: A New Foundation for Design. Addison-Wesley.

Woods, W. A. (1975)

What's in a Link: Foundations for Semantic Networks. In 'Representation and Understanding', Bobrow, D.G., Collins, A. (eds). Academic Press. New York.

APPENDIX 1

ISSUES CONCERNING THE USE OF NATURAL LANGUAGE FOR HUMAN COMPUTER INTERACTION

With reference to the development of a natural language front-end for INTELLIPSE

Clive Bright
Aston University
25th March 1987

ISSUES CONCERNING THE USE OF NATURAL LANGUAGE FOR HUMAN-COMPUTER INTERACTION

This short report looks at the use of natural language as a mode of human -computer interaction. It first considers the general issues concerning natural language interfaces and then relates some of these issues to the development of a front-end to 'Advisor' and 'Designer ' in the present INTELLIPSE project.

Introduction

In the original specification for the INTELLIPSE project it was stated that it would be desirable to produce a natural language front-end (NLFE) for the computer system under development. We are now almost half way through the project and more concrete ideas are developing concerning the nature of the system. In the light of this it is worth re-assessing the role of a NLFE as the system interface. By highlighting some of the problems associated with the use of natural language I hope to demonstrate that we ought not simply think in terms of natural language interaction but also consider the alternative methods available. I am not suggesting that we totally disregard the idea of a NLFE, just that at this stage we give the alternatives equal consideration. This report will mention some of the problems which are associated with the use of natural language in a computer interface. Given these inherent problems it would be unwise to adhere solely to the idea of a NLFE. It is too early to state exactly what form the interface will take, this should wait until we have further assessed the coverage of the system, both in terms of it's functionality and it's potential users.

Human-computer dialogue styles

There are several existing methods of human-computer dialogue. These can generally be classified into the following four main groups : -

i - Command language
ii - Menus
iii - Form-filling
iv - Natural language

The aim of all of these dialogue styles is to provide an efficient channel of communication (entering and eliciting information) between the computer user and the computer application (i.e. software). Within interface design the decision to implement one style

rather than another is dependent upon an assessment of the nature of the application and more particularly the nature of the users involved. No one method of interaction is seen as wholly superior over the rest. Each method tends to be better suited to a particular type of environment, for example menus are more suited to novice users of a system whilst command languages are better suited to more experienced users.

There are associated disadadvantages as well as advantages with each dialogue method. For example, menu hierarachies can become complex and rather torturous for users. Similarly command languages require considerable learning and re-learning, especially if users have experience of several different systems, each with their own command language and accompanying syntax.

Research into natural language understanding within A.I. has provided another potential interaction medium, i.e. communicating with a computer using the language you would use when communicating with another human. At first this idea sounds very seductive, however on closer inspection it appears that there are also several pit-falls associated with the use of natural language.

Some of these pit-falls, which will be set out below, should be duly considered when we are developing the front-end for Advisor and Designer.

Natural language dialogue

In general the applied use of natural language front-ends have largely been directed in the area of data base systems (e.g. PLANES - data base question/answering system; ROBOT - a NL processor for data base query). A NLFE provides a method for interrogating large databases which is considerably more flexible than menus and which also removes the need to learn an unnatural syntax of the artificial command language. Beyond the area of data base systems use of natural language has largely been confined to more theoretical A.I.projects (e.g. Winograd's SHRDLU).

There are several arguments both for and against the use of natural language. Natural language interaction does have certain advantageous characteristics, however technology in this field (both in terms of hardware and software) is not at a stage where robust NL systems can yet be developed. For instance we are still relying upon the keyboard as a main input device, this in itself is unnatural. The advent of more advanced speech recognition systems may well rekindle interest in NL processing systems. At present though, there is a movement away from NL interaction within MMI design.

General advantages and disadvantages of natural language interaction :-

Advantages

- it is most natural for users; does not require learning for specific application
- flexible and powerful for constructing queries/responses
- user can construct his or her own queries

Disadvantages

- high programming/memory overheads
- user attributes system with more capabilities/intelligence than really exists
- user not always aware of what questions can be asked
- flexibility of NL does not provide any guiding structure or form
- NL is full of ambiguities
- often requires long clarification dialogues where attempts are made to resolve mis-understanding by system or user
- natural language is seldom syntactically correct - spoken English is very different from written English
- requires several keystrokes - typing errors slow the interaction and increase need for clarification dialogue
- the keyboard is an unnatural communication channel

Natural language and INTELLIPSE

These issues cited above must be considered when designing the interface to Advisor and Designer in our own project.

At present we have a prototype Advisor system which contains an element of natural language processing, namely the query-analyser. This is used solely for one purpose - to view a particular piece of text on a topic. Advisor is unlike data base systems where when interrogating a data base a user may only want certain pieces of information, or may want information in a certain order and will formulate his query accordingly. Advisor does not have the same underlying structure as a data base system.The text within Advisor is 'static' and cannot itself be manipulated. It is the emphasis on manipulation of information which makes natural language a potentially powerful tool as

a front end to data base systems. Such manipulation does not exist within Advisor and consequently the need for natural language is diminished. Indeed a natural language facility may be undesirable as it may give a false impression of the underlying structure of Advisor.

Natural language interfaces are more appropriate where there is a more continuous dialogue between user and system. Within such dialogues a system must have the capacity to deal with such elements as ellipsis and pronomial reference. However Advisor as it stands does not require continuous dialogue. At the most what is required is a clarification of synonyms and perhaps basic spelling/typing errors. There is no doubt, though, that some kind of 'fast-path' facility, acting as an alternative to the menu system in Advisor, is required. We might look at form-filling as an alternative where the user is directed to the elements which he must enter to access text (i.e 'query-type' and 'topic name'). Some kind of clarification dialogue will be necessary to cover incorrect entries (synonyms etc) but this might be simply involve a list of 'best matches' or an error message.

It is possible that a natural language dialogue will be desirable for Designer, however it is too early to say exactly what form the interface should take. The possibility of a NL interface should be considered equally alongside the other alternative modes of dialogue. As mentioned earlier it is important to remain open-minded about the ultimate mode of interaction. Following closer analysis of what is required of Designer we should be able to say which mode(s) are most appropriate. As we can see natural language processing has several accompanying problems which increases the need to be cautious before sticking rigidly to the idea of implementing a natural language front end.

APPENDIX 2

A PRELIMINARY APPRAISAL OF ADVISOR :

CONSIDERATIONS FOR POSSIBLE IMPROVEMENTS

CLIVE BRIGHT - ASTON UNIVERSITY - 24th November 1986

Further considerations :

MENU SELECTION -

1. The wording of each menu option should be reduced to improve searching by the user. At the moment each menu option is quite lengthy and this may well increase the amount of time taken for a selection. It also may hinder any improvement in selection time which occurs as the user becomes familiar with the menu layouts.

2. The use of a MOUSE for menu selection should be considered. A decision on this should be taken quickly. If a mouse is to be used then this might result in considerable re-structuring of the present screen layouts.

3. The numbering of each option in a menu should be considered. This might be used in conjunction with a mouse, therefore facilitating two modes of selection (mouse and keyboard).
   IF numbering is used then thought must be taken over the present "dynamic" menus (i.e. options changing place in a menu according to the position in the system). The user may associate a number with a particular option and if menus were not to be "static" then once again any improvement in menu searching/selecting over time will be impaired.

4. It has been reported that when using menus the contents should remain the same at each level i.e. do not remove items/options from a central menu if they become unavailable at a particular point. Instead if an option becomes unavailable at a given point then this should be indicated by using italics or a different font etc.

EXTRA FACILITIES -

5. When asking about a particular topic it is worth considering implementing a better method for accessing all query-types for the same topic. At present this requires returning to the activity menu screen and beginning again. A menu option to look at several query types might be provided. It seems likely that a user will occasionally want to look at all the information on a topic rather than one particular query "slant".

   At present there is a proposed facility ("Examine text of multiple available topics") where the query type is kept static and different topics are chosen i.e text of one query type is generated for different chosen topics. However it seems more likely that the user would want a facility where he/she can keep one topic static and then choose a combination of query types for that topic (for example they might want to find out everything about data analysis).

6. There must be some Scroll facility between pages of text. At the moment there is no such facility which means if the user wants to read something from a previous page he/she must go through the procedure of asking the question again which is very undesirable.

7. At present it is necessary for the user to establish which query types are available/permissible for a particular topic. To find this out the user must choose the relevant option from the activity menu. It would be preferable if instead the query types pertaining to a topic were displayed without such a search. This might be done within a separate menu which was displayed in the main screen - then the user also select the required query.

   If two menus were displayed on the same screen then a mouse would almost certainly be a preferred extra - facilitating swapping between menus.

8. The use of FUNCTION KEYS should be considered. Such keys might be used to allow the user to return to a previous screen or indeed to move to the next screen. At present the user has the option to move up or down a "level" but this has to be done via menu options. Such an implementation would have the added advantage of reducing the number of options in a menu.

Appendix 3

An Analysis of Consultant-Client advisory sessions:

Issues arising from interview with Mr Keith Ball (Senior Consultant, BIS Applied Systems, Manchester. 8.4.87)

GENERAL ISSUES ARISING FROM MEETING WITH KEITH BALL - BIS MANCHESTER (8-4-87)

1. Keith stated that the consulatnt advisory role was very much one of providing re-assurance to the clients rather than as a more formal 'teacher' role. Clients would ask whether what they were doing was correct rather than ask how to do something. This point has consequences for the development of a computer-based support system. It reinforces the need to avoid the development of a CBT system.

2. An important, and re-occuring point was that client organisations do not take simply take on a set of MODUS techniques (i.e SSA or SSA etc) but that they have the general techniques "tailored" to meet their own requirements and the nature of their domain. In terms of providing automated support (re: 'Advisor') any system would have to have it's own 'tailorable' capability (or else be very general).

3. Keith expressed the idea that an Advisor type system should contain several examples as well as general text. He stressed the importance of inclusion of examples and said it would be useful if the system could also facilitate storage of real-life examples (problems which were solved etc.) which an organisation experienced at any point in the application of the new techniques.

4. It was mentioned that there is often noticeable resistance to implementing new techniques within organisations. A support system must be made attractive enough to ensure it's use!

5. Advisory sessions involve a considerable amount of 'whiteboard' work (or similar medium). This means that tape-recording sessions would prove largely un-fruitful.

Specific issues arising from interview with Keith Ball (interview notes)

Do you deal with a common set of companies/organisations ?

- **Cover a wide variety of institutions**

- **Clients include -** **Banks**
  Insurance companies
  Breweries
  Manufacturing companies

How long does the advisory support continue (on average) ?
What determines its termination (assessment of ability or client choice) ?

- **Continues indefinitely**

- **Client retains the services and determines whether to continue or not**

- **Sometimes client organisations work on the basis of a consultancy budget**

Is all the training and supplementary advice carried out 'in-house'?

- **Training is carried out both 'in-house' and through public courses**

- **Subsequent advisory support is largely provided 'in-house'**

Have the organisations previous experience of MODUS ?
Any other methodology ?

- **Although they have experience in systems design generally they do not usually have experience of MODUS other than through the courses**

- **They are therefore relatively new to the specific BIS techniques**

At what level are the people that you advise (project managers, designers ..) ?

- **They range from users to development staff**

- **In particular system analysts and designers**

Is any advisory support provided from within the organisation (formally or informally via colleagues) ?

- **Yes. It is obviously in the clients best (financial) interests to clarify problems internally**

- **This is more feasible if there are people within who have already been trained, as is sometimes the case**

How long does each advisory session usually last ?

- **Highly variable and difficult to predict at outset**

Do you often take work away from these sessions (to later comment on) ?

- **Documents are often sent to the consultant prior to session for him to analyse and comment on during a session**

- **Problems can arise during a session which the consultant will need to go away and check-up on**

Do you refer to the manuals for your own clarification during a session ?

- **Yes. Reference is often made to the standards manuals during a session**

Is your advice largely based upon the manuals or upon your own experiential knowledge ?

- **Most advice is based upon personal experience of applying the methodology**

- **The m/dology provides a framework of reference but this has to be embellished with consultant's own knowledge from his practical experience**

Is advice often given/sought over the phone ?

- **Used largely for general problem definition rather than for specific advice**

- **(N.B. as a result not condusive to my overall proposed analysis)**

Are the advice sessions orientated towards people who all have the same level of experience/knowledge ?

- **Generally yes. It is assumed that they know the basics and have been on a course**

- **They are therefore relatively new to the technique**

Do the advisory sessions cover all stages within MODUS ?
Is there a tendency to cover particular stages more often ?

- **Advice will be given on SSA or SSD etc, but usually one particular consultant is well versed in one field and less experienced in the others. Unless the problem is within his capabilities he will pass it on to another appropriately experienced consultant.**

Are there common problem areas which always require attention ?
(Within context of SSA)

- **Interestingly, no. Tendency for different types of problems to arise. This is probably accounted for by the effect of applying, and 'tailoring' the methodology to many different domains within different institutions (i.e. financial, manufacturing etc)**

- **However, concerning courses there are sometimes topics which often cause problems. This is at a general level and prior to applying the techniques to a specific domain. It is at the more specific level which we are directing our system - the intention is not to develop a CBT system.**

What form do the advisory sessions take ?

- formal / informal
- group / individual
- sit-down / 'move-around'

- **The sessions tend to be very fluid and cannot easily be classified.**

- **The consultant's service is retained and will be called in when necessary.**

- **He is often sent documentation from an on-going project where the organisation are beginning to apply the new techniques. He will study it and later meet to suggest changes and give subsequent advice.**

- **The consultant will sit in on 'reviews' carried out by clients and will comment at any point when necessary. (From discussion it appears that these review sessions would not be sufficiently well structured for detailed advisory analysis)**

Appendix 4

Questionnaire on User interface design practice in Expert System development.

(Distributed internally at Aston University and subsequently formed basis for interviews carried out with commercial and industrial developers)

Name Department

This questionnaire acts as a pilot survey and is being distributed to people involved in expert system work internally at Aston. It will later be distributed to people outside in industry who are involved in such work. After filling in the questionnaire any comments concerning the style of the questionnaire (format, content etc) would also be appreciated before it is distributed externally. There is space provided at the end for you to make any comments.

The Design of Expert System User Interfaces

This questionnaire is directed at people involved, either currently or in the past, in the building of expert, or knowledge-based, systems. The main aim is to assess the level of importance given to the construction of the user interface within the design of the expert system.

The user interface refers to such system features as the style of dialogue between system and user (typically via one or more of menus, form filling, natural language, icons and direct manipulation); the type of input devices (e.g. keyboard, mouse etc.); use of windowing facilities; the built-in facilities of the expert system to explain its outputs and decisions (explanation facility).

Section 1 of the questionnaire deals with more general issues whilst Section 2 addresses in more detail the design considerations of the expert system user interface.

Section 1 - General design issues

1. What domain is your system addressing, e.g., medicine, engineering, electronics ? Please be as informative as possible.

2. What is the function of your expert system, e.g. diagnosis, advice-giving, intelligent front-end etc? Please specify.

3. At what stage are you in the development of your expert system ? (Please ring) :-

a) feasibility

b) design/prototyping

c) implementation

d) evaluation and testing

e) everyday use

4. What is the purpose of your expert system work (Please ring) :-

a) to become familiarised with expert system technology

b) to demonstrate expert system feasibility

c) to design and implement an expert system tool

d) other purpose (please specify) -

5. Is your system being built using :-

a) an expert system shell
(e.g. ESP/Advisor, ExpertEase, Xi, Crystal, TIMM)

If Yes then please specify.

If Yes then what facilities, if any, does the shell contain to allow you to build your own user interface?

b) a programming language (e.g. LISP, Prolog, etc)
Please specify.

---

6. Please indicate which of the following hardware environments you are using to develop your system. (Please ring any combination)

a) mainframe

b) mini

c) workstation

d) PC

e) other (please specify)

Section 2 - **User interface consideration**

---

1. How much attention has been paid, or will be paid, to the design of the user interface of the expert system.
Please ring the level of consideration you believe is closest to your particular situation :-

a) little or no attention i.e. - vague specification of intended end-users

- little consideration to the design of the user interface beyond the use of the basic facilities inherent in the shell or programming language being used.

(go on to question 2)

b) considerable attention i.e. - clearly defined set of intended end-users

- some user evaluation during design
- significant programming effort concentrated on the user interface

(go straight on to question 3)

2. (**Little or no attention**)

For what reason(s) has the user interface received little attention (please indicate any combination of the following reasons) :-

a) There is a lack of awareness of user interface issues/guidelines for expert systems.

b) The user interface is not seen as sufficiently important to warrant much attention.

c) There are insufficient tools or techniques to support the design of the user interface.

d) There are no members involved in the design of the expert system who have suitable knowledge of user interface design issues.

e) You believe it is too early in the development of the system to assess the user interface requirements.
(if this is so then please state the present stage that you are at) -

f) There is insufficient project time to allow much attention to the design of the user interface.

The following possible reasons are specifically directed at those using an expert system shell :-

g) The design of the user interface is catered for sufficiently by built-in facilities in the shell and does not require further consideration

h) Any consideration of the user interface is largely constrained by the particular shell being used.

If there are any other factors which you think have led to lack of attention to the design of the user interface then please comment below :-

Move straight on to question 4

3. **(Considerable attention)**

In what ways have you focused, or do you intend to focus, on the design of the user interface ?
Please ring any combination of the methods listed below and add any other procedures you may have carried out

a) special user interface development packages (e.g. Trillium, Grape MMI) have been used to prototype interfaces. If so please specify the package(s) used.

b) one or more members involved in the design of the system (e.g. project manager, knowledge 'engineer', programmer) has particular responsibility for overseeing the development of the user interface. Please elaborate briefly on their role(s).

c) prototyping and user evaluation of the system with regard to the user interface.

Any further information :-

Move on to question 4

---

4. To what extent have you evaluated your system in respect to potential users? Please ring most appropriate.

a) evaluation has only been done within the group working on the design

b) the system has been shown to a potential user who is not directly involved in the design
(if so then specify how often and at what stages in the development) -

c) evaluation has been carried out with several users and at different stages in the development

Any further comments :-

General comments on this questionnaire :-

Thank-you for filling in the questionnaire. If you would like to discuss any of the points raised in it then please contact me at IHD.

Please return the completed questionnaire via internal mail to me at IHD.

APPENDIX 5

SOME IDEAS ON KNOWLEDGE REPRESENTATION FOR USER INTERFACE DESIGN GUIDELINES

18 / 11 / 1987

GUIDELINE = a static representation of knowledge

That is the problem as design is a 'dynamic' activity :- in applying guidelines one is applying a static piece of information to a very fluid problem (i.e. specific design instance).This leads to the problem of generality of guidelines.

If this problem is to be overcome then information, guidelines, have to in some way have the facility to be dynamic, i.e. can be manipulated, tailored, and can change to fit a specific problem.

Such a facility must be constrained, i.e. although a guideline can be tailored it must also maintain it's general structure and content otherwise it will change from being to general to being too specific.

POSSIBLE APPROPRIATE KNOWLEDGE REPRESENTATIONS

Although too early to say how the information might be represented here are some preliminary thoughts.

Two particular knowledge representations might be appropriate :-

1. PRODUCTION RULES
2. FRAMES

Why these two ?

1. PRODUCTION RULES

   If a computer-based tool is going to give guidance then it must perform two basic functions :-

   i. Identify user's problem.

   ii. Generate possible appropriate information (guidelines).

   It is in connection with the first of these functions (identification) that you might consider

   Production Rules. A simple scenario :-

   IF user inputs "MENUS"

   AND user inputs "NAIVE USERS"

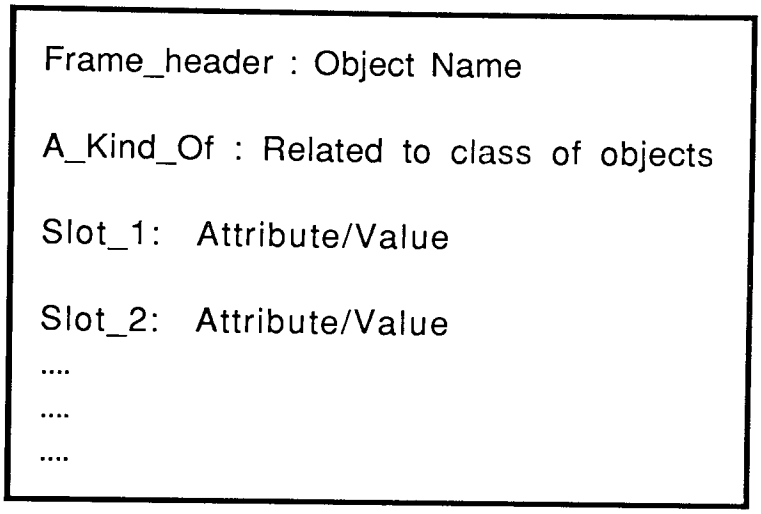
   THEN generate Guidelines G1, G2, G6 G9.....

Production rules might also be used to ascertain where in the development cycle a particular design is and then generate guidelines (possibly in the form of consequences of a particular implementation, e.g. use of menus etc). Whether this will work or not is largely dependent on whether guidelines can be found which pertain to particular stages in the design cycle.

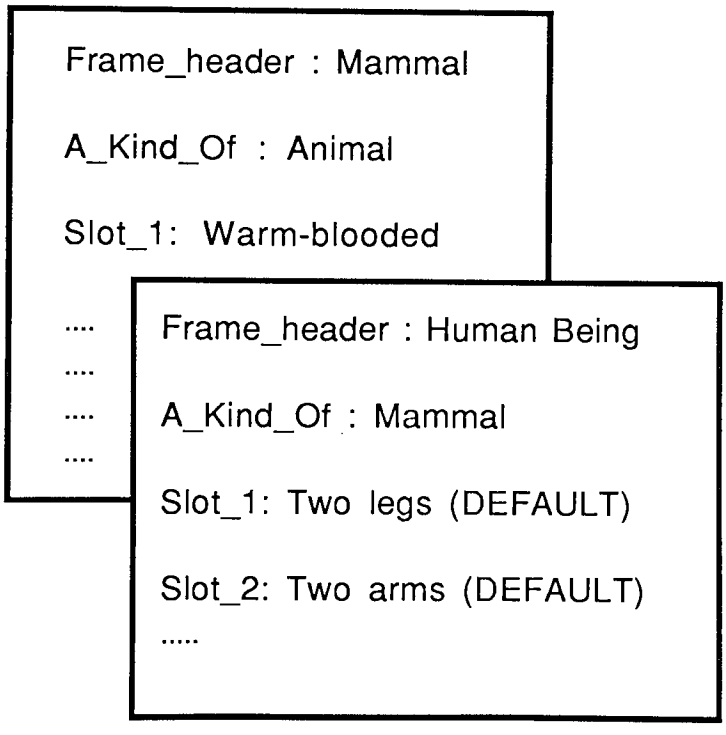
2. FRAMES

Of greater importance is to try and establish how one might represent a guideline, given the problem of generality / specificity mentioned earlier. It is here that one might consider the use of FRAMES as a potential KR. Frames have particular characteristics which might be appropriate.

Here is the basic architecture of a frame :-



Here is an instantiated example:-



Two user interface design guidelines represented as related frames:-

HEADER : MENU SELECTION

A_KIND_OF : DIALOGUE STYLE

RULE : Consider menu selection for tasks ...

EXAMPLE : (graphic)

EXPLANATION :
"Menu selection permits a user to specify control entries by pointing at displayed options or keying associated codes."

RELATED
GUIDELINES :

HEADER : SINGLE SELECTION PER MENU

A_KIND_OF : MENU SELECTION

RULE : "Each menu should permit only one selection by user"

EXAMPLE : (graphic)

EXPLANATION :

"Novice users will be confused by any more complicated procedure, such as a "Chinese menu" requiring one choice from Column A, one from Column B, etc."

RELATED
GUIDELINES:

(Guidelines from Smith and Mosier 1986; p 231)

INHERITANCE HIERARCHIES

A frame system comprises of many of these single frames. Such a system generates "inheritance hierarchies" whereby a frame in the system can inherit characteristics of a related frame further up in the hierarchy. In the example above 'Human Being' can inherit characteristics of the 'Mammal' (i.e a human is an animal and is warm-blooded) as it is related through the AKO slot.

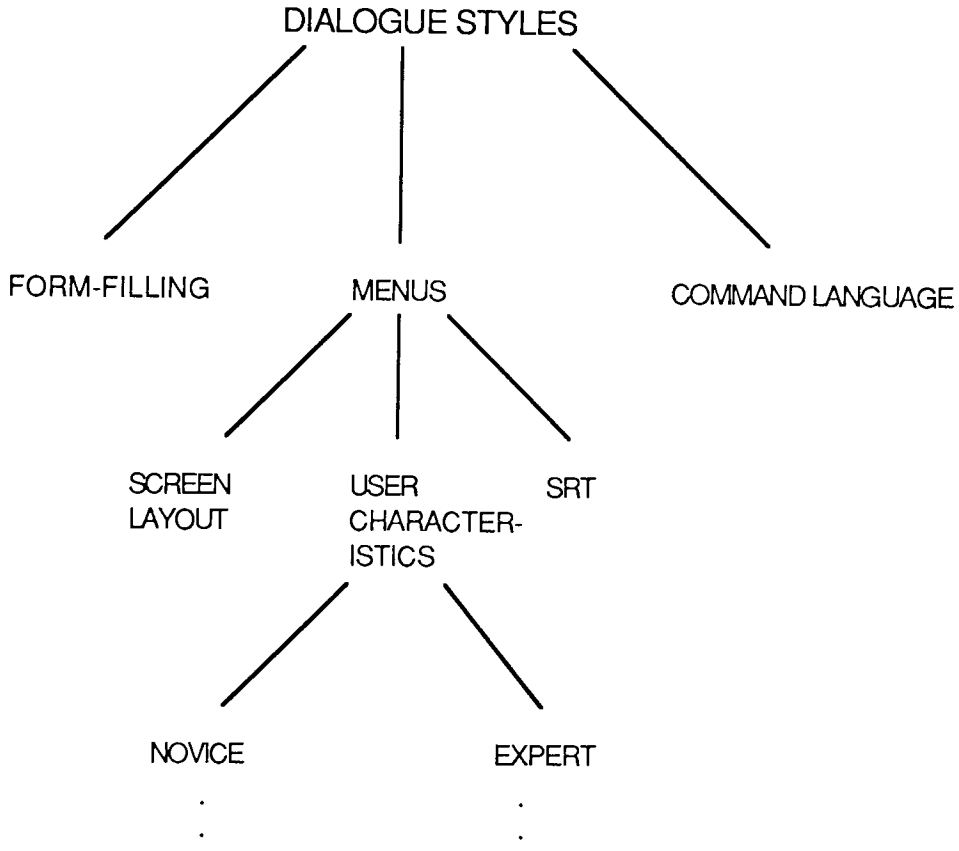
SLOTS
Each frame has a series of slots containing attributes, and/or values of attributes, pertaining to that particular frame. These slots can be fixed attributes (e.g. AKO slots, a human is always a mammal) or can be default attributes/values where a specific example of an object may vary in some of its attributes or values of attributes (e.g. a human may only have one leg, but he/she is still a human.

DEMONS
A value of a slot can be a computer procedure which if called can carry out actions on the frame system, i.e. it can add or remove pieces of information, change the structure etc. Such procedures are called "demons".

The characteristics listed above may suit some of the problems to be addressed with guidelines.

1. A set of guidelines might be represented as a network of related frames in a frame system. THE APPROPRIACY OF THIS MUST BE CHECKED BY LOOKING AT THE STRUCTURE OF PAPER-BASED GUIDELINES (Smith and Mosier as an example ?).

   To illustrate a possible hierarchy :-



   If such relationships can be built this may facilitate 'intelligent' cross-checking of information provided by a user, e.g., if the "NOVICE USER" frame is fired at some point in the interaction then this will automatically 'alert' other related frames, such as those on inappropriate dialogue styles for novice users.

2. Use of *Demons*

Demons could be used to facilitate the cross-checking just mentioned. They might also be used to permit the generation of new, tailored, guidelines based on a specific design situation. In effect a tailored guideline can be seem as being a statement of :-

WHAT A PARTICULAR (GENERAL) GUIDELINE MEANT IN TERMS OF OUR DESIGN

If a general guideline has, in part, been appropriate to a design then a new (more specific) guideline is generated. Appropriacy will be decided by the user (through dialogue with the system).

So the following situation would exist :-

General Guideline (GG)

MATCHES Specific Instance (SI)

GENERATES Tailored Guideline (TG)

This would leave a new tailored guideline (TG). However the original guideline (GG) must also be left in tact in the database, remaining as a template for future matches.

BIBLIOGRAPHY

Barr, A., Feigenbaum, E.A. (1981) *The Handbook of Artificial Intelligence.* Volume One. Pitman.

Rich, E. (1983) *Artificial Intelligence.* McGraw-Hill.

Smith, S.L., Mosier, J.N. (1986) *Guidelines for Designing User Interface Software.* MITRE Corporation Report No. 10090. Bedford, Massachussetts, USA.

Winston, P.H., Horn, B.K.P. (1981) *LISP.* Addison-Wesley.

Christie, B., Gardiner, M.M. (1987) *Future Directions.* Chapter Ten in 'Applying Cognitive Psychology to User-Interface Design. Gardiner, M.M., Christie, B. (eds). John Wiley & Sons

The British Coal specification for an advisory KBS: an initial analysis of user interface requirements

18th February 1988

Function Table and associated inputs and outputs from initial specification for an advisory KBS for fire detection and control in mines. Provided by British Coal.

FUNCTION	INPUT	OUTPUT	USER	OTHER
ANALYSE	✓	✓	✓	✓
HYPOTHESISE	✓*	✓	✓	✓
ADVISE (on actions to locate 'heating', or open fire)		✓	✓	
ADVISE (on emmergency procedures)		✓	✓	
DETERMINE SAFEST ROUTE	✓ (specify locations)		✓	

* Needs to be clarified whether HYPOTHESISE requires an input which is separate from informatin provided by ADVISE function.
Present assumption is :- That the user inputs discrete information on which a hypothesis is then based.

Heating analysis module

1. List of functions
    i. ANALYSE (symptoms, locations and causes)
    ii. HYPOTHESISE (existing or possible heatings)
    iii. ADVISE (a) (actions to locate heating or open fire)
    iv. ADVISE (b) (on emmergency procedures)
    v. DETERMINE SAFEST ROUTE

1.1. Identify input associated functions
    i. ANALYSE
    ii. HYPOTHESISE
    iii. DETERMINE SAFEST ROUTE

1.1.1.Identify user associated inputs

i. ANALYSE
ii. HYPOTHESISE
iii. DETERMINE SAFEST ROUTE

The following questions are now asked to try and determine how the user will be prompted for required input for each of these functions.

a) **What is the nature of the information required by the system to facilitate a function?**
Is it - single / multiple item ?
numerical / textual ?
co-ordinates (as in case of pointing) ?

b) **What is the information required by the user to make an input?**
How does he/she know they are expected to, or can, make an input ?

c) **Is the function accessible at any point in the dialogue, or is it specific to certain points in the dialogue ?**

1.1.2. Identify 'other source' associated inputs

i. ANALYSE
ii. HYPOTHESISE *
iii. DETERMINE SAFEST ROUTE (user inputs specified locations)

* at present it isn't clear whether HYPOTHESISE will requrie a separate input from that in ADVISE

1.2. Identify output associated functions

i. HYPOTHESISE
ii. ADVISE (a)
iii. ADVISE (b)
iv. DETERMINE SAFEST ROUTE

1.2.1. Identify user associated outputs

i. HYPOTHESISE
ii. ADVISE (a)
iii. ADVISE (b)
iv. DETERMINE SAFEST ROUTE

At this point the system designer is again expected to answer particular questions concerning the nature of the output in order to assess the best method for representing it to the user.

As was mentioned in the detailed outline there is a distinction between output that is solicited by the user and that which is produced by the system, unsolicited.

Not having enough information it is currently not possible to distinguish the solicited and unsolicited outputs in the heating analysis module.

The following questions are then asked :-

a) **What format does the output take ? (i.e. textual, graphical, etc ?)**

b) **What clarification or explanation of an output is likely to be required? (e.g. clarify use of terminology)**

This is particularly important in the 'Heating analysis module' where the operator can make enquiries about the system's recommendations.

c) **If solicited by the user, then by what method does the user solicit? (e.g. via menu options)**

1.2.2. Identify 'other source' associated outputs
(In this example no such outputs exist)

2. Describe 'user environment'

No description exists to be able to apply the associated questions. See the earlier outline.

The specification for ITAM from the *Intellipse* project: an initial analysis of user interface requirements

2nd February 1988

Function Table and associated inputs and outputs from initial specification of ITAM; a KBS module for monitoring the performance of a TOTAL database.

FUNCTION	INPUT	OUTPUT	USER	OTHER
EDIT	✓		✓	
FILE WARNING	✓ (selection of file)	✓ (warning and explanation)	✓	
FILE TRACE	✓ (selection of file)	✓ (display file history)	✓	
ANALYSE STATISTICS				✓ (statistics via floppy disks)
REPORT		✓ (warnings)	✓ (hard-copy)	
EXTRACT HISTORICAL DATA		✓		
TREND ANALYSIS	✓ (from statistics)	✓ (warnings)	✓ *(output)*	✓ *(input)*

ITAM - Monitor performance module

1. List of functions

   i. EDIT (threshold data)
   ii. FILE WARNING (specific file)
   iii. FILE TRACE (historical data for a file)
   iv. ANALYSE STATISTICS
   v. REPORT (on analysis)
   vi. EXTRACT HISTORICAL DATA (on each file)
   vii. TREND ANALYSIS (to provide warning if files exceed)

1.1. Identify input associated functions

i. EDIT
ii. FILE WARNING
iii. FILE TRACE
iv. ANALYSE STATISTICS
v. TREND ANALYSIS

1.1.1. Identify user associated inputs

i. EDIT
ii. FILE WARNING (selection of a file)
iii. FILE TRACE (selection of a file)

It is now necessary to try and establish how the user will be prompted to make the required input for each of these functions. Questions will be asked of the system designer concerning the factors appropriate to input prompts. The questions seen as relevant (they are expected to be elaborated upon as the method is tried out) are as follows:-

a) **What is the nature of the information required by the system to facilitate a function?**
Is it - single / multiple item ?
numerical / textual ?
co-ordinates (as in case of pointing) ?

b) **What is the information required by the user to make an input?**
How does he/she know they are expected to, or can make an input ?

c) **Is the function accessible at any point in the dialogue, or is it specific to certain points in the dialogue ?**

1.1.2. Identify 'other source' associated inputs

i. ANALYSE STATISTICS
ii. TREND ANALYSIS

1.2. Identify output associated functions

i. FILE WARNING
ii. FILE TRACE
iii. REPORT
iv. EXTRACT HISTORICAL DATA
v. TREND ANALYSIS

1.2.1. Identify user associated outputs

i. FILE WARNING (warning and explanation)
ii. FILE TRACE (display 'history' of a file)
iii. REPORT (warnings)
iv. TREND ANALYSIS (warnings)

At this point the system designer is again expected to answer particular questions concerning the nature of the output in order to assess the best method for representing it to the user.

As was mentioned in the detailed outline there is a distinction between output that is solicited by the user and that which is produced by the system, unsolicited.

In this example *user solicited* outputs are :-
i. FILE WARNING (the user selects a particular file)
ii. FILE TRACE (the user selects a particular file)

Unsolicited outputs are :-
i. REPORT (produced as hard-copy print-out)
ii. TREND ANALYSIS (warnings)

The following questions are then asked :-

a) **What format does the output take ? (i.e. textual, graphical, etc ?)**

b) **What clarification or explanation of an output is likely to be required ? (e.g. clarify use of terminology)**

In ITAM the FILE WARNING function is accompanied by explanations of the warnings.

c) **If solicited by the user,then by what method does the user solicit ? (e.g. menu options)**

1.2.2. Identify 'other source' associated outputs

(In ITAM no such outputs exist)

2. Describe 'user environment'

As yet there exists no written description of the user environment for ITAM. Therefore not much can be said concerning the user environment. Only some of the questions described in the detailed outline can be answered.

2.1. Identify user types

All that exists is a notion that the system will be used by :-

a) Database administrator (DBA)
b) an assistant DBA

From this description we can establish that in the case of user (a) that he has a high level of expertise in the task domain. In the case of the assistant DBA (b) it is inferred that there does not exist such a high level of expertise.
This suggests the need for varying levels of help and clarification (explanations of warnings).

2.2. Identify mode of use

i. Are the users expected to learn explicitly about the task from the system ?

- No explicit training is envisaged from the system

ii. Will the system be used regularly or occasionally ?

- The system will be used regularly, at least weekly.

APPENDIX 7

Report on IBS Project
User Issues

Clive Bright
18.10.1988. Aston University

This report describes some recommendations on an approach to establishing user requirements for IBS. This would take place as part of the overall project development. These recommendations are based on my perspective of the IBS project at present and relate to user interface design issues within expert systems development in general.

Present state of IBS

An initial concept demonstrator system representing some of the ideas behind IBS has been implemented using the development software KEE. This demonstrator has been shown at a 'Credit Scoring' conference in September

The design meetings which I have personally been involved in have focussed on issues of the user interface. During these meetings we have discussed interface features which we thought might be desirable in IBS. Having generated initial ideas for interface requirements, amongst ourselves, it is now important to meet with potential end-users of IBS (i.e. Managers, Statisticians). This is crucial in developing a clearer specification of the system requirements.

Recommendations

Overview

There are two main activities involved in the development of this project :-

1. Construction of the knowledge base

   Conducting knowledge engineering with existing experts in the domain (eliciting rules, business models etc). followed by representation of this information in a computable knowledge form (e.g. building a knowledge base in KEE, or other tool).

2. Capturing IBS users' requirements

   This involves clearly defining the intended users of IBS. Establishing the requirements that users have of the system at two levels :-

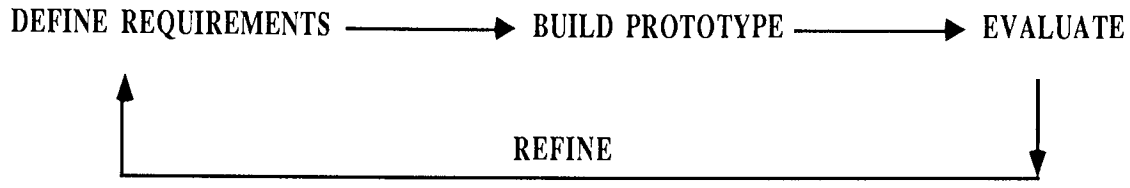
   a) Functional requirements

      "What information do they need to support their tasks?"

   b) Interface requirements

      "How is this information (a) represented to them?"

These two activities will feed into each other. Having established the initial 'first-pass' specification for 2a the process of knowledge acquisition with the expert(s) then begins where the knowledge needed to meet the requirements is obtained. At this point the two activities should continue, both adopting something resembling the iterative cycle described below :-



The approach suggested by this cycle should be conducted in both :-

i.) the activity of building and testing the knowledge base; where evaluation will involve the domain expert(s);

and

ii) in capturing user requirements; where evaluation will involve representative users. *The need to evaluate prototypes with users cannot be overstressed, especially given the size and importance of a project like IBS.*

Methods for capturing user requirements

Concentrating on the approach to capturing user requirements I shall elaborate on each of the activities described in the cycle above.

1. Define user requirements

   Initially it will be important to develop a clear picture and understanding of the intended users (statisticians and managers) and their current tasks, practices and knowledge relevant to IBS. The first stage in this process should be to meet and interview representative users to establish such a picture.

Information concerning the following issues should be obtained, as such information will be important in developing the requirements specification :-

Information needed	**Reasons for information**
Present tasks* of user	To establish system functions which will support (fit in with) present tasks. To establish where IBS can improve the existing situation.
Present information used to carry out present tasks (including, existing support tools)	To establish what the information "looks like" (i.e. graphical? tabular? etc) and hence the types of representation needed in the system. To establish the source of information. To see where more information is required and can be provided by IBS.
Variations in levels of user knowledge (in terms of domain and general computer use)	To establish the type(s) of dialogue and on-line support (i.e, help, explanations) required.
Frequency of tasks	Estimated frequency functions of IBS will help indicate the degree of 'prompting' or reminding on how to access functions of the system

* in the context of IBS it is likely that 'tasks' will commonly mean 'types of decisions made'.

---

2. Prototyping

User interface prototyping should have two roles in the project.

i. Following on from the initial information gathering with users, a prototype should be built to act as a concrete vehicle for further clarifying and developing the functional requirements of the system. It should demonstrate what information it is going to provide the user, and hence what tasks it will support. Evaluative feedback on amendments can then be obtained from users. Such a prototype does not require any underlying knowledge base.

(On a more specific note)

From my own experience in another project, a very useful tool for constructing prototypes for this purpose has been Hypercard (previously demonstrated at a meeting). Whilst Hypercard only runs of Macintosh computers, it does have several benefits over other packages (such as KEE).

Advantages of the use of Hypercard as a user interface prototyping tool :-

- it allows you to develop screens, and navigation between screens very quickly. Changes, following comments from users, can be made very rapidly.
- the Macintosh is very transportable and so can be taken to users for demonstrations.
- there is already some considerable experience, within another part of TSB Manchester, in the use of Hypercard.

NB. I should stress that Hypercard would not be appropriate for constructing the knowledge base, or building any serious functionality. It's use would be purely as a user interface prototyping tool, to gather user requirements.

ii. The second role of the user interface prototype is to establish the physical interface requirements (screen layouts, use of menus etc). This stage of development would be reached when a fairly clear picture of the overall system requirements has been obtained.This 'picture' will have resulted from both interactions with the expert (knowledge engineering and prototype knowledge base evaluation) and from the initial phase of the user-centred requirements capture process (above).

Having captured the initial system requirements, user interface prototyping should concentrate at a deeper level to help establish the style of dialogue between the user and the system. Here a prototype should be used in evaluation with users to determine the best forms for dialogue. By dialogue I mean elements such as :-

- representation of inputs and outputs

  covering

  data entry (form filling? pointing with mouse ? etc)

  option selection (menus? command language? natural language ? etc)

  display of data (graphical? tabular ? how much information displayed at a time? etc).

Continued overleaf

- help facilities required
  covering
  on-line help about general system, and contextual help at points in the application.
- explanations required (for example, explanations of underlying business model)

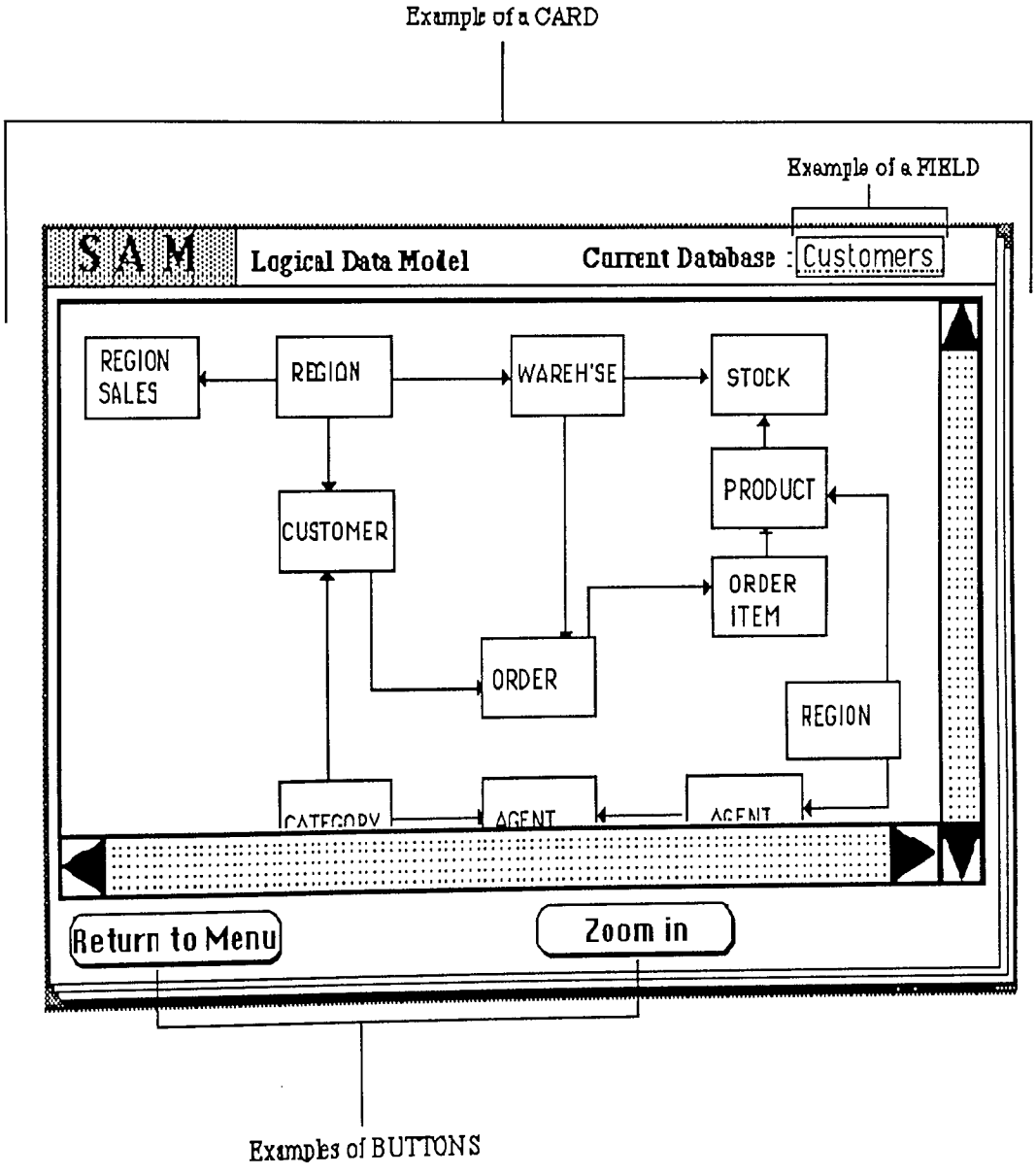
3. Evaluation

Given the potential investment in the IBS project it is crucial to have a set of representative users who can be called upon to evaluate the prototypes at stages in development. Feedback from these users will help ensure that the system is appropriate to the requirements of the organisation. Such evaluation will incur some extra cost (in terms of development time, and user's time) at the early stages of design. However, if users are exposed to the system too late then the the costs incurred from having to make any major changes at that stage will be greater. There are a number of cases of knowledge based systems which have failed to be used because they have inadequately captured the user's requirements. Often this has occurred because design and evaluation has centred around the expert and the construction of the knowledge base, rather than also considering end-users. Obviously, we want to avoid this mistake in the development of IBS.

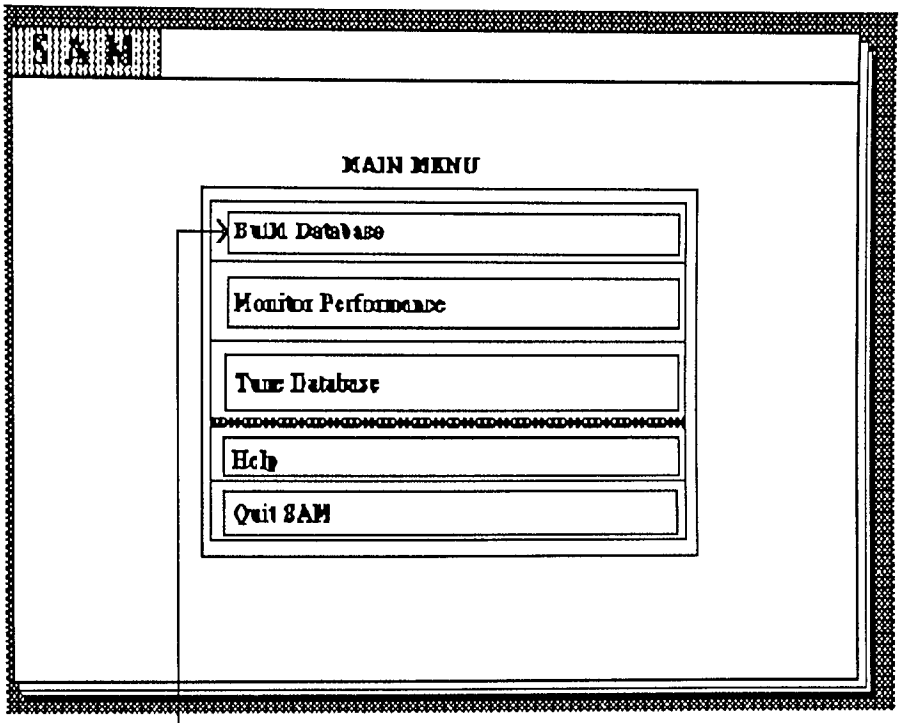
APPENDIX EIGHT

SOME NOTES ON HYPERCARD AND ITS USE IN THE USER INTERFACE PROTOTYPING OF *SAM*

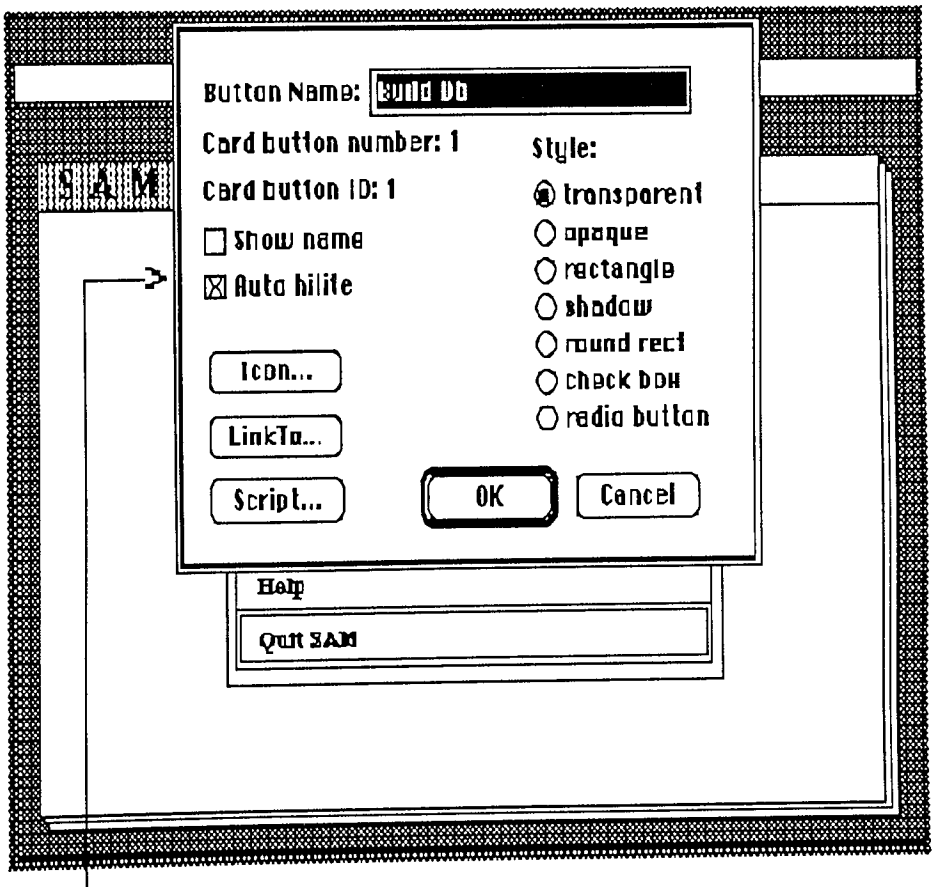
Examples of the use of Card, Field and Button objects from HyperCard: as used in the SAM prototype



Buttons



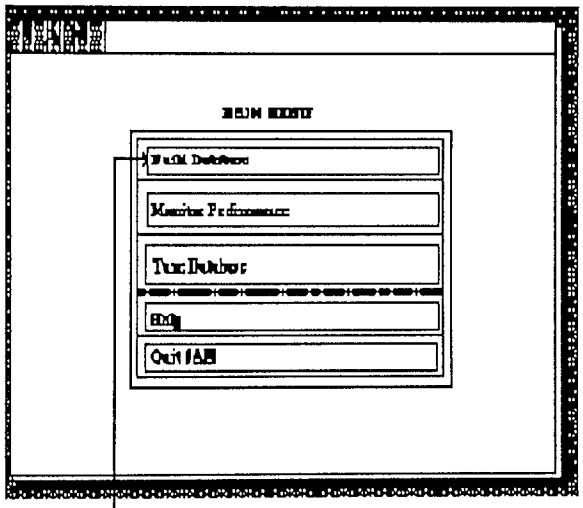
Button name: Build Db



Characteristics of Button 'Build Db'

:Transparent
:Auto hilite - i.e. is highlighted when mouse selected

Scripts



'Script' for button 'Build Db'
i.e. Instructions activated upon
mouse selection of button area

```
on mouseDown
  ask "Enter the name of the database, please:"
  if it = "" then
    go to this card
  else
    set lockScreen to true
    repeat with x = 2 to the number of cards
      go to next card
      put it into field 1
    end repeat
  end if
end mouseDown

on mouseUp
  go to card 2
  set lockScreen to false
end mouseUp
```

When area of this button is clicked upon

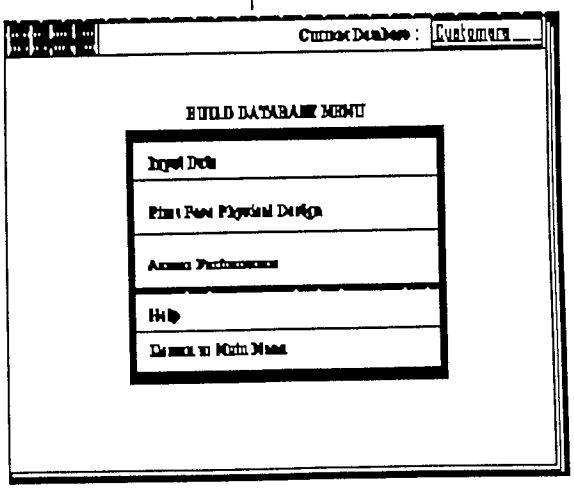
Generate Dialogue box requesting name of database

Entered name of database becomes bound to variable 'it'

Value of 'it' is placed in specified field of other cards (screens) in the stack, i.e. name of database is created as a header on rest of the cards.

On release of mouse click

move to the next card in the stack



Here are three example of scripts written in the construction of *SAM*. They include some features of the HyperTalk scripting language which were particularly useful.

**** *** denotes comments and not actual code*

---

1. - GENERATING A DIALOGUE BOX USING "ask"
   - BINDING A VALUE TO VARIABLE 'it'
   - USING 'repeat for number of cards in stack' to send data to fields

```
on mouseDown
 ask "Enter the name of the database, please:"
 if it = "" then
  go to this card
 else
  set lockScreen to true
  repeat with x = 2 to the number of cards
   go to next card
   put it into field 1
  end repeat
 end if
end mouseDown
```

---

2. USING RESOURCE 'PopUpMenu' to display options in a pop-up menu and to then retrieve the user's selection

```
on mouseDown
 get PopUpMenu ("Access Point,Relation Box",0,250,350)

              *** display options (Access point etc) in a pop-up
              menu at a specified screen location ***
               *** do not highlight either option (0) *** cont'd overleaf
 put it into target
              *** bind the option selected (which will be a
              numerical value 1, or 2) to variable 'target' ***
 set lockScreen to true
 get item 1 of field id 29
 put it into temp
 repeat with x = 1 to the number of cards
  put temp into item 1 of background field 2
 end repeat
 if target = 1 then
 *** if option selected was 'Access Point' then.. ***
```

```
ask "Enter name of Access Point, please:"
        *** display a dialogue box (using 'ask') which asks user
       for name of access point ***
put it into target
*** assigns name entered to variable 'target' ***
set lockScreen to true
go to card id 3728
set name of button 1 to target
            *** puts name of access point into button which is
            displayed on a related card ***
set lockScreen to false
end if
set lockScreen to false
end mouseDown
```

---

3. PARTIAL SCRIPT FOR A BUTTON WHICH WHEN ACTIVATED, ITSELF CREATES A BUTTON AT A SPECIFIED LOCATION. IT ALSO BINDS A SCRIPT TO THAT NEW BUTTON

```
on mouseDown
  ...
  set lockScreen to true
        *** Until further notice keep the screen static whilst
        carrying out the following instructions ***
  doMenu "New Button"
        *** create a new button ***
  set style of button "New Button" to rectangle
  set name of "New Button" to target
        *** set the various characteristics of this new button **
  show button target at 265,148
        *** specifiy the screen location  ***
  set the script of button target to empty
        *** ensure that the script of this newly created button is
        empty ***
put "on mouseUp" & return & "beep 4" & return & "end
mouseUp" into scriptHolder
        *** assign these instructions to variable 'scriptHolder
        (note: inclusion of specified format, i.e., carriage returns)
        ***
  set script of button target to scriptHolder
        *** assign value of variable scriptHolder to be the script
        of new button***
  set lockScreen to false *** make the screen 'active' once again
***
... end mouseDown
```

Bibliography

Apple Computer, Inc (1987) — *HyperCard User's Guide.* Apple Computer Inc.

Goodman, D. (1987) — *The Complete HyperCard Handbook.* Bantam Books. *

*Particularly good guide to HyperTalk scripting

APPENDIX 9

A Set of Screens from the SAM (Supra Advisory Module) User interface Prototype; SAM a KBS for the Design and Performance Monitoring of large Databases.

This appendix contains screen 'snapshots' from the user interface prototype from the SAM module developed as part of the *Intellipse* project.

INSTRUCTIONS:
As far as possible the various sequences (i.e. the navigation between the various screens) within the system are indicated here. For this purpose, each screen has a number in its bottom, right-hand corner. Also, the selection of a particular menu option, or some other action which leads to a further related screen, is indicated with an X beside the appropriate location on the screen. (Selection in the prototype was made via a mouse 'click'.)

The key below indicates the various navigation scenarios which are then shown in the series of actual screens overleaf:-

KEY

Sequence A: Screens 1 to 4

1. Select Option : *Build Database*
2. Select Option: *Input Data*
3. Select Option: *Enter Data Relations*
4. Displayed: *Form for entering information about a single Data relation*

Sequence B: Screens 5 to 10

5. Select Option : *Enter Access Profiles*

6. Click on : *Create*
(leading to Dialogue 'Box" which requests the name of a profile ('Cust.1.), and the access point name which in this example is 'cust.number' - not shown).

7. Click on: *'cust.number' lozenge*

8. a) Displayed: *Dialogue Box for entering information on a related data relation*
b) Click on: *Select*

9. Displayed: *Further element in profile*
Click on: *'cust.order' box*
(leading to Dialogue Box as in 8.a., above - not shown)

10. Displayed: *Completed Access Profile*

Sequence C: Screens 11 to 12

11. Select Option: *Enter File Volumes*

12. Displayed: *Form for entering information about each file (its capacity ,etc)*

Sequence D: Screens 13 to 15

13. Select Option: *View Logical Data Model*

14. a) Displayed: *Logical Data Model diagram (part of)*
b) Click on: *Zoom in*

15. Displayed: *Magnified view of diagram*

Sequence E: Screens 16 to 17.

16. Select Option: *Validate Existing Data*

17. Displayed: *Data entry error report*

Sequence F: Screens 18 to 21

18. Select Option: *First Pass Physical Design*

19. Select Option: *View Physical Design*

20. a) Displayed: *View of Physical Design*
b) Click on: *Scroll down arrow*

21. Displayed: *View of Physical Design (shifted down)*

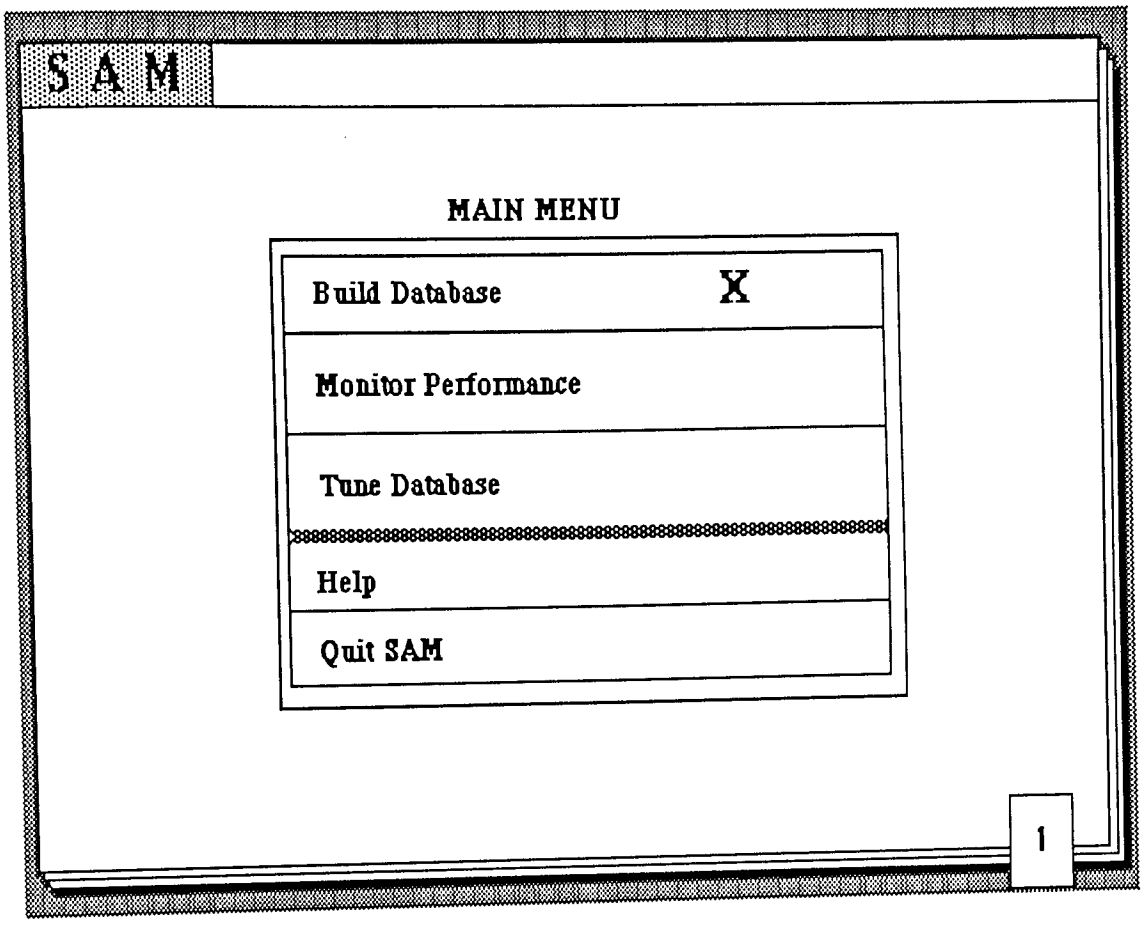
Sequence F: Screens 22 to 24

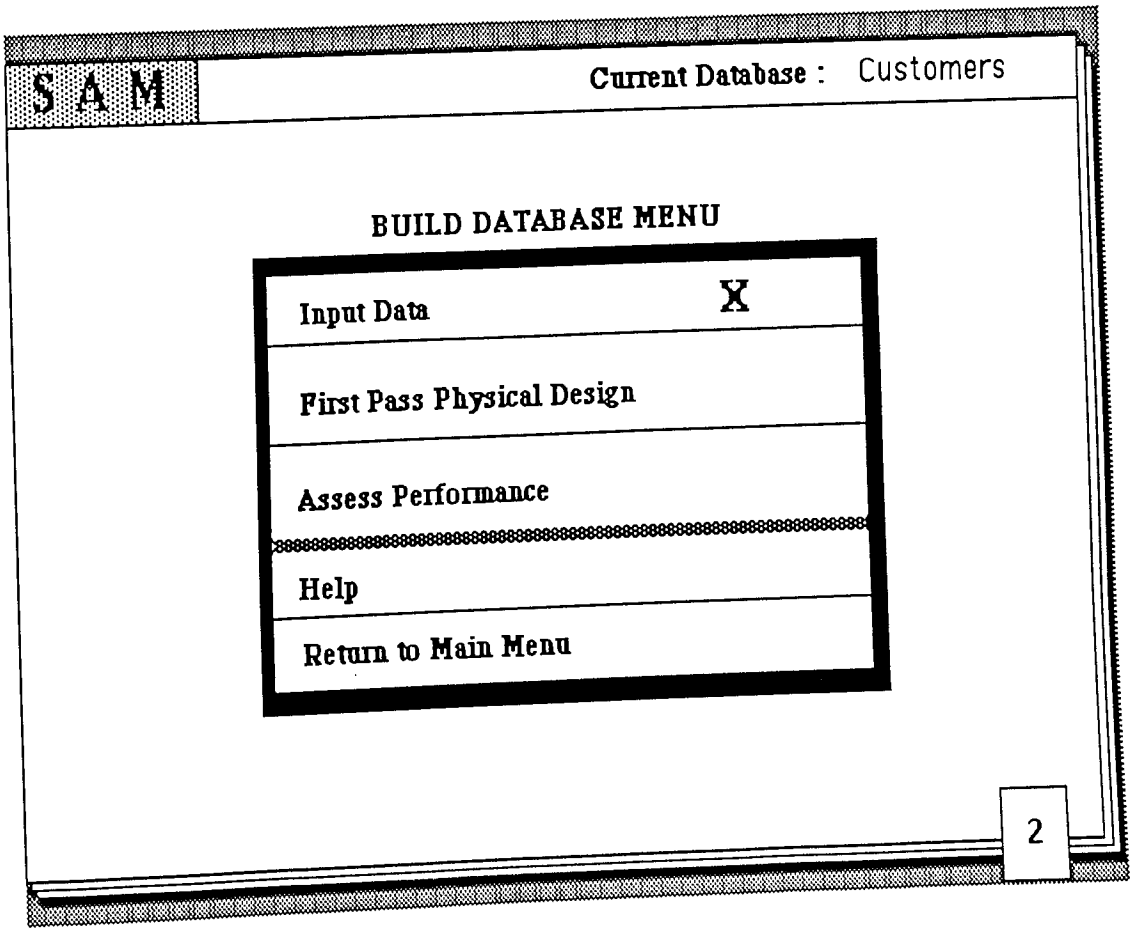
22.	Select Option:	*View File Parameters*
23.	a) Displayed: b) Click on:	*Form displaying calculated File Parameter values* *'E' circle*
24.	Displayed:	*Explanation pertaining to the calculation of Physical record lengths - explanation in example is inaccurate as it stands; used for demonstration only*

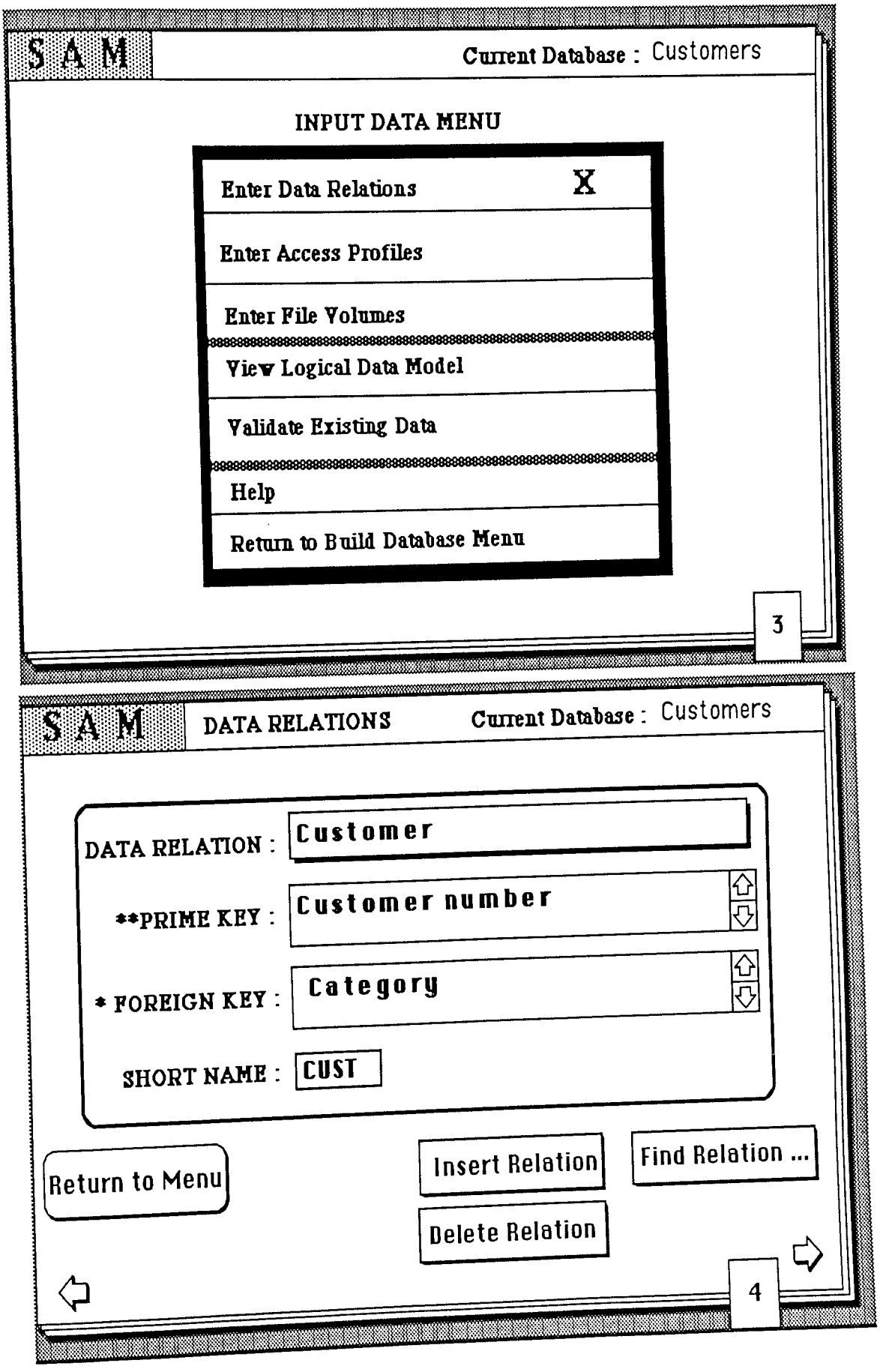
Sequence G: Screens 25 to 26

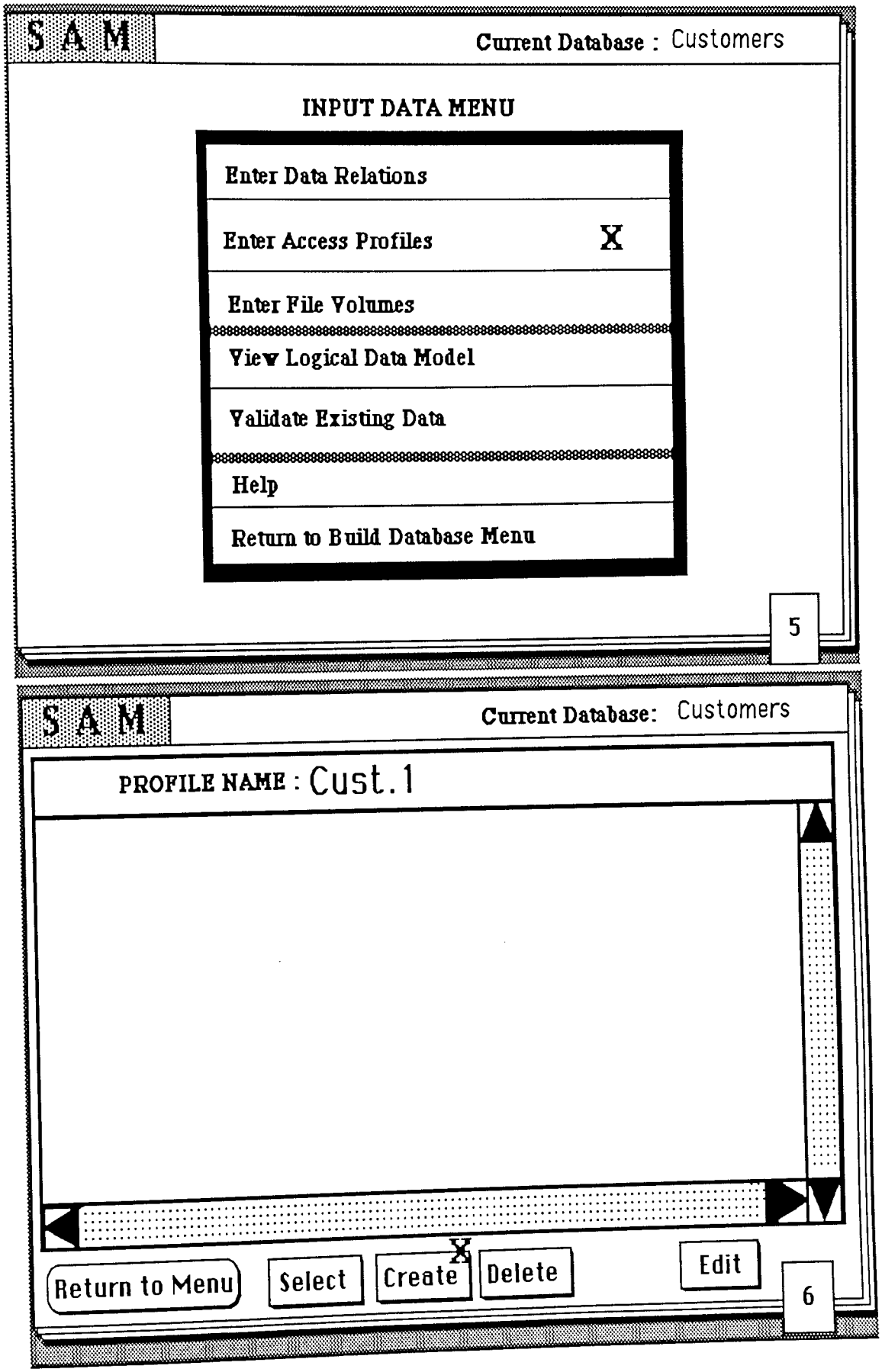
25.	Select Option:	*Assess Performance*
26.	Displayed:	*Calculated Performance of existing physical design of database*

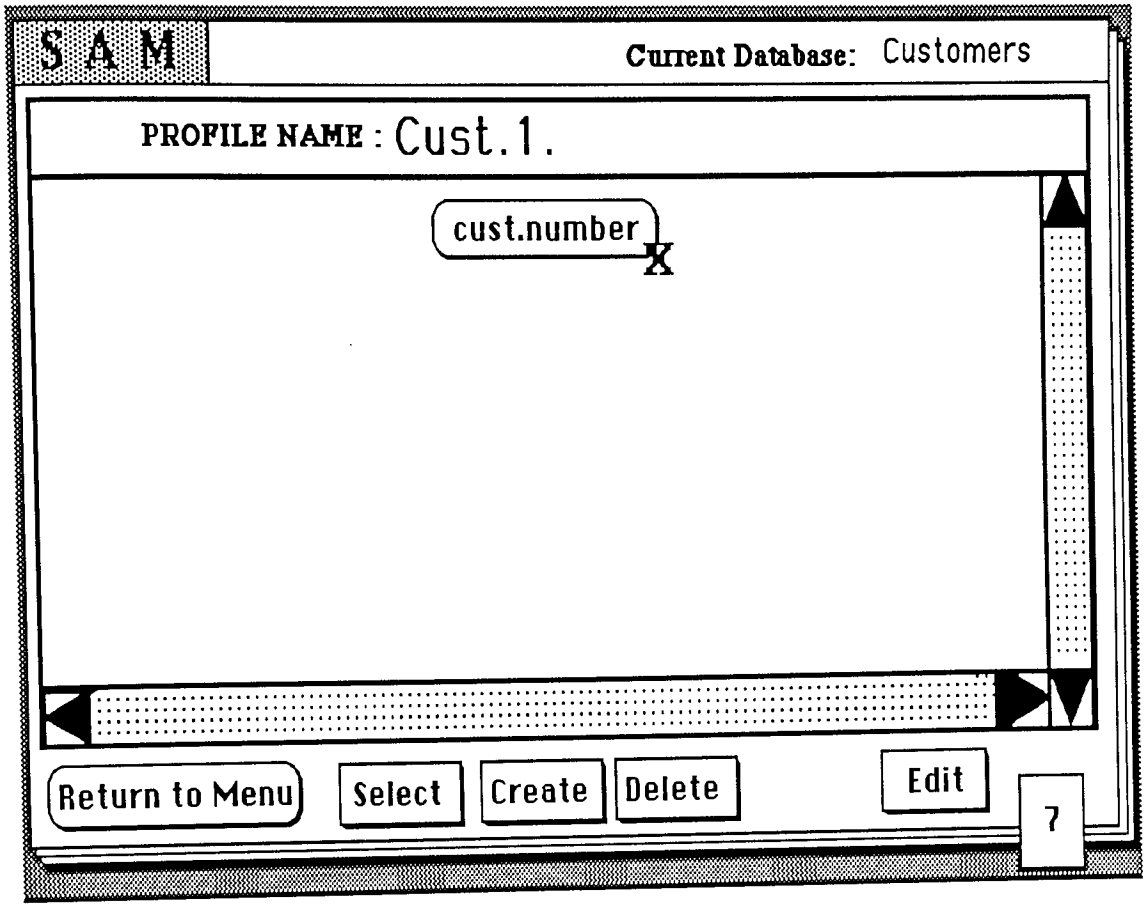
Continued

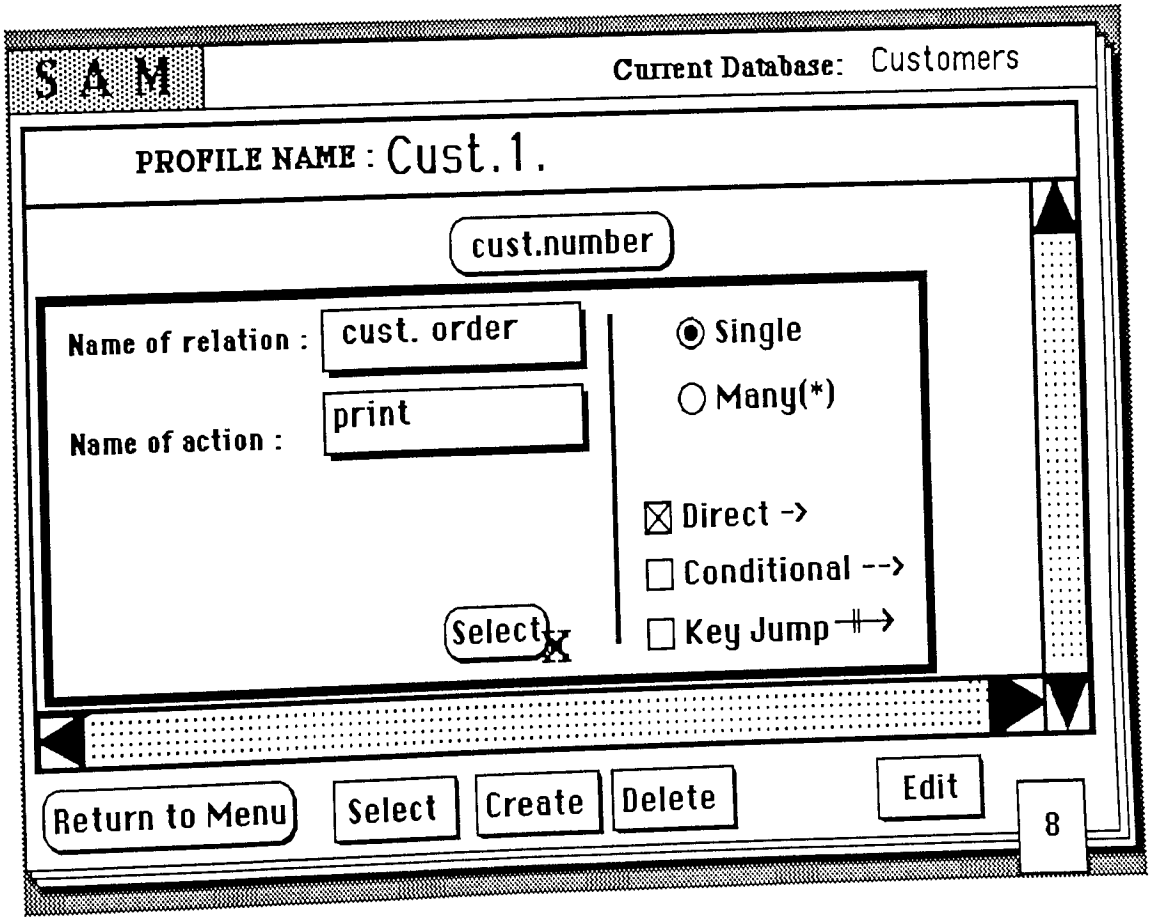

S A M
MAIN MENU
Build Database X
Monitor Performance
Tune Database
Help
Quit SAM
1

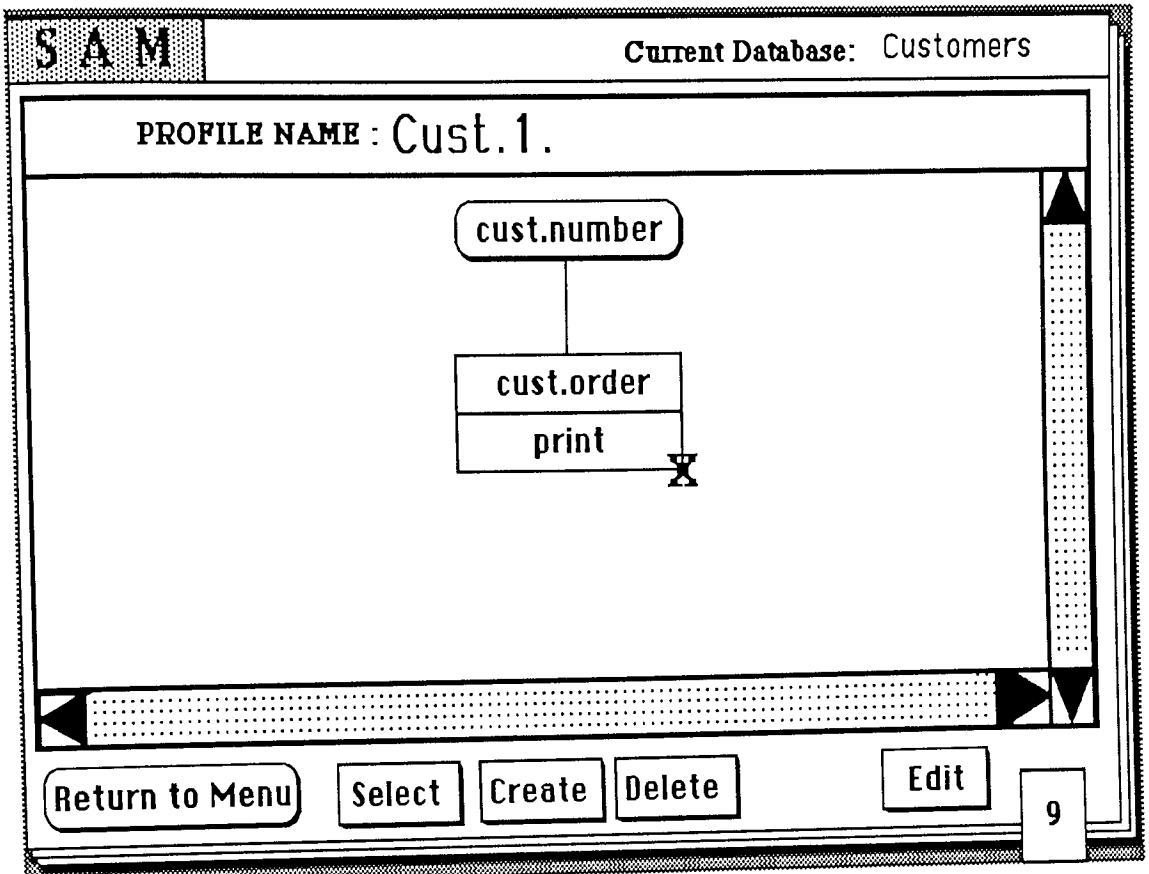

S A M
Current Database : Customers
BUILD DATABASE MENU
Input Data X
First Pass Physical Design
Assess Performance
Help
Return to Main Menu
2

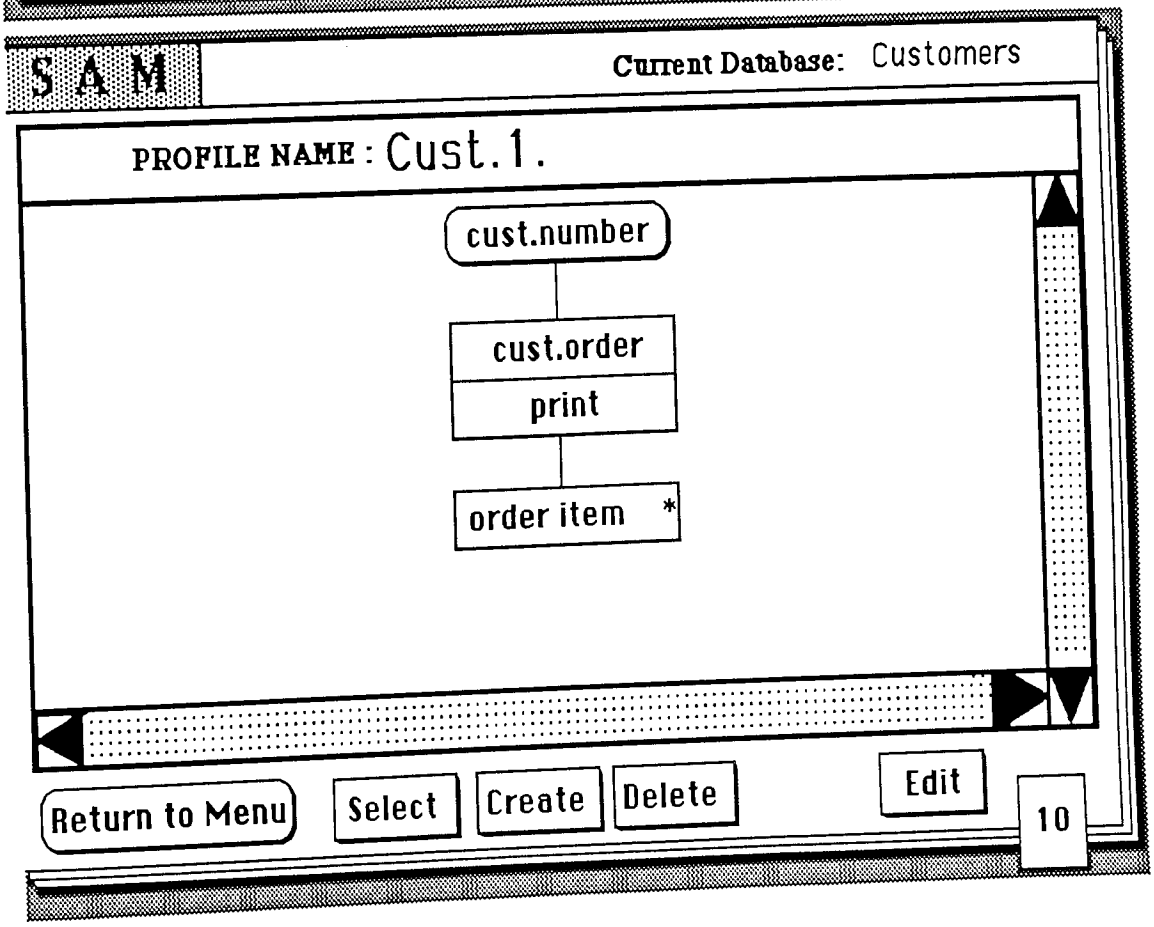

S A M
Current Database : Customers
INPUT DATA MENU
Enter Data Relations
X
Enter Access Profiles
Enter File Volumes
View Logical Data Model
Validate Existing Data
Help
Return to Build Database Menu
3
S A M
DATA RELATIONS
Current Database : Customers
DATA RELATION :
Customer
**PRIME KEY :
Customer number
* FOREIGN KEY :
Category
SHORT NAME :
CUST
Return to Menu
Insert Relation
Find Relation ...
Delete Relation
4

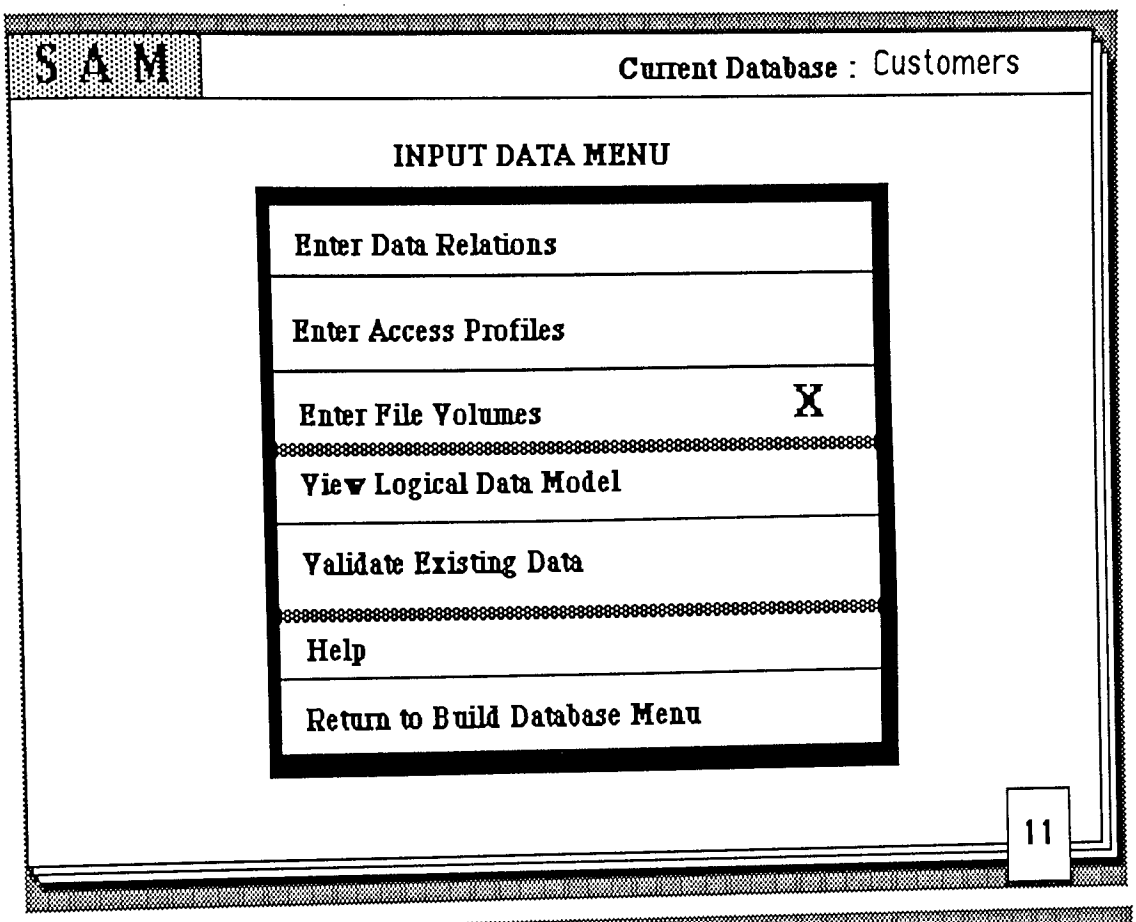

S A M
Current Database : Customers
INPUT DATA MENU
Enter Data Relations
Enter Access Profiles
Enter File Volumes
View Logical Data Model
Validate Existing Data
Help
Return to Build Database Menu
5
S A M
Current Database: Customers
PROFILE NAME : Cust.1
Return to Menu
Select
Create
Delete
Edit
6

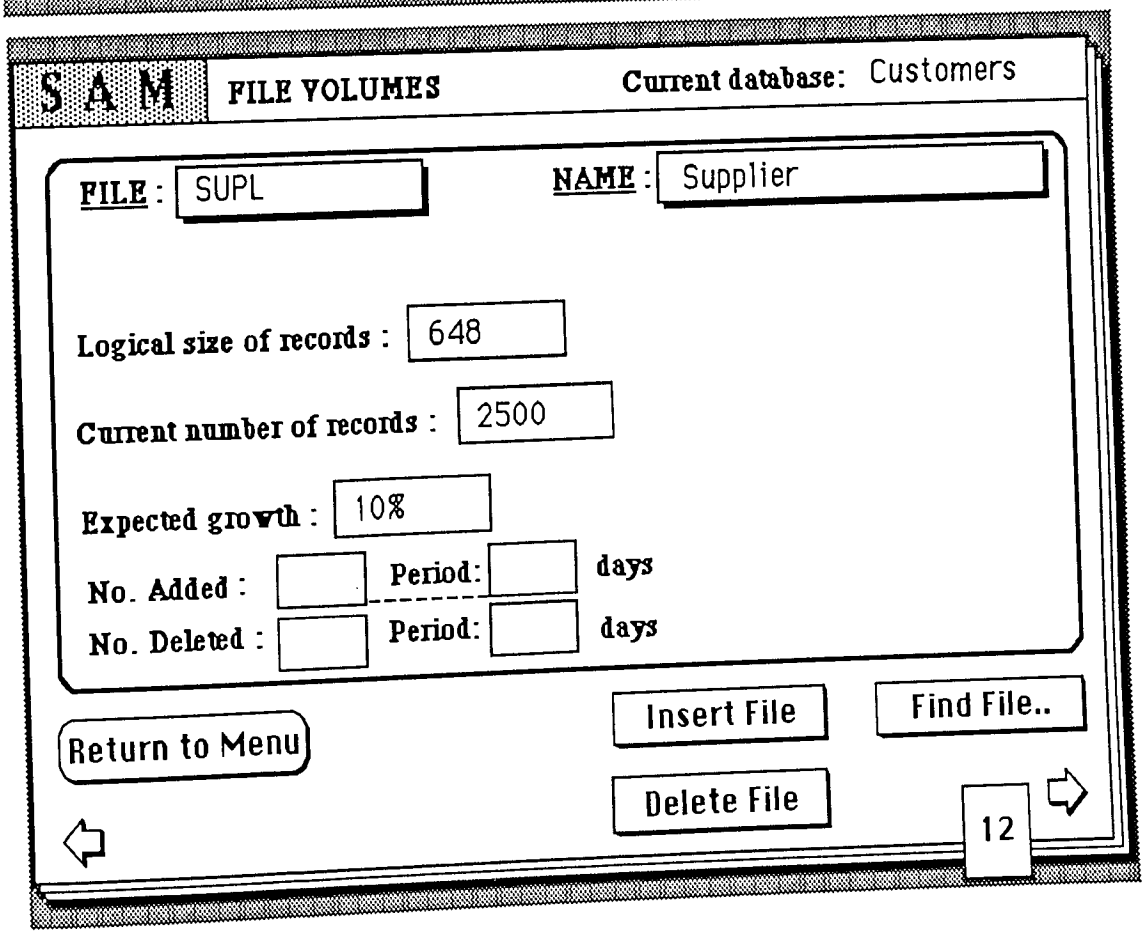

S A M
Current Database: Customers
PROFILE NAME : Cust.1.
cust.number
Return to Menu
Select
Create
Delete
Edit
7

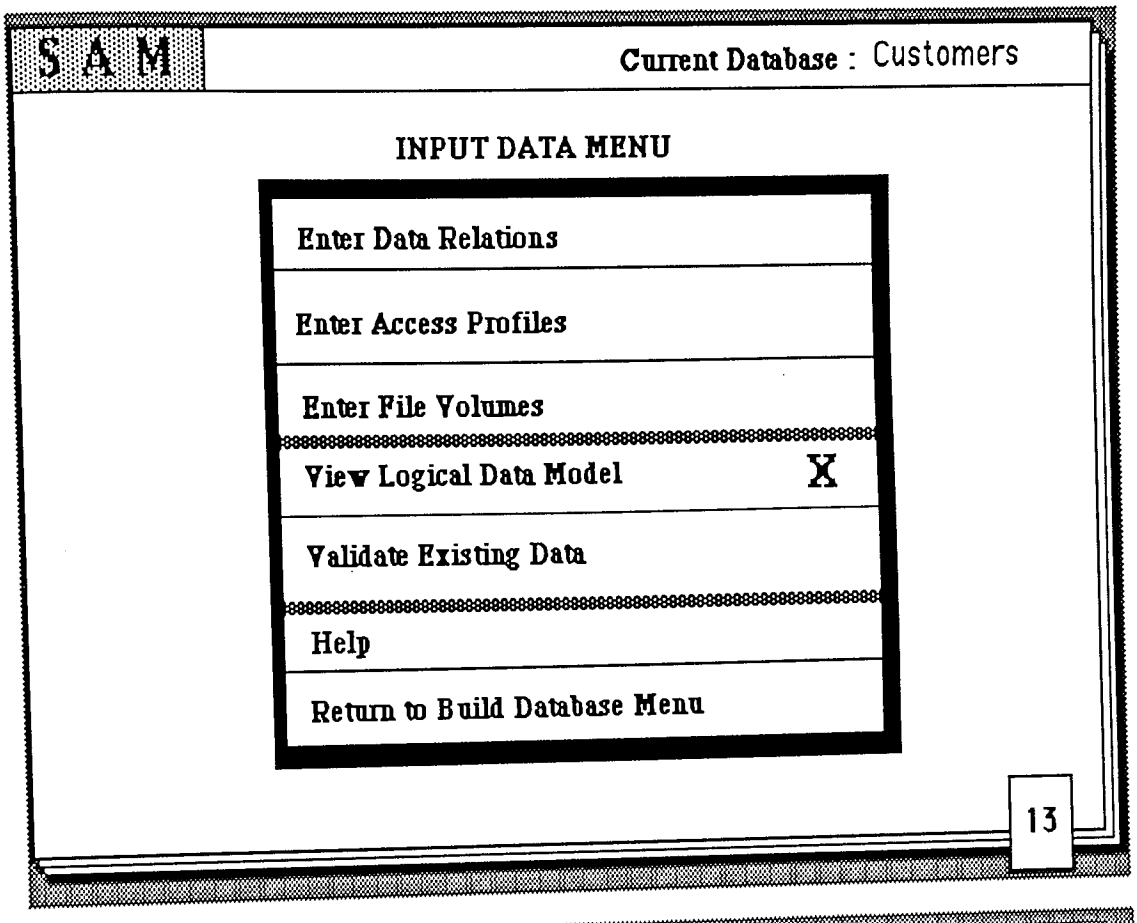

S A M
Current Database: Customers
PROFILE NAME : Cust.1.
cust.number
Name of relation :
cust. order
Name of action :
print
Single
Many(*)
Direct ->
Conditional -->
Key Jump
Select
Return to Menu
Select
Create
Delete
Edit
8

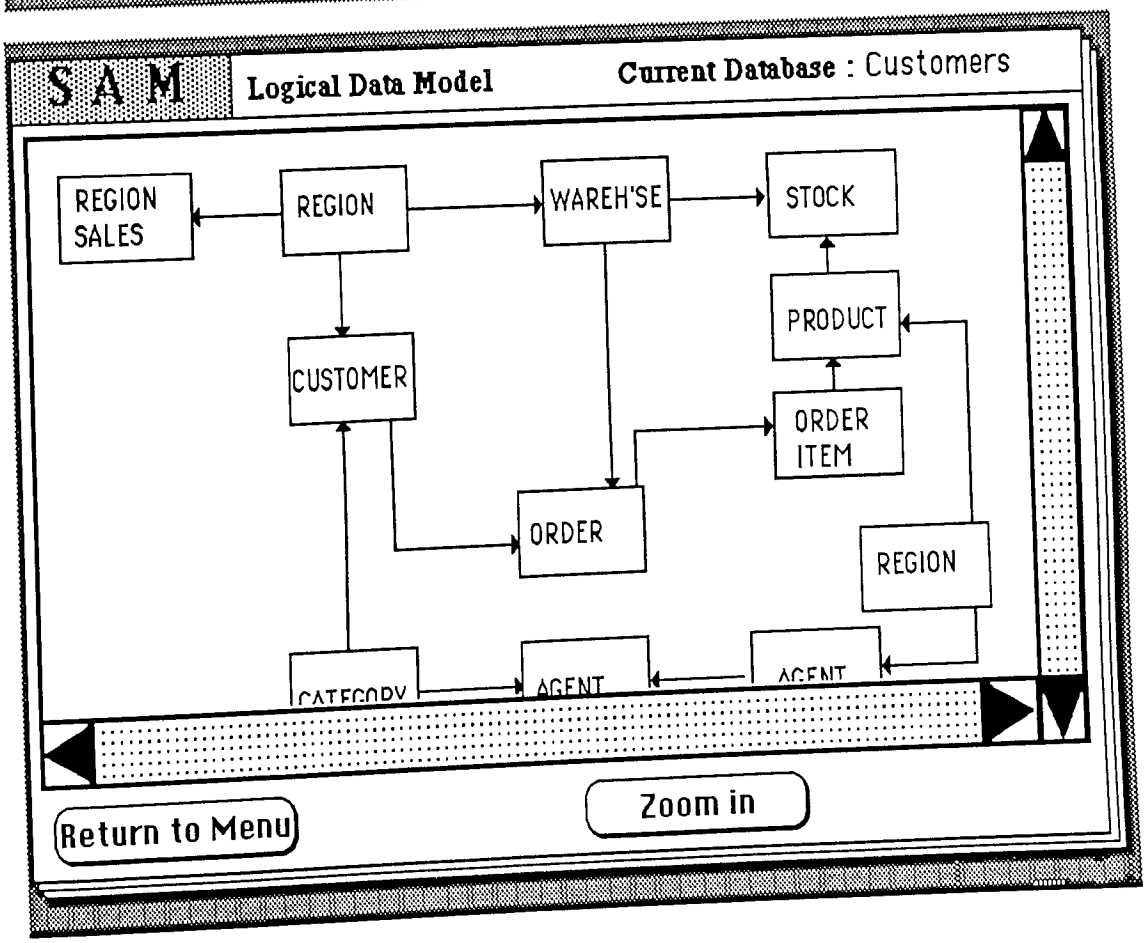

S A M
Current Database: Customers
PROFILE NAME : Cust.1.
cust.number
cust.order
print
Return to Menu
Select
Create
Delete
Edit
9

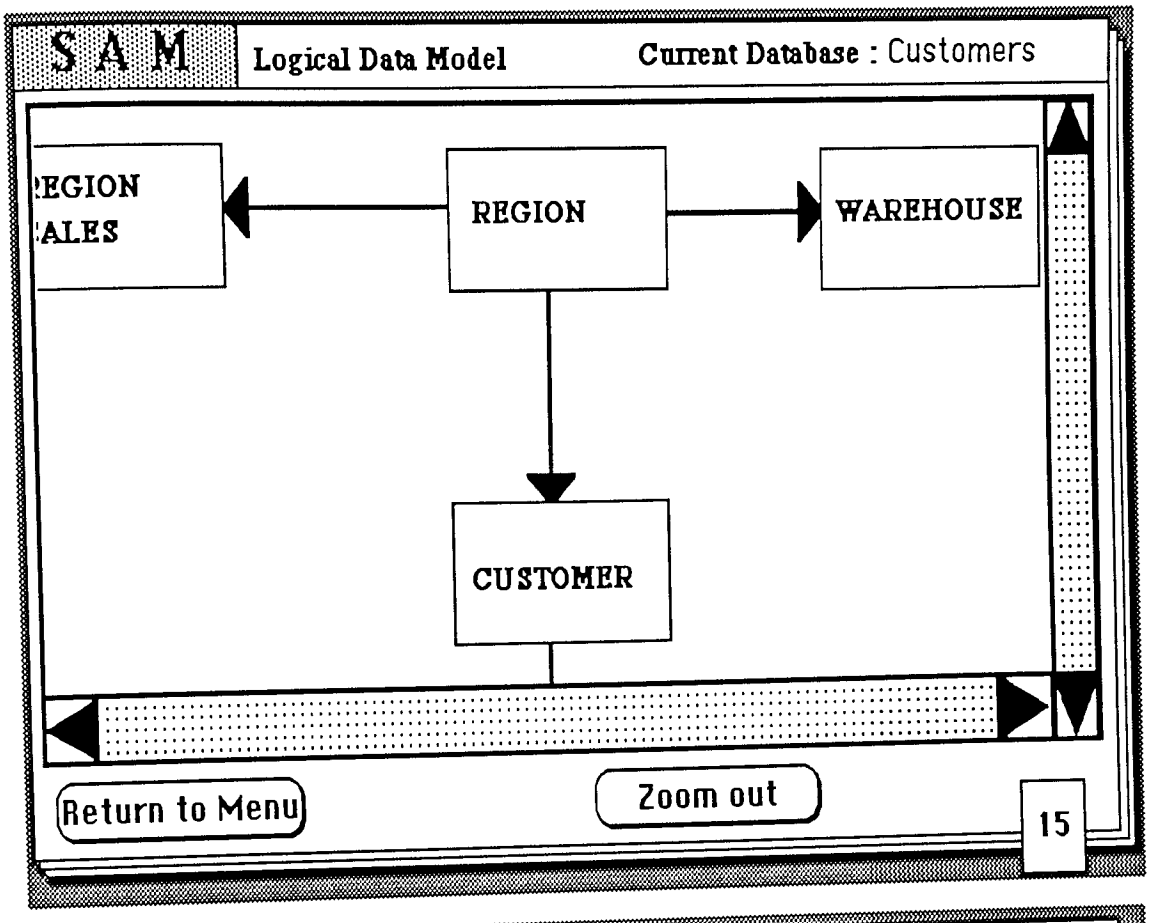

S A M
Current Database: Customers
PROFILE NAME : Cust.1.
cust.number
cust.order
print
order item *
Return to Menu
Select
Create
Delete
Edit
10

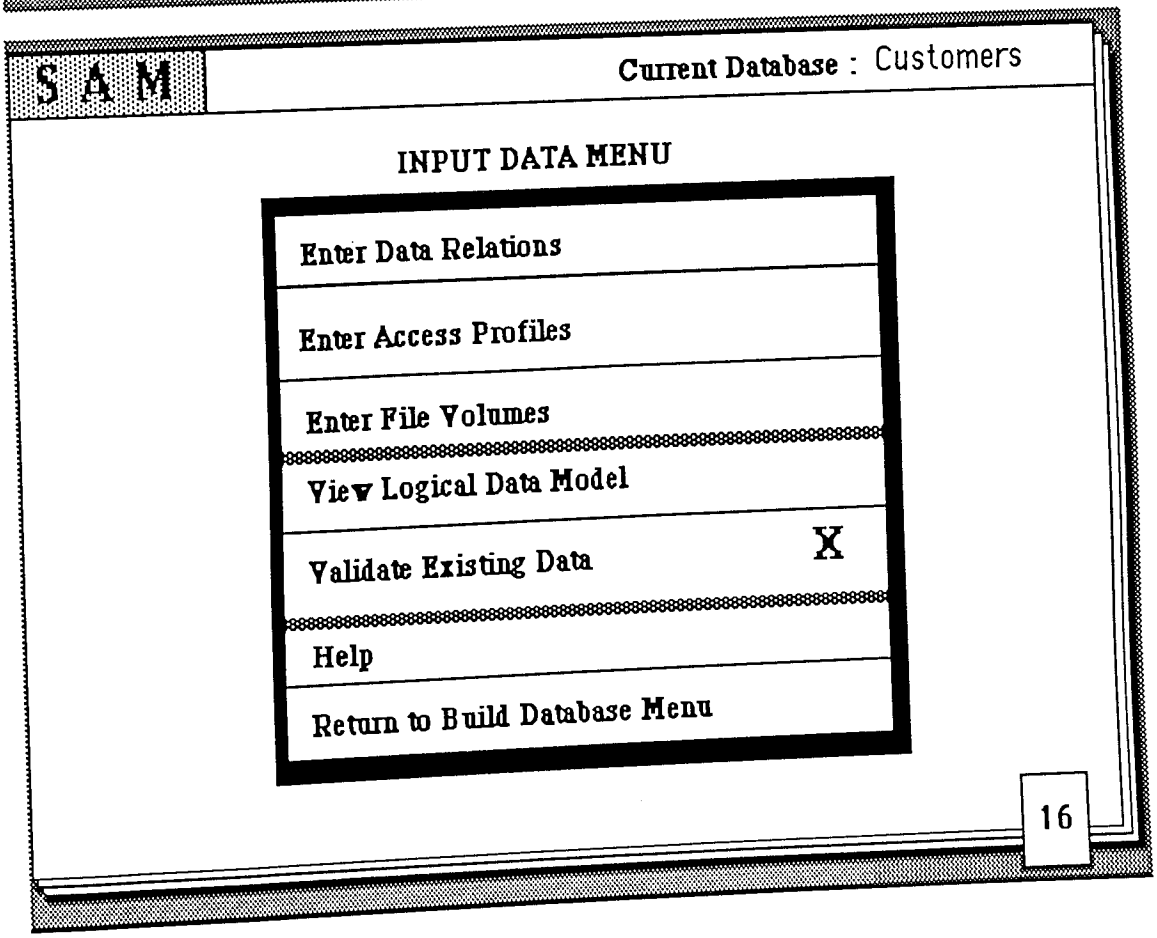

S A M
Current Database : Customers
INPUT DATA MENU
Enter Data Relations
Enter Access Profiles
Enter File Volumes
X
View Logical Data Model
Validate Existing Data
Help
Return to Build Database Menu
11

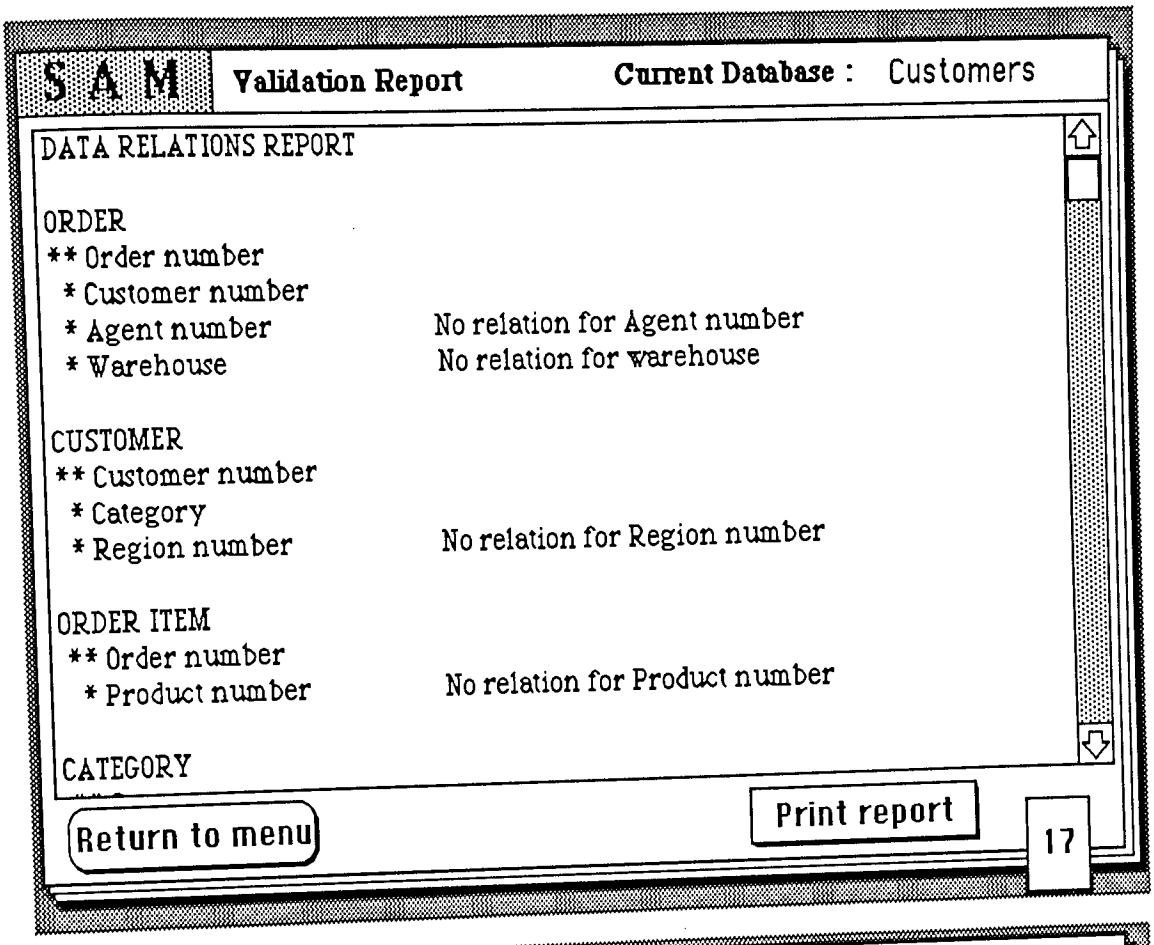

S A M
FILE VOLUMES
Current database: Customers
FILE : SUPL
NAME : Supplier
Logical size of records : 648
Current number of records : 2500
Expected growth : 10%
No. Added :
Period:
days
No. Deleted :
Period:
days
Return to Menu
Insert File
Find File..
Delete File
12

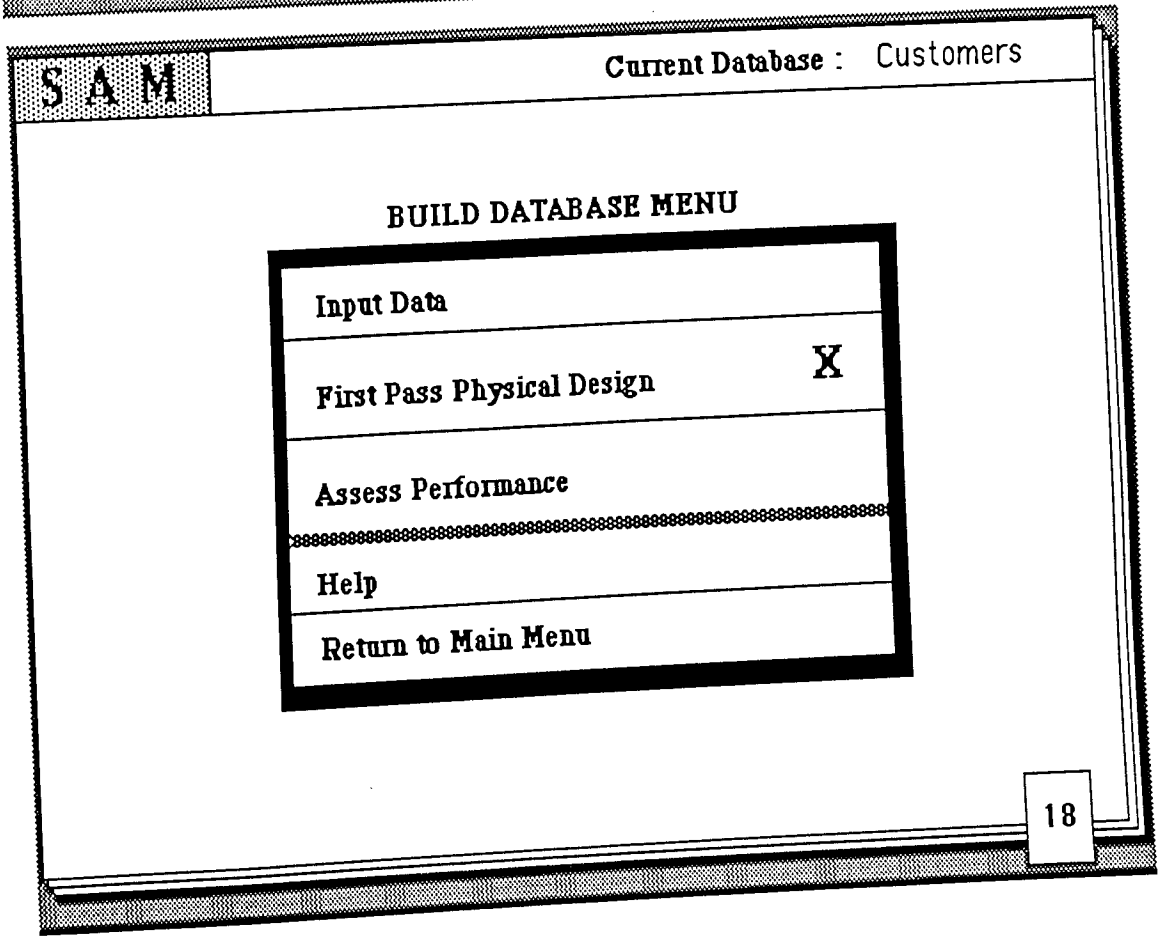

S A M
Current Database : Customers
INPUT DATA MENU
Enter Data Relations
Enter Access Profiles
Enter File Volumes
View Logical Data Model
X
Validate Existing Data
Help
Return to Build Database Menu
13

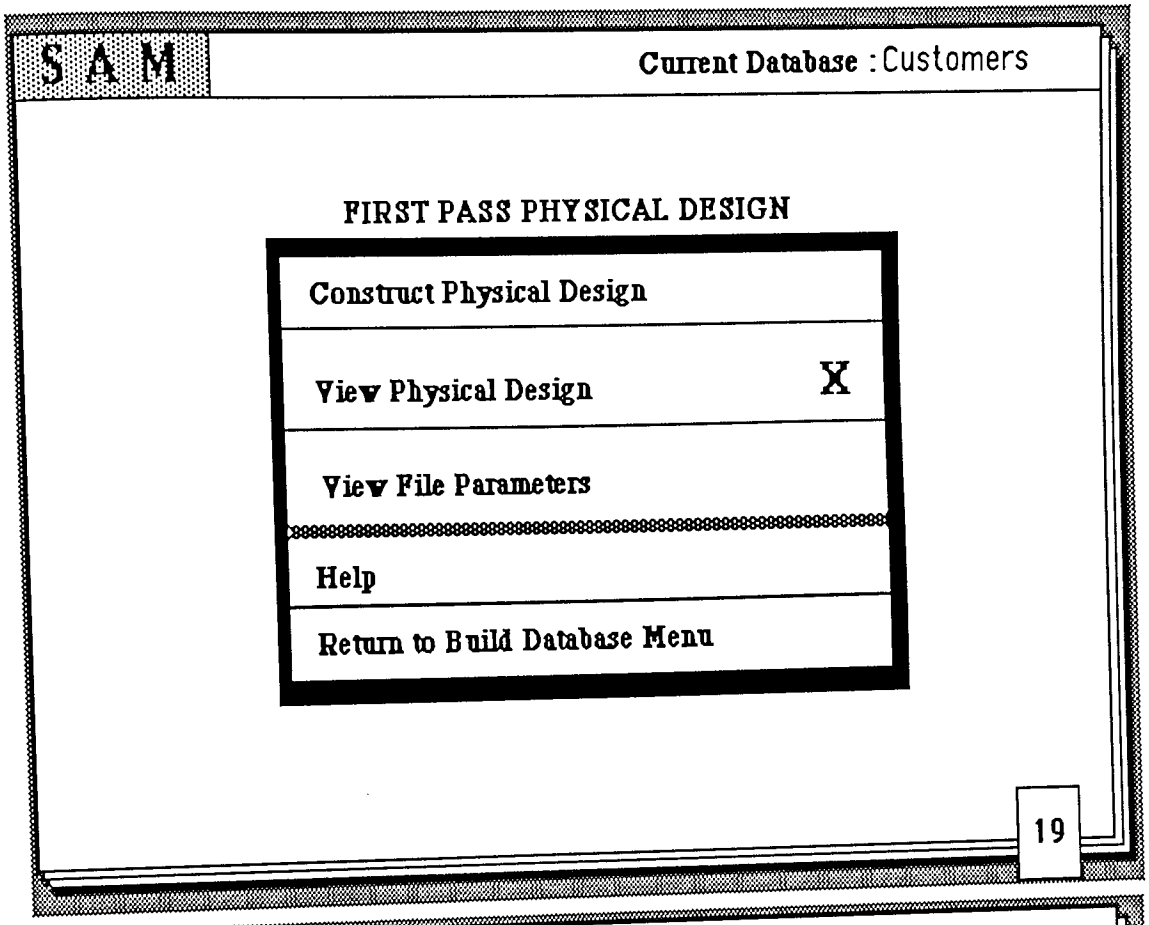

S A M
Logical Data Model
Current Database : Customers
REGION SALES
REGION
WAREH'SE
STOCK
PRODUCT
CUSTOMER
ORDER ITEM
ORDER
REGION
CATEGORY
AGENT
AGENT
Return to Menu
Zoom in

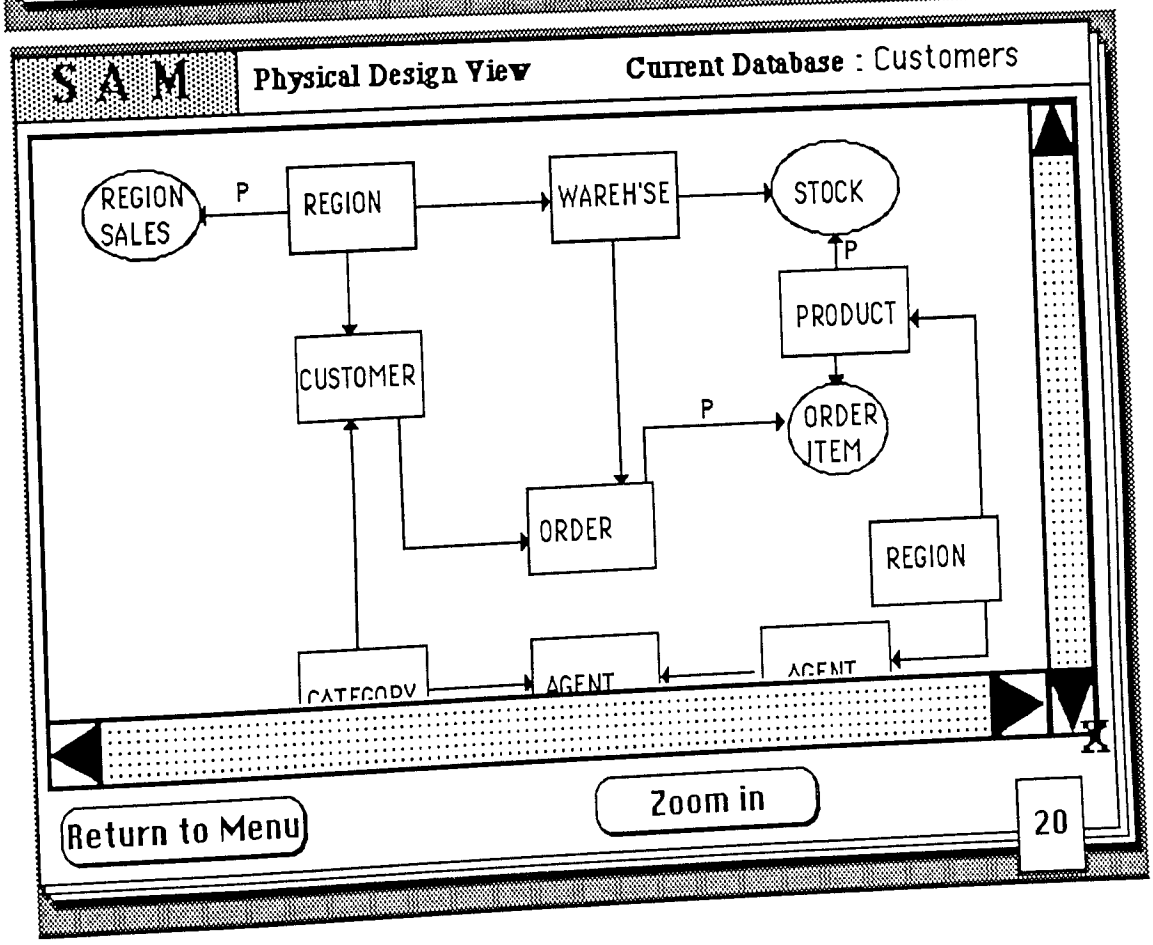

S A M
Logical Data Model
Current Database : Customers
EGION
ALES
REGION
WAREHOUSE
CUSTOMER
Return to Menu
Zoom out
15

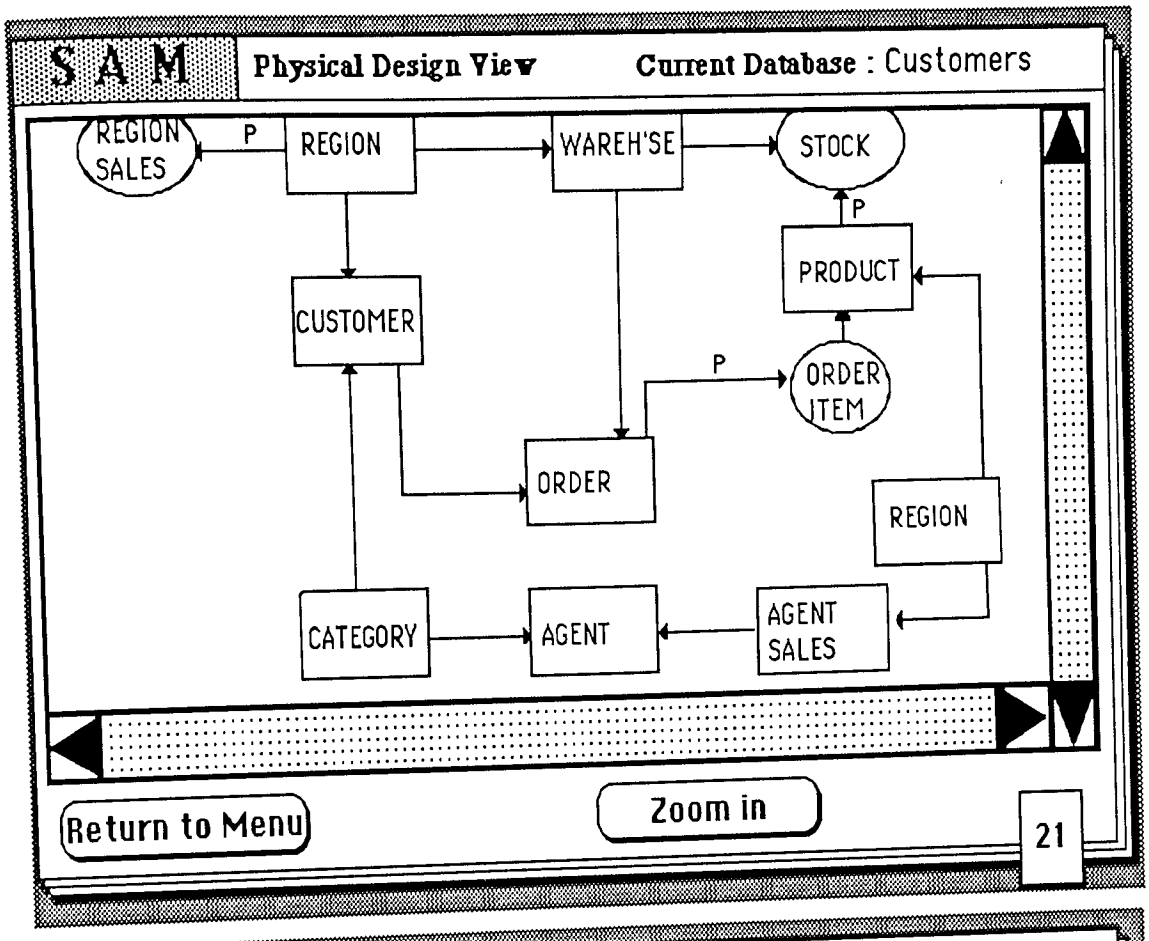

S A M
Current Database : Customers
INPUT DATA MENU
Enter Data Relations
Enter Access Profiles
Enter File Volumes
View Logical Data Model
Validate Existing Data
X
Help
Return to Build Database Menu
16

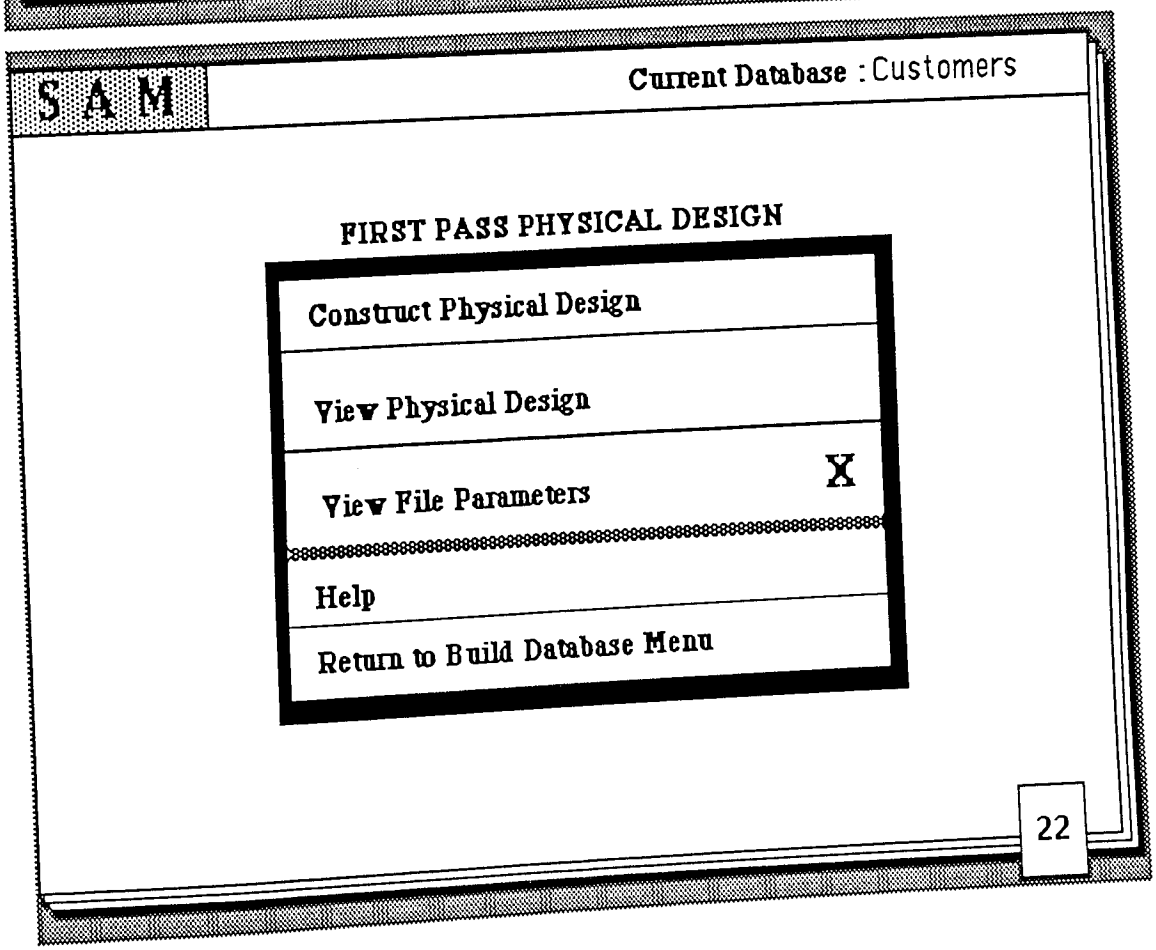

S A M
Validation Report
Current Database : Customers
DATA RELATIONS REPORT
ORDER
** Order number
* Customer number
* Agent number
No relation for Agent number
* Warehouse
No relation for warehouse
CUSTOMER
** Customer number
* Category
* Region number
No relation for Region number
ORDER ITEM
** Order number
* Product number
No relation for Product number
CATEGORY
Return to menu
Print report
17

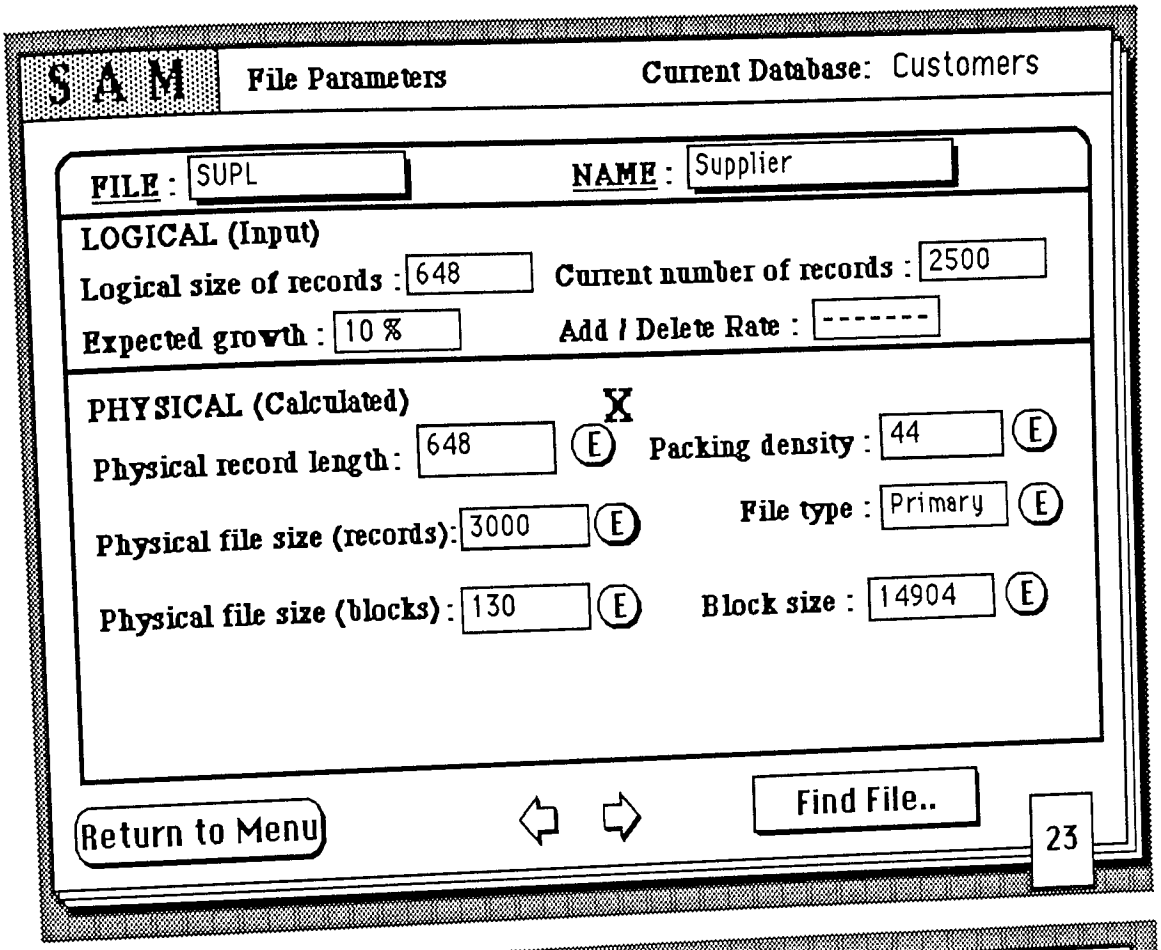

S A M
Current Database : Customers
BUILD DATABASE MENU
Input Data
First Pass Physical Design
X
Assess Performance
Help
Return to Main Menu
18

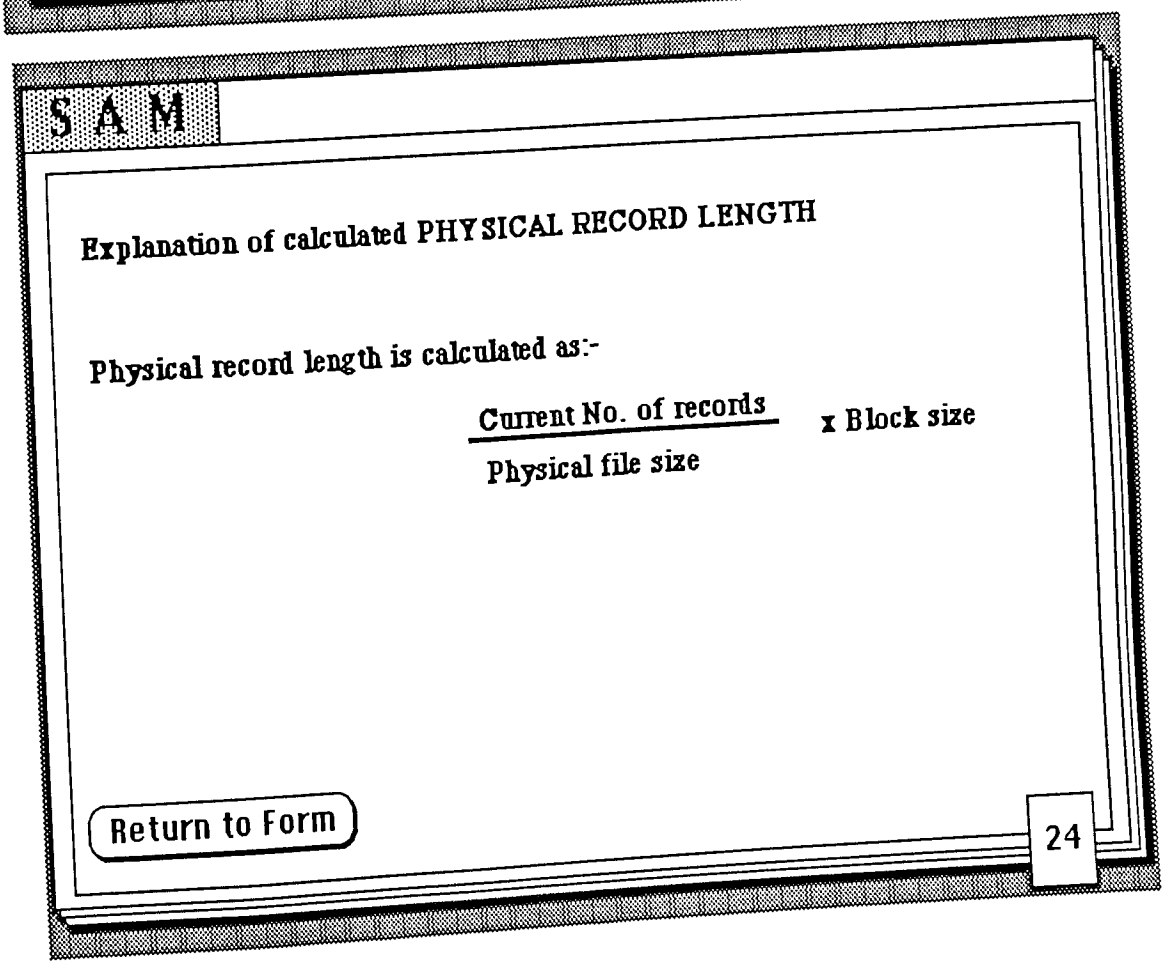

S A M
Current Database : Customers
FIRST PASS PHYSICAL DESIGN
Construct Physical Design
View Physical Design
X
View File Parameters
Help
Return to Build Database Menu
19

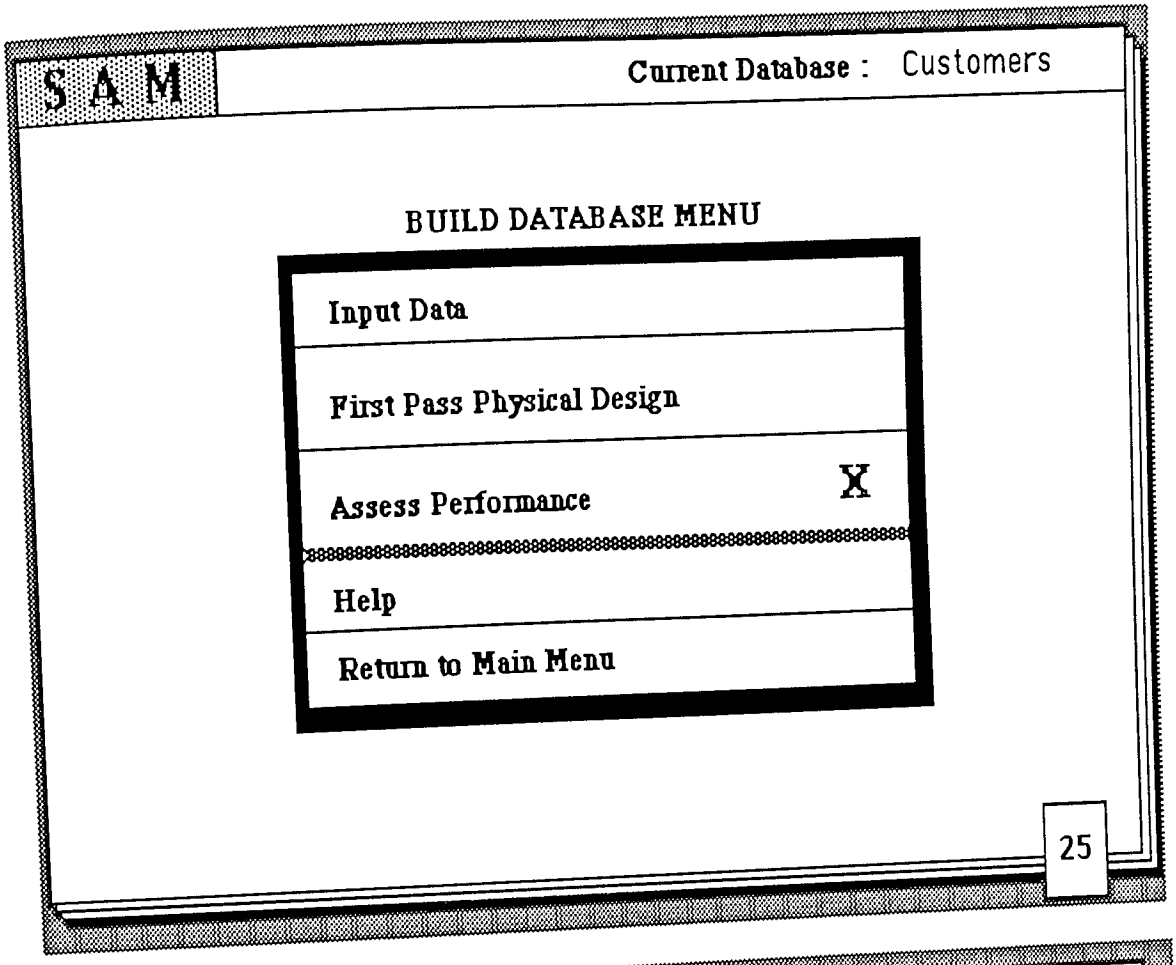

S A M
Physical Design View
Current Database : Customers
REGION SALES
P
REGION
WAREH'SE
STOCK
P
PRODUCT
CUSTOMER
P
ORDER ITEM
ORDER
REGION
CATEGORY
AGENT
AGENT
Return to Menu
Zoom in
20

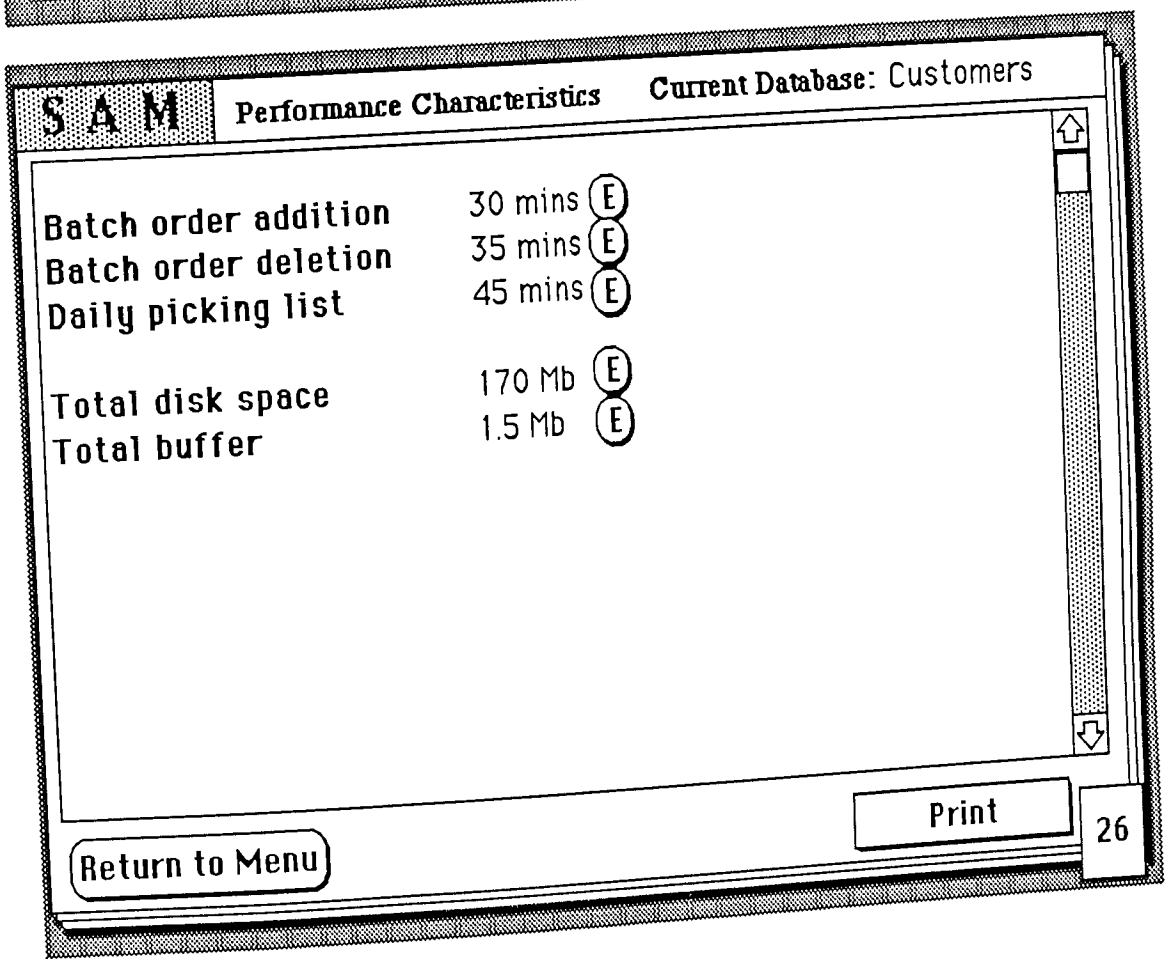

S A M
Physical Design View
Current Database : Customers
REGION SALES
P
REGION
WAREH'SE
STOCK
P
PRODUCT
CUSTOMER
P
ORDER ITEM
ORDER
REGION
CATEGORY
AGENT
AGENT SALES
Return to Menu
Zoom in
21


S A M
Current Database : Customers
FIRST PASS PHYSICAL DESIGN
Construct Physical Design
View Physical Design
View File Parameters
X
Help
Return to Build Database Menu
22


S A M
File Parameters
Current Database: Customers
FILE : SUPL
NAME : Supplier
LOGICAL (Input)
Logical size of records : 648
Current number of records : 2500
Expected growth : 10 %
Add / Delete Rate : -------
PHYSICAL (Calculated)
Physical record length : 648
Packing density : 44
Physical file size (records): 3000
File type : Primary
Physical file size (blocks) : 130
Block size : 14904
Return to Menu
Find File..
23


S A M
Explanation of calculated PHYSICAL RECORD LENGTH
Physical record length is calculated as:-
Current No. of records / Physical file size x Block size
Return to Form
24


S A M
Current Database : Customers
BUILD DATABASE MENU
Input Data
First Pass Physical Design
Assess Performance
X
Help
Return to Main Menu
25


S A M
Performance Characteristics
Current Database: Customers
Batch order addition 30 mins E
Batch order deletion 35 mins E
Daily picking list 45 mins E
Total disk space 170 Mb E
Total buffer 1.5 Mb E
Print
Return to Menu
26