AN EXPERT SYSTEM TO ASSIST IN DESIGN

Nicholas Kenelm Taylor

BSc (Wales), MSc (London), CEng, MBCS, AFIMA

Thesis submitted to the

University of Nottingham

for the degree of

Doctor of Philosophy

October 1990

In appreciation of

Winifred Alice Tyrrell, née Mackenzie

(1896 - 1954)

"This one's for both of us"

CONTENTS

ABSTRACT

This thesis describes a knowledge-based approach to Computer Aided Design (CAD). The need for a re-appraisal of the conventional CAD approach is demonstrated and a number of alternative approaches are examined. A methodology based on the manipulation of design constraints is developed and its implementation in an expert system shell called ALFIE is described.

ALFIE (Auxiliary Logistics For Industrial Engineers) is a concrete realisation of the ideas developed in the thesis. It combines a rule-based with a model-based approach to expert knowledge and the inferences that can be made from such knowledge. In its current form the system would be used alongside a conventional CAD system to assist in the design process, particularly during the preliminary stages of concept formation, as well as to provide expertise throughout the development of a complete design. The potential for, and problems associated with, integrating the ALFIE system with a CAD system are discussed towards the end of the thesis.

This work is primarily aimed at the provision of ergonomics knowledge to design engineers. The field of ergonomics is therefore used as a vehicle to demonstrate the developments which can be made to improve the CAD process. The resulting system is not, however, restricted to the domain of ergonomics alone. A case study of its application to a totally different type of problem is provided to indicate the variety of domains which can be handled using the ALFIE paradigm.

ACKNOWLEDGEMENTS

This thesis reports on work carried out over a number of years on grants from the ACME (Applications of Computing in Manufacturing Engineering) Directorate of the Science and Engineering Research Council, without whose financial support it would not have come to fruition.

The support and encouragement which I have received from June, Simone, Ursula, Hugh, Alexander and Matthew have given me great solace during the pursuance of this research. I thank them all for their patience but especially June, my wife.

I am indebted to all of my colleagues, from both home and abroad, for freely giving of their time in discussions and correspondence. In this respect I wish to mention specifically Dr Richard Barson, Dr George Boggs, Miss Alison Connolly, Dr Bruce Coury, Dr Phil Edwards, Dr Andy Freivalds, Mr Stuart Gibson, Dr Nick Milner, Dr Richard Schweickert and Mr Mike Simpson.

Finally, I wish to express my deepest gratitude to my two supervisors, Professors Nigel Corlett and Maurice Bonney, who jointly obtained the original research grant which supported this work. Professor Corlett supervised the bulk of the work after Professor Bonney left the department and I am especially indebted to him for inspiring me with comments and questions whilst allowing me the freedom to explore some unusual ideas. Professor Bonney took on the task of chivvying me through the write-up upon his return to the department and subsequent to Professor Corlett's retirement. His subtle Christmas cards

following my departure to Heriot-Watt University in Edinburgh kept this thesis alive. I am not sure which of them had the more onerous task!

Chapter 1

INTRODUCTION

1.1. Enhancing the Designer's Knowledge

If one of the objectives of this research had to be identified as its driving force, it would be that of providing expert back-up facilities to industrial designers. All who have ever attempted design work, on no matter how small a scale, know that a large proportion of the time is spent trying to determine the consequences of each decision made. This time is a function of the designer's own knowledge and how conscientious he is in acquiring further knowledge from other sources.

The efficiency of the design task is clearly inversely proportional to the time spent on it and directly proportional to the quality of the final product. The requirement to speed up the design process has spawned a vast array of design aids from drawing boards to complex Computer Aided Design (CAD) systems as a visit to any of the, now numerous, exhibitions will testify. The requirement to increase the quality of designs has led to the development of training techniques, standards organisations such as the ISO, and professional institutions such as the Engineering Council and the Design Council which attempt to regulate those involved in design activities.

However, the relationships between efficiency and time and between efficiency and quality place conflicting requirements on the designer. On the one hand, the dictate of time encourages the designer to make 'guesstimates' in areas where his knowledge is limited whilst on the other, the quality imperative indicates that he should be prepared to investigate these areas more rigorously.

In resolving this conflict, the attitude of the designer is perhaps the most important factor and attitudes are more susceptible to the effects of complacency than those of training or professional guidelines. Witness the tragic Bopal, Chernobyl and space shuttle Challenger disasters in recent years.

However, over the past two decades a number of advances have been made in the application of computers to the design task. One of the results of this has been to bypass part of the conflict mentioned earlier. CAD systems have enabled designers to evaluate many more design options and analyse them with a variety of software tools in less time than was previously possible. Consequently, the efficiency of the design activity has been improved with respect to both time and quality.

Whilst much development work can be expected before we have exhausted the potential value of these computer based analysis packages and their eventual integration with CAD systems there is no excuse for ignoring those problem areas which cannot be tackled in this manner. Design is one of those fields where the tremendous value of past experience is manifestly obvious. Yet experience is one of those intangible concepts which makes it very difficult to encapsulate in a piece of software. The potential benefits to be gained from any success in providing designers with ready access to the experience of others surely justify a considerable research investment in this area.

1.2. The Appropriateness of Expert Systems

The problems associated with capturing human knowledge within computer systems have long been the subject of investigation by researchers in the field of Artificial Intelligence. One of the more useful ideas to have come out of this work has been that of an Expert (or Intelligent Knowledge Based) System. The basic objective of an expert system is to emulate, on a computer, the characteristics of a human expert or experts.

The techniques involved are generally of a heuristic nature and permit the manipulation of data sets which can be incomplete and uncertain. An example of an incomplete data set is the currently known physical laws of the universe - we have not discovered them all by any means and yet we seem justified in reasoning about and generalising from those that we do know. These laws also provide us with an example of a data set which includes uncertainty - no physical experiment results in an absolutely certain outcome due to Heisenberg's Uncertainty Principle and yet we can build theories based on the results of these experiments and, by and large, they appear to be valid.

The up-shot of all this is that we use heuristics a lot and much of what we do and think is not amenable to straight-forward deterministic modelling. In particular, the manner in which we interpret, accommodate and assimilate our experiences is highly heuristic. It would therefore seem reasonable to adopt an expert system approach in order to create a model of an expert's experiences.

1.3. The Example of Ergonomic Design

From the point of view of design, ergonomics poses some particularly difficult problems. To begin with, ergonomics is a multi-disciplinary subject which covers such diverse areas as physiology, anatomy, psychology and sociology. Many of the interaction effects within these areas are not fully understood, let alone those interactions between them. Ergonomic theory is therefore distinctly patchy, with not so much gaps in the knowledge as *islands of knowledge in a large void.*

Another problem with ergonomic design is that ergonomics is not regarded as a mainstream design objective - as much as we might like it to be so. Ergonomics is not the primary reason for any design task, it is, at best, a secondary objective and the ergonomist has to be aware of this fact when analysing or assisting in the design of a work situation. Whilst it may well be possible to *fit the task to the man* the outcome after performing the task must always be preserved. The ergonomist, or a computer system which acts as an ergonomist, must always be prepared for certain restrictions, which are not necessarily intelligible in terms of ergonomics, to manifest themselves as a result of the primary objective.

Ergonomics is a field with patchy theory, a heavy reliance on practical experience and which is applied in a consultative fashion with the client retaining a considerable degree of autonomy. Ergonomic design is therefore a particularly hard nut to crack with computerisation and, as such, should provide a good sounding board for the computer based design paradigm advanced in this thesis.

1.4. Organisation of the Thesis

The approach put forward in the thesis would not hold water without a suitable vehicle which demonstrates its validity. Ergonomics was chosen as the vehicle and so an overview of this field is given in Chapter 2. This thesis brings together a number of differing themes. The major themes are those of computer aided design and expert systems whose development and combination are reviewed in Chapter 3.

Chapter 4 considers the fundamental requirements of a computer aid for ergonomic design and sketches the overall structure of the ALFIE (Auxiliary Logistics For Industrial Engineers) system which is proposed as a novel solution. Chapters 5, 6 and 7 detail the basic components of the ALFIE system - the Knowledge Representation, the Inference System, and the Interaction facilities respectively.

Chapter 8 describes the implementation details of the ALFIE system and Chapter 9 consists of a discussion of the problems involved in knowledge acquisition.

Chapter 10 describes, in detail, the knowledge domains which have been tested on the system. These are mainly ergonomic domains but one entirely different type of problem is included as an indication of the flexibility of which the ALFIE shell is capable.

Chapter 11 consists of a critical evaluation of the ALFIE system in which the main advantages and disadvantages are highlighted whilst Chapter 12 describes future work which the author hopes to carry out. This includes further

investigations into the general applicability of the shell of the system and a proposal for fully integrating the expert system with a conventional CAD system in order to combine the benefits of both approaches.

Chapter 13 summarises the contribution of the thesis to the fields of ergonomics, computer based design, expert systems and knowledge engineering.

Chapter 2

ERGONOMICS - AN OVERVIEW

2.1. The Importance of the Human Factor

2.1.1. Definition

The terms *Ergonomics* and *Human Factors* are used interchangeably as McCormick and Sanders (1983) indicate in their definition of the subject:

The field of human factors - referred to as ergonomics in Europe and elsewhere - deals with the consideration of human characteristics, expectations, and behaviors in the design of the things people use in their work and everyday lives and of the environments in which they work and live. In simple terms, human factors has been referred to as designing for human use.

Grandjean (1980) elaborates on what this entails:

Ergonomics is interdisciplinarian: it bases its theories on physiology, psychology, anthropometry and various aspects of engineering.

Clark and Corlett (1984) re-iterate these themes but emphasise the importance of the human element to a greater extent:

Ergonomics can be defined as the study of human abilities and human characteristics which affect the design of equipment, systems and jobs. It is an interdisciplinary activity based on engineering, psychology, anatomy and physiology and its aims are to improve efficiency, safety and operator well-being.

2.1.2. Ethics and Employee Morale

At the 1988 annual conference of the Ergonomics Society, the society's president, Professor Nigel Corlett, exhorted his colleagues not to be shy of stating their ethical position with regard to the application of their ergonomics knowledge (Corlett 1988).

Forever the optimist, I too shall assume that ethical arguments still carry some weight in circles where terms such as 'economic imperative', 'productivity bonus' and 'no strike deal' normally hold sway. If this optimism seems overly naive perhaps the following extract from a speech by John Selwyn Gummer in 1984 whilst Minister of State for Employment will help to explain its rationale:

It is both an historical and political truth that you cannot have a society in which the media give the overwhelming majority of people the feeling that they have a right to know and understand everything that is happening in the worldwide community ... in which man is dignified in his leisure and is expected to take an intelligent view of the rest of the world ... if he then goes to an industrial situation where he is ignored and where the demands of the production line are put ahead of any interests or concerns that he may have.

McCarthy (1988) indicates that the ambitions of employees are not simply limited to higher wages and job security. Job enrichment, greater autonomy and a higher degree of worker participation in management decisions have been shown to lead to greater job satisfaction and a consequent increase in employee morale (Eklund 1988, Larsson 1984, Lawlor 1984 & Herzberg 1959).

The problems of employee alienation are not new - they pre-date even the activities of the Luddites during the Industrial Revolution - but, as Amram (1984) points out, they may well be exacerbated by the technological revolution which we are now experiencing. It would therefore seem to be a very foolish as well as callous society which could turn its back on the possibility of improving the quality of working life at this point in our history. In the United Kingdom there is, at least, cross party agreement on the political expediency of a participative workforce, even though the major parties would employ different means to implement the policy:

Owning a direct stake in industry not only enhances personal independence; it also gives a heightened sense of involvement and pride in British business.

(Conservative Party Manifesto 1987)

We believe that the law should be used to enlarge, not diminish, the freedom of workers to control their environment.

(Labour Party Manifesto 1987)

Abolish class discrimination in the workplace ... by ... creating opportunities for all employees to share in the profits, decisions and ownership of firms.

(SDP/Liberal Alliance Manifesto 1987)

At a less political and more practical level, no discussion of the ethics of working practice can be complete without a consideration of the ideas of Frederick Winslow Taylor. The philosophy of *Taylorism* - the *-ism* should forewarn the

reader of its controversial nature - which Taylor himself called *Scientific Management* revolutionised managerial thought and practice (Buffa 1983). No longer would workers be left to determine the means by which production should be achieved according to their skills and experience - the methods used would be decided by management.

Taylor's approach is now regarded as "mechanistic and anachronistic" (Tricker 1979) but he did pave the way for the development of the modern discipline of management science without which there would be no structure through which to implement the more far-sighted, human-orientated policies of today's managerial practice.

2.1.3. Productivity and Quality

Tynan (1984) calculated that the cost of raising a child to the age of sixteen, so that he is sufficiently well educated and healthy for work, was ninety-seven thousand pounds in total in 1984. Assuming that this 'asset' is available to his/her employer for one quarter of the time available (forty-two hours per week) he/she represents an investment by the community of just over twenty-four thousand pounds in the company employing him. Tynan then asks the unanswerable question:

What return is appropriate for this, the only flexible, fully mobile, thinking, creative, innovative, talented asset?

In doing so, I believe he misses the main point of these figures. Nor is he alone in this. The point is that an employer invests very little in labour - the community at large bears the greater part of the effective purchase cost and the costs incurred by the employer are more in the nature of maintenance costs than capital investment. This, in turn, encourages employers to regard labour as a recurrent outlay and not invest sufficiently to ensure its productivity. This is in stark contrast to the usual attitudes to investment in plant and machinery - assets which represent an important capital investment to a company.

Much has been said concerning the reasons for the success of Japanese companies, see, for instance, Gillan (1987). I would not gainsay the majority of those put forward - they all play their part - but perhaps the degree to which the large Japanese companies invest in the education and health care of their employees and their offspring makes them more aware of the capital investment which an employee represents.

Once the idea of an employee as an asset has been grasped it is only logical to invest further funds in order to ensure a good return. However, when we try to quantify the appropriate amount of investment we meet Tynan's unanswerable question again. The following extract (Harbury & Lipsey 1983) is typical of economics textbook statements on the matter:

Some assets are locked up, for example, in the skills acquired by education and training. Such intangible assets are hard to evaluate in money terms and they are not included in [the figure].

How can we decide on the appropriate level of investment to ensure worker optimisation under such circumstances? We are left with only two assessment criteria - the productivity of the worker and the quality of the work he produces. The justification for investment in ergonomics is often based on potential improvements in these areas.

Ergonomics can provide answers to such varied questions as:

How should this workplace be arranged so as to optimise the productivity of the workers employed within it?

How long can an operator work efficiently and safely in this temperature?

How much illumination should be provided for this operation?

Should this warning signal be visual or auditory?

How much information should be displayed on this visual display unit?

The solutions provided to such questions as these can, undoubtedly, lead to productivity and quality improvements. Corlett (1988) cites a study by Spilling in which a work situation was improved to the extent that the company involved gained ten times the costs incurred in making the improvement, over a ten year period.

2.1.4. Safety and Compensation

I shall not dwell here on the ethical aspects of deliberately endangering the health of one's employees for the sake of financial gain. I feel certain that the reader will be horrified to learn that, in some underdeveloped countries, seven-day working weeks with up to twelve-hour working days can be found to this day and that, in Malaysia for instance, young children, whose bones are still growing, can be employed all day in a posture which is bound to lead to deformity (Corlett 1988).

Since 1945 there has been a considerable body of legislation passed by the UK government concerning worker health and safety. For example, the National Insurance Act, the Industrial Injuries Act, the Health and Safety at Work Act and the Factories Acts (Childs 1986). Failure to comply with the requirements laid down by this legislation can lead to hefty fines and even imprisonment.

Apart from the legal case for giving due consideration to health and safety factors there is also an economic argument. Poor health amongst employees can lead to reduced production through workers being 'under-the-weather', lost production due to sick leave and even direct financial loss in cases where industrial compensation has been awarded.

Mention should also be made here of the importance of consumer protection legislation. The design of a product is constrained by a host of British and International standards. Compliance with an ever-increasing number of these is obligatory and nearly all are concerned with the safety of the person who is to use the product. Once again, consideration for the human factor is finding its way

onto the statute books and manufacturers ignore ergonomics at their peril (McCormick & Sanders 1983).

2.2. The Development of Ergonomics

2.2.1. The Birth of Ergonomics

In one sense ergonomics dates back to the first use of tools by our pre-historic forebears. Whilst it is clearly absurd to suggest that anybody actually *studied* work towards the end of the last Ice Age, the invention of tools showed an appreciation of the fact that certain tasks could be performed more quickly, more easily or could even be made possible, through the application of a little thought. Once these first tools were developed more followed and those which already existed were refined to become more efficient and powerful. This specialisation led, for instance, to the enormous variety of such items as flint arrow-heads - each crafted for a particular purpose - which we continually dig up today. Generalisation followed too as the value of flexibility became apparent along with the simplicity of producing one tool which could be turned to many purposes, such as the double-headed axe. Clearly, early man thought not just about the goals or objectives of the tasks which he set himself but also about how he performed them (Oakley 1975).

From these early times through to the Industrial Revolution, man's development of ever more useful and complex tools went on apace. I will not labour the importance of the Industrial Revolution to the economies of the major European

powers of the time - the reader will be familiar with this. However, I think it is important to stress what a watershed the period was in terms of attitudes to production. From this point on industry geared itself to the idea of men operating machines with the machines being seen as the primary actor in production. This *machine age* represented a significant shift from the previous *aeons of tools* when man used tools to **assist** him and he was the major force in production (Christensen 1976).

The period from the Industrial Revolution to the Second World War consolidated the dominance of machines in the workplace. More and more attention was given to 'fitting the man to the job' by means of selection and training and during the First World War the military began to show an interest in these ideas. By the outbreak of World War II the industrial worker was of de facto secondary importance to the task he was performing.

However, the complexity of machines continued to increase and during the Second World War it became obvious that certain pieces of equipment were so complex that very few people could operate them effectively and safely. This signalled the birth of the modern discipline of ergonomics. The pendulum was beginning to swing back in the operators' favour. From this point on, more and more tasks have had to be 'fitted to the man' because of the sheer complexity of the machinery and organisational structures which they entail and the consequent demands that these make on the workers involved.

2.2.2. First Steps - At the Double

Unlike most other fields of study, ergonomics was not born out of curiosity. It was born out of necessity. Right from the beginning there were problems to be solved and there was very little theoretical work from which to derive the solutions. Ergonomics really did have to run before it had learnt to walk and it does not appear to have paused for breath ever since.

The practical bias which resulted from this problem-solving impetus has been a double-edged sword. Whilst it has mitigated against the ivory tower syndrome so often found in other disciplines it has also led to a very patchy theoretical base. This is not to say that there is a lack of basic data in the area of human factors. On the contrary, there is so much that no simple general rules or theories have ever been accepted by the ergonomics fraternity at large because data could always be found which falsified them. Indeed, the lack of simple theories which could be refined and developed over a period of years has probably been the biggest block to the development of any general theories of ergonomics.

Another obstacle to the development of a true science of ergonomics lies in the nature of the ergonomics data base. The problem-solving approach has led to experimental data gathered for the solution of individual, highly specific problems which does not provide much help to anybody who might take it upon himself to start building a general theoretical framework.

To summarise, the current state of ergonomics consists of very little theory, large amounts of very specific experimental data and a vast knowledge base gained by practitioners as a result of their experience but which is locked away inside their heads.

2.3. Ergonomics and Industrial Design

Very few design departments have a significant working knowledge of ergonomics and few managers know much beyond the term itself. Yet the efficient performance of the equipment and workplaces being designed is dependent on good ergonomic design giving workers the best possible opportunities to achieve work objectives with minimum hindrance from their surroundings.

(Corlett 1985)

For the purposes of this thesis we shall restrict ourselves to a consideration of the design of workplaces and practices and only consider equipment in as much as it affects the working environment. Size, strength, visual and auditory acuity, susceptibility to heat stress and draughts, comfort, posture - these are just some of the human factors of which the industrial designer *should be*, but often isn't, aware.

A recent survey of design engineers carried out in a number of American companies (Evans 1985) indicated a disturbing unwillingness to use ergonomic information in preference to 'common sense'. Moreover, even the most conscientious designer will have problems with eliciting information from libraries or colleagues if he/she does not know which questions to ask. The only

solution hitherto available was the, sometimes prohibitive, expense of hiring a consultant ergonomist. The bulk of this thesis is devoted to demonstrating an alternative computer-based solution which is both more readily accessible and less costly.

Chapter 3

COMPUTER BASED DESIGN - A REVIEW

3.1. The Development of CAD Systems

3.1.1. What is CAD?

CAD, or Computer Aided Design in full, is a fast developing area of computer usage. It has proved its worth in applications ranging from workplace layout and integrated circuit board design to the analysis of air-flows in aviation and tidal effects on coastlines. There are numerous CAD systems available commercially - each one developed for use in a particular field.

Whilst the precise definition of CAD is in a state of flux, probably the most general definition to appear in recent years is that of Groover and Zimmers (1984):

Computer-aided design (CAD) can be defined as the use of computer systems to assist in the creation, modification, analysis or optimisation of a design. The computer systems consist of the hardware and software to perform the specialised design functions required by the particular user firm.

However, CAD systems have had a chequered history, particularly from the point of view of the user. John (1988) summarises this in a table of the various meanings which have been attributed to CAD over the past decade or so. They are:

Cannot Actually Do (anything with it)

Computer Aggravated Draughting

Computer Aided Draughting

Computer Aided Design

Computer Automated Design (partial)

Computer Automated Design (full)

To put these labels in perspective, Computer Aided Design is accepted as the best description of the current state-of-the-art. Those meanings which appear earlier in the list properly belong to its past and those which appear later in the list are developments which seem likely to appear in the future.

3.1.2. The Past

The early developments in CAD went hand in hand with developments in the field of computer graphics (Besant & Lui 1986). Developments in this latter field were, as Foley and Van Dam (1984) point out, mainly concerned with hardware. In 1950 the MIT Whirlwind Computer was equipped with cathode ray tube (CRT) displays for output and by the mid 1950s the SAGE Air Defense System was using CRT displays in conjunction with light pens for input. However, SKETCHPAD (Sutherland 1963) is generally regarded as the first modern interactive graphics system. This system employed data structures for holding symbol hierarchies and interaction techniques for keyboard and light pen selection, pointing and drawing. Many of the ideas developed by Sutherland were so fundamental that they are still in use today (Foley & Van Dam 1984).

The computer graphics scene looked set for a major breakthrough in man-machine interaction. However, this was not to be for some years yet. Interactive computer graphics was not a viable proposition for any but the most technologically advanced enterprises. Foley and Van Dam give four main reasons for this:

1). Graphics hardware was expensive (there were no economies of scale);

2). Graphics required enormous computing resources (in terms of both processing power and storage);

3). Large interactive programs were difficult to write (concepts fundamental to software engineering were not known);

4). The software employed to drive the graphics devices was machine specific and not portable.

Over the past thirty years developments in display technology have given us vector, storage and raster displays culminating in the 'intelligent terminals' on the market today which have local display memories and processing power. Developments in input technology have yielded digitisers, joysticks, mice and even a limited speech understanding capacity (Foley & Van Dam 1984).

Developments on the software side have led to fast algorithms - sometimes hard-wired into the terminal for extra speed - which can perform tasks in a fraction of a second which would have been impossible a decade or so ago. Today's computer users take high-level graphics for granted and the scope for developments in CAD has increased tremendously.

3.1.3. The Present

Whilst developments in computer graphics can still create minor revolutions amongst the CAD community (viz. the advent of cheap colour graphics for instance) the area of geometric modelling is probably of even greater importance. A picture may speak a thousand words but if those words are to be meaningful there must be a sound model behind the picture. What is more, that model must be flexible enough to generate a wide variety of pictures.

For as long as the 'D' in CAD stood for Draughting both the picture on the screen and the model underlying it could be two dimensional and many problems were avoided. However, as the complexity required of CAD systems increased, the need for a three dimensional model became apparent. Once this idea of modelling more than just the two dimensional picture displayed on the screen is accepted a number of extensions to the basic CAD paradigm become possible.

A variety of non-geometric properties can be associated with the entities in a modern CAD system - colour, strength, manufacturing data, etc. The earliest major development in combining analysis with CAD systems was the application of the finite element method for stress analysis. This, like CAD, started with two dimensions and has been extended into the third dimension (Barson 1982). We are currently experiencing a rapid expansion in the types of analysis which can be applied to the models held within CAD systems which has culminated in the idea of *feature based modelling*.

Feature based modellers (Shah & Rogers 1988a, 1988b) have been developed in order to derive higher level information about the shapes of forms from the low level details of model geometry. These systems have appeared primarily in response to the requirements for greater automation of the link between design and manufacture.

CAD/CAM (Computer Aided Design and Computer Aided Manufacturing) systems are now commonplace and these can directly produce tapes to control NC (Numerically Controlled) machine tools (Groover & Zimmers 1984, Besant & Lui 1986). A more recent development has led to systems which can, at least in theory, produce off-line programs for the control of industrial robots. Edwards (1987) has investigated the application of such techniques to the highly skilled task of cutting patterns on manually blown crystal glassware for instance.

In summary, we currently have, in CAD, a very powerful modelling paradigm which can assist in the conceptualisation, modification, analysis and even manufacture of a design. So where can we expect this to lead us in the future?

3.1.4. The Future

There is no doubt that the conventional CAD system, based on a solid or surface geometric modeller and equipped with a variety of facilities to assist in visualisation, will continue to be the mainstay of design departments well into the future. Such systems are also certain to be enhanced with more efficient algorithms for standard tasks such as hidden line removal and interference checking. Data input techniques, which are currently one of the most primitive

aspects of CAD systems, will also be improved as the use of digitisers becomes more widespread. Improvements in computer graphics hardware and software are likely to result in exceptionally powerful stand-alone systems in which even the distinction between solid and surface modelling will become blurred.

Communication between different CAD systems will also become easier as standards such as IGES (International Graphics Exchange Specification) are adopted. The future of the conventional CAD system would therefore seem to be assured.

Feature based modellers, however, form the vanguard of a new trend in computer based design systems. This trend is moving away from the conventional visualisation approach to CAD and towards design systems which incorporate application specific expertise and techniques for reasoning about designs and the design process.

From reasoning about features a new initiative towards knowledge based design is developing. We are already seeing the advent of systems which can reason about shape and fit for instance (Carney & Brown 1989) which will progress to draw on developments in the new discipline of geometric reasoning (discussed further in Chapter 12) and take computer based design techniques well beyond the stage of mere visualisation of designs.

3.1.5. Summary

CAD started with computer graphics, moved on to geometric modelling and has now blossomed out to encompass analytical and reasoning techniqes from a number of engineering and scientific specialisms. In bringing a wide range of disciplines into the design system and therefore the design process, we meet the problem of the designer's knowledge of these areas. The average designer is unlikely to be an expert in all of the fields of stress analysis, manufacturing, electronics, etc. His expertise will lie in the functional requirements of the item being designed - an area which computer aids are unlikely to help for some time yet although Medland (1986) has proposed some interesting approaches which will be discussed later.

There would therefore appear to be a requirement to support the application of the various analytic tools with which the designer is being provided. In particular he should be able to draw on expertise which will assist him in deciding which tools to use and how to interpret the results produced. These are the sort of problems for which *expert systems* have been extolled as a solution.

3.2. The Development of Expert Systems

3.2.1. What is an Expert System?

Whilst there is no clear-cut definition as to what exactly qualifies for the label *expert system* a number of features are generally accepted as being important. Four of these are regarded as essential (after Forsyth 1984b):

A Knowledge Acquisition Module;

A Knowledge Base;

An Inference System;

An Explanatory Interface.

Every expert system needs some mechanism through which to capture the knowledge which it is going to expound. This is the function of the *knowledge acquisition module*. Its architecture, however, can vary enormously from a simple compiler which creates a database from a series of statements derived by a knowledge engineer from books and interviews to an inductive learning system which generalises from examples to create its own rules (Schweickert et al. 1987).

Having created a mechanism by which knowledge can be imparted to the system a suitable representation must be chosen to store the knowledge in. This representation and the knowledge contained in it are generally known as the *knowledge base*. A variety of representations can, and are, employed ranging from simple sequential lists of facts to complex networks which attempt to model the underlying structure of the domain. See Raphael (1976) or Winston (1984) for

example.

Once the domain knowledge has been imparted to the system and represented in an appropriate form a means must be provided whereby the system can actually 'reason' about the knowledge. This is achieved by the *inference system*. As the reader will have surmised, the inference system and knowledge representation are inter-related since the way in which the knowledge is represented will limit the type of processing which can be performed upon it and, vice versa, the type of processing required will restrict the forms in which the knowledge can be respresented. However, it is essential that the inference system does not depend on the content of the knowledge representation. That is to say that it should be possible to remove the knowledge and replace it with knowledge of a different domain without having to modify the inference system in any way. An expert system without any knowledge is called an expert system 'shell'. It is therefore common to state that shells should be domain independent. See Hayes-Roth et al. (1983b) for a more detailed discussion of this.

The fourth and final essential for an expert system was an *explanatory interface*. The emphasis here is clearly on the word *explanatory* since it goes without saying that any computer system should be provided with a satisfactory user interface these days. Expert systems will, under normal circumstances, have a sufficient explicit knowledge of what they are doing to provide an explanation of their actions to the user. This fact, combined with the probability that some of the system's actions will appear strange or even irrational to the user, means that a shell producer would be highly irresponsible if he did not provide an explanation

facility within his system (Forsyth 1984b). Michie (1982) introduces the concept of a *human window* to describe the idea that complex systems should be able to explain their reasoning in terms which are both intelligible and **executable** by a human. That is to say that the human should be able to learn and reproduce the system's rationale. Such is the importance accorded to explanation within the 'intelligent systems' fraternity today.

3.2.2. The Past

The earliest expert systems, in particular the DENDRAL system for inferring the chemical structures of unknown compounds (Feigenbaum et al. 1971), generated and then exploited a new paradigm in the field of Artificial Intelligence. Feigenbaum (1979) states this new paradigm thus:

We must hypothesise from our experience to date that the problem-solving power exhibited in an intelligent agent's performance is primarily a consequence of the specialist knowledge employed by the agent and only secondarily related to the generality and power of the inference method employed.

The subsequent development of expert systems, whilst utilising the earlier traditional tools of artificial intelligence research, concentrated much more on the problems associated with representing large amounts of knowledge than on improving the efficiency of search techniques or the generality of problem-solvers for example.

When we try to pin-point the origin of expert systems we can find a bridge between the traditional problem-solving paradigm and the specialist knowledge paradigm in the symbolic integration systems SAINT (Slagle 1963) and SIN (Moses 1967). Whilst these systems were based on a sound problem-reduction approach they also incorporated much specialist expertise and were never intended to be general problem solvers (Jackson 1974). Developments in this domain eventually produced the MACSYMA system (Martin & Fateman 1971) which surpasses the abilities of most human experts and is in daily use by mathematics researchers all over the world (Hayes-Roth et al. 1983b).

However, despite the prior claim of the SAINT system, the DENDRAL program is generally regarded as the first true expert system and the work carried out at Stanford University where it was developed rightfully deserves pride of place in any history of the subject.

The DENDRAL program surpasses all humans at its task and has led to a redefinition of the roles of humans and machines in chemical research (Hayes-Roth et al. 1983b). Begun in 1965 it has, according to Feigenbaum (1979), 'become one of the longest-lived continuous efforts in the history of AI [Artificial Intelligence]'. The DENDRAL program enumerates plausible structures for organic molecules given data from mass and nuclear magnetic resonance spectrometers and constraints provided by the user. The problems associated with acquiring knowledge for the DENDRAL program led to the development of META-DENDRAL which infers rules from mass spectrometry for subsequent use by DENDRAL.

Another landmark in the field of expert systems which emerged from Stanford University was the MYCIN system (Shortliffe 1976). MYCIN diagnoses and recommends treatments for antimicrobial infections of the blood. MYCIN incorporates explanation facilities, an ability to handle inexact reasoning and a truly deductive approach to the diagnosis task (Johnson & Keravnou 1985). These properties have enabled MYCIN to perform as well as, if not better than, medical experts. Again the problem of acquiring the knowledge which MYCIN needed led to the development of a tool which could assist in the construction of large knowledge bases. This was called TEIRESIAS and the reader is referred to Davis (1977) for further details.

The PROSPECTOR system (Duda et al. 1979) is the third expert system to have received world-wide acclaim. PROSPECTOR employs probabilistic reasoning to assist geologists in evaluating mineral sites for potential ore deposits. The system now includes over a dozen knowledge bases for different types of deposits and owes much, in terms of its architecture, to the earlier MYCIN system.

We could not conclude a discussion of the early expert systems without mentioning HEARSAY-II and its off-shoot systems (Erman et al. 1980). HEARSAY-II was developed to understand speech and, whilst it did not achieve the high standards of the previously mentioned systems, a number of its features have been noted as important for future developments in the expert systems field (Hayes-Roth et al. 1983b). Of particular interest amongst these features is the concept of multiple, co-operating specialists which may work at different levels of abstraction - a sort of management hierarchy - which, in the light of recent

developments in parallel distributed processing (Rumelhart et al. 1986), appears to be a very promising line of inquiry.

3.2.3. The Present

In recent years developments in expert systems technology have led to a plethora of expert system shells appearing on the market. Shells are available for most, if not all, platforms and a diverse range of inference and knowledge representation combinations can be found (Guilfoyle 1986a & 1986b).

Expert systems have now developed to the point where they can often be treated as just another computer package. In fact, many users of computer systems are not even aware that the packages which they are using are based on expert systems. Given the heuristic nature of expert systems, this is, perhaps, a little worrying; especially when one appreciates that the reason for this lack of awareness is partly due to a deliberate marketing policy on the part of the vendors of these systems. The arrival of expert systems technology in the commercial marketplace in the early '80s was accompanied by much hype and the usual exaggerated claims which we have come to expect from those who trade in the wares of the AI research community. Needless to say, expert system shells fell dismally short of expectations (Kidd & Sharpe 1988, Harris et al. 1990). Whilst, initially, this could be attributed to the inappropriateness of the applications, latterly it has been mainly due to the phenomenal effort required to impart the all-important knowledge to the empty shells (Taylor et al. 1987). As a result expert systems got a bad press and those companies selling packages which were

based on successful expert system implementations were naturally not too keen to broadcast this fact.

The successes of expert systems and shells over the past decade, however discreet, have enabled the technology to build up a solid base of applications which, in turn, have acted as a basis for a renewed perceived credibility in the marketplace. This 'new dawn' is exemplified in a recent Department of Trade and Industry brochure (1990) which cites a diverse range of applications of expert systems being employed by some of the largest and best-known names of the UK commercial and public sectors, some of which are reproduced below:

* *Real time process control in cement plants - Blue Circle Industries plc*
* *Advice on International Atomic Energy regulations - British Nuclear Fuels*
* *Dyspepsia diagnosis and treatment - Glasgow Southern General Hospital*
* *Fault recovery management in computer operations - Inland Revenue*
* *Product design analysis - Lucas Engineering and Systems Ltd*
* *Personnel selection screening - Marks and Spencer plc*
* *Interactive activities scheduling - Rolls-Royce and Associates Ltd*
* *Personal pension forecasting - Department of Social Security*

There are, however, many problems still to be addressed if expert systems technology is to progress to a point where large systems can be used by non-experts without major reservations. Typical of such reservations is the following quote from Harry Pople (1984), co-developer of CADUCEUS (an expert consultation system for internal medical) with Jack Myers (a specialist in internal medicine):

CADUCEUS is a beautiful tool in Jack Myers' hands. I don't trust it with anybody else at this point.

3.2.4. The Future

Bower (1988) lists a number of limiting factors to the wider application of expert systems, some of which I reproduce below:

** Domain Choice. Some areas of knowledge are just not suited to a logical reasoning process.*

** Acceptability. Not everybody wants to use, let alone rely upon, a computer system.*

** Uncertainty. Expert systems are not as adept as humans are at handling incomplete or uncertain data.*

** Learning. Human experts learn from experience but, as yet, the vast majority of expert systems cannot.*

** Limitations. A human expert has a lot more so-called 'metaknowledge' concerning his own limitations whereas the majority of expert systems will tend to produce an answer even if it is incorrect.*

** Testing. It is very difficult to validate a large system for completeness and soundness.*

** Behaviour. Expert system consultations tend to be system driven as opposed to user driven.*

** Knowledge Acquisition. Obtaining knowledge for an expert system is far from easy.*

** Sensory Experience. Much can be learnt in forms other than rules and humans can perform 'latent' learning without making a conscious effort.*

** Common Sense. Expert systems cannot exhibit anything like the degree of common sense that humans can.*

** Spatial and Temporal Reasoning. Expert systems have made very little headway in this area.*

It would seem reasonable to discuss the future of expert systems in terms of these drawbacks and to indicate the likelihood of progress reducing them.

Domain Choice

I believe that there is a certain amount of speculation implicit in Bower's statement here. Whilst there will always be domains which have not been investigated fully enough to permit a formalised rule-based description, it is hard to see why any particular domain should be impossible to formalise given enough time and effort unless it be for want of an understanding of the domain in the first place (in which case it can hardly be considered an intrinsic limitation of expert systems). It is worth noting also, that the *Golden Rule* of expert system development, namely *select a small domain of expertise*, is currently being undermined by modular approaches utilising multiple co-operating expert systems. See Popplestone (1987) for an example of this.

Acceptability

This is a general problem for any form of new technology. Whilst individual successes and failures will affect the short-term acceptability of a new idea, time, as always, will be the final arbiter on the matter.

Uncertainty

Reasoning with *incomplete data* and *uncertain theories* and whether it is at all possible to justify any conclusions reached from such reasoning are problems which have taxed the minds of philosophers since the ancient Greeks first raised them. I make no apology for the length of the following discussion since it is critical to the philosophy underpinning the ALFIE system described in the body of this thesis. The fundamental problem of human reasoning was stated by David Hume:

Thus not only our reason fails us in the discovery of the ultimate connexion *of causes and effects, but even after experience has inform'd us of their* constant conjunction, *'tis impossible for us to satisfy ourselves by our reason, why we shou'd extend that experience beyond those particular instances, which have fallen under our observation.*

(Hume 1739)

In other words, whilst we know that the repetition of certain events gives us no reason to draw any rational conclusions from such repetitions, we persist in drawing such conclusions in our everyday lives. This *irrationalist* observation clearly causes a problem for empiricists and scientific realists. Sir Karl Popper

claims to have solved this *problem of induction* in 1927. He refutes the 'commonsense' principle of induction† and his argument runs as follows:

Whilst our experience cannot amount to a proof that a particular theory is true, it can permit us to prove that a theory is false. Furthermore, a *preference* with respect to the truth or falsity of some competing theories can be justified by empirical reasons. For instance, at any point in time some of the theories may have been shown to be false and clearly we should *prefer* those that have not. We should also prefer the *best tested* theories (ie. those that have been subjected to the most severe and crucial tests) and Popper argues that it is this rational trial-and-error process which underlies our commonsense reasoning. This is the basis of Popper's *Conjecture and Refutation* approach first published in 1963 and which is now very popular (Popper 1989).

Popper first put his solution of the problem of induction into print in German (Popper 1933). It was largely ignored. In order to gain a wider audience for the solution and its implications for the scientific community he re-iterated it in English in 1959 (Popper 1977). Again it aroused little interest and so he attempted to clarify it further in 1972 (Popper 1983). I have included this historical aside to put the attitude of the scientific community towards the problem of induction into perspective.

Nowadays, Popper's solution is tacitly accepted by scientists who have taken the trouble to acquaint themselves with it and yet we still find discussions of reasoning under uncertainty completely ignoring its consequences (Alvey 1984).

† Note that this is *not* the same as the mathematical principle of induction which requires a *proof* of the validity of the induction step.

We are developing *probabilistic* expert systems which attempt to quantify the 'reasonableness' of their outputs using the irrational notion of *sufficient* evidence. In the very restricted and closed world described by an expert system knowledge base this may well be acceptable but the results produced by expert systems are applied to the real world which is, as far as our knowledge of it is concerned, still open. In order for such expert systems to bear any fruit a human expert is required to interpret the output, imposing a semantics on it which varies depending on the precise circumstances (Shafer & Tversky 1985). Current approaches to reasoning under uncertainty are not suitable for expert systems which are intended to be used by non-experts and the doubts raised by Pople (1984) and mentioned earlier remain.

Learning

Much work has been accomplished in the field of machine learning. In recent years one of the driving forces behind this work has been the need to acquire knowledge for expert systems without performing the time-consuming and error-prone tasks of interviewing human experts. See the discussion of Knowledge Acquisition below. Bower, however is thinking more of an ability to learn from mistakes and schemes of this nature are plentiful. The only problem is to decide whose feedback can be trusted and whose can be safely ignored!

Limitations

There is no reason why expert systems cannot reason about their own reasoning processes. In fact, expert systems are expected to explain their reasoning nowadays and this cannot be achieved at anything other than a superficial level

without the use of meta-knowledge. The extension of these meta-logical ideas to cover identification of problems which an expert system cannot solve is really quite trivial.

Testing

The problem of testing the validity of an expert system's responses is a very thorny issue and it is for this reason that the majority of expert systems are designed for use by *experts* and not *novices*. It goes without saying that if one could fully validate a particular expert system then that system didn't actually need to be an expert system in the first place. Once again, the test of time will determine which systems are reliable and which are not.

Behaviour

As the domains of expert systems increase in size, so it will be easier to pass control over to the user of the system. The ALFIE system described in this thesis has an architecture designed to cope with a large number of domains. It therefore employs a mixture of user-driven and system-driven dialogue styles. The majority of expert systems have domains that are so narrow that there is little point in providing the user with any great degree of control since she/he will have to answer the same questions either way and the system-driven way will probably be the most efficient for him/her.

Knowledge Acquisition

The problems associated with eliciting knowledge from human experts are described in detail in Chapter 9 of this thesis. These problems have led knowledge engineers to turn to work being carried out on machine learning. Since the area of machine learning which is most useful in knowledge acquisition is that of generalising from examples, and since this is an example of *induction*, Popper's refutation becomes relevant to the applicability or usefulness of any rules induced (though not necessarily to the algorithm used to induce them). Once we devote a bit more thought to which areas induction is plausible for, then the developments in machine learning will be of great benefit.

Sensory Experience

The argument that the potential of current computer programs to learn is limited because they are not able to directly sense the real world is, in my opinion, not a generally valid one. All computer programs exist in an environment which they can examine if required. The environment is a filespace and file access routines can be used to investigate and manipulate that environment. The view that current programs are not free to experiment is patently false. The fact that current programs have not exploited this opportunity for an empirical approach to learning is solely due to the fact that the application (file manipulation) has not yet appeared to merit any effort.

However, particular application programs, such as the PROSPECTOR mineral exploration expert system, which attempt to reason about the real world could well be improved if endowed with some means of directly measuring the

variables on which their reasoning is based. Latent learning - the things we learn subliminally - is particularly difficult for an expert to formalise since, in a sense, the knowledge which has been gained has never been consciously analysed. Yet latent learning can be very important. See, for instance, my work on cognitive map creation with a mobile robot (Taylor 1979). By no means all expert systems deal with domains where this would be appropriate though.

Common Sense

Just what is Common Sense? In expert system terms, does it belong in the knowledge base or in the inference system? There is a difference between common sense *knowledge* and common sense *reasoning* which is often overlooked. If the limiting factor is a result of the specificity and small size of current knowledge bases then we might have grounds for optimism since there appear to be many opportunities for handling large volumes of data on the horizon (such as, parallel and/or distributed processing and very large data base technology). If, however, the limitation lies in the specificity of the inference techniques used by current systems then we have only to recall the fate of the General Problem Solver (Newell & Simon 1963) in order to become more pessimistic.

Spatial and Temporal Reasoning

Though difficult topics to tackle, space and time have been subjected to much investigation in recent years by the Artificial Intelligence community. A brand new discipline, *Geometric Reasoning*, is beginning to take shape (Kapur & Mundy 1988). This subject has grown out of the work of a number of researchers

in different fields each with different goals in mind (Woodwark 1989). The design of spatial arrangements within the context of CAD has been of considerable importance in this development (Green 1987, Bonney et al. 1989). This field will be further discussed at the end of the thesis.

The area of temporal reasoning can be readily divided into two fields; one concerned with reasoning about chronology per se and the other concerned with non-monotonic reasoning where the truth of statements may vary over time.

The former, typified by Hayes (1979), is concerned with the limitations of the classical situation calculus in which actions and events are represented as functions which transform given world states into other world states. The concept of a *history* which Hayes suggested permits an extension of the classical state-space representation to a state-*space/time* representation in which the time at which a state exists is as important as the state itself. Reasoning about when a given state was extant then becomes no more difficult than reasoning about which states are extant at a given time, for instance.

The second field of temporal reasoning is particularly relevant to real-time expert systems employed in monitoring tasks which have to deal with a continually changing world. This requires a knowledge base containing statements which may be true and then become false and then return to true again, etc. In order to cope with such situations non-monotonic logic must be used. To this end work has been progressing in the development of truth maintenance systems (Doyle 1979, de Kleer 1986) which are designed to maintain a consistent set of beliefs in an environment of continually changing information.

3.2.5. Summary

Expert systems have successfully emerged from the periods of uncertainty of the 1980's to demonstrate that they can provide solutions to problems in a number of domains. The basic paradigm of building systems with a large amount of knowledge relating to a small area of expertise appears to have been vindicated and the future of expert systems looks bright with a number of exciting developments already underway.

One area in which expert systems have not made much headway as yet is that of design. The creativeness inherent in the activity of designing an artifact has probably been the major obstacle in this regard. The goal of developing expert systems which can autonomously and creatively design an artifact seems a long way off. However, if the goal of expert systems is reduced slightly when it comes to design then perhaps their applicability can be increased. Two ways in which the goal of expert systems in design could be restated are discussed in the next section. One is to set the goal of expert systems to be that of an *assistant* to a human designer who would handle the creative aspects of the design. The other is to apply expert systems only to the more routine design tasks which require no creative element and which could be handled by an autonomous agent.

3.3. Augmenting CAD with Expert Systems

A tremendous amount of work has been done in providing knowledge based systems as back-up sources of information for designers. However, in order to integrate these systems into an overall design system, a number of problems must be addressed. To date, attempts to improve the facilities available to the users of CAD systems can be placed into three broad categories. Each tackles one major problem but often pays little heed to the others.

The first is concerned primarily with the problems of preliminary design. During preliminary design the underlying design model is subjected to a lengthy iterative process of modification before a final consistent design manifests itself. This approach typically engages the designer in an interaction in which a collection of parameters and relationships (normally numerical) are added, removed and amended until the design system contains a mutually consistent set of parameter values. Since the designer makes the decisions in this approach, with the system acting as a consistency checker, I shall call this the *design assistant* approach.

The second category is concerned with automating as much of the design process as possible. Typically a design is broken down into a succession of sub-design tasks until the final set of tasks is more or less routine and can be drawn straight out of a library of previously designed components. This I shall call the *design automation* approach. Much design work is sufficiently routine for this approach to be viable but there is no scope for any really creative or novel ideas to be generated.

The third category is concerned with fundamentally changing the way in which design is currently performed. It seeks to employ the computational power in order to create a sea-change in the way we view design. In order to be managable, any large design task has to be broken down into smaller tasks. Often these smaller tasks are distributed to different groups of designers. Communication problems can leave each group knowing very little about the overall function of the artifact being designed and hence dangerously ignorant of how their particular piece of the overall jigsaw has to fit in with other pieces. Medland (1986) has proposed a methodology which seeks to break a complex design down into sub-designs on the basis of function (rather than geometry or manufacturing process required, etc.). His method also makes clear communication between the design groups a central theme. Following Medland, I shall call this the *functional design* approach.

3.3.1. Design Assistant Approaches

The pioneering work of Latombe (1977 & 1979), which sought to utilise AI *problem-solving* techniques in the development of CAD systems, has generated much interest and research over the past decade. Latombe's approach was based on the view that design is a process in which some aspects are best handled by people whilst some are amenable to automation. He introduced the idea of constraining relationships within a design which was taken up by later reasearchers. Sapossnek (1989) distinguishes between two approaches to handling relationships between design parameters. These he calls *parametric* and *constraint-based.* Both are model-based (as opposed to rule-based) and are

normally composed of numerical equations in the design parameters.

The Problem-Solving Approach

Latombe's TROPIC system underlies many of the approaches outlined in this brief review. However, not all of the ideas incorporated in the design of TROPIC have been followed up and so it is deserves a brief discussion in a section of its own.

The TROPIC system distinguishes between three different types of knowledge. The *domain* knowledge is concerned with the general type of design being undertaken. The *problem* knowledge is concerned with the particular goals which are to be achieved by the current design. The *problem-solving* knowledge is concerned with guiding the system towards a solution. TROPIC employs a hierarchical system of entities, as commonly found in CAD systems, in which the geometry or material of a component is the key feature. However, TROPIC also heeds the importance of functional relationships between entities and includes associative links to represent them.

The TROPIC system works by accepting, from the designer, as much knowledge as can be supplied along with a list of constraints which the final design must satisfy. The output of the system can be summed up as a set of embellishments to the design constraints entered by the designer. Clearly, a large amount of domain and problem-solving knowledge will reduce the amount of problem knowledge which is required. In the extreme, the problem knowledge might be reducible to a single statement of the form "design a family saloon car". The amount of domain and problem-solving knowledge required for such a task would be vast and so, for

the forseeable future, the TROPIC system will remain a design assistant system rather than a design automation system.

The Parametric Approach

The parametric approach is typified by the work of MacCallum et al. at Strathclyde University on the DESIGNER system (MacCallum & Duffy 1987, MacCallum et al. 1985). This system uses the relationships between the design parameters to create a dependency network between the parameters which records which parameters are dependent on which others. Any changes can therefore be percolated through the network to arrive at a new consistent set of parameter values whenever one or more is changed by the user. As Sapossnek points out, this approach has the drawback that the network is a di-graph and so does not really represent the relationships between the parameters but rather provides a method for evaluating a single pre-selected parameter from a number of others. It is not possible to use the same equation to evaluate any of the other parameters with which it is concerned. Sapossnek makes much of this but, as we shall see later, it is quite possible, and perhaps preferable, to use this approach with a number of equations each having a different subject variable by performing the algebraic manipulations needed prior to incorporating these equations in the model-based system (Chan & Paulson 1987). A rule-based decision process can then be used to select the appropriate form of the equation at any given time without having to use any computationally expensive numerical approximation techniques.

Sapossnek's second criticism of the parametric approach is more credible. The design process is, by its very nature, iterative. Much of the iteration which the user of a CAD system performs could be handled automatically by a suitably constructed system. So far, parametric systems have not had this functionality although, whilst difficult, it would not seem to be an impossible task to equip such systems with iterative techniques.

The Constraint-Based Approach

The constraint-based approach has received much interest at the Engineering Design Research Center at Carnegie Mellon (Westerberg et al. 1989) and has led to the development of systems such as Arbab and Wang's (1989) Operational Transformation Planning (OTP) and Sapossnek's (1989) Design Objects and Constraints (DOC). These systems, like the DESIGNER system, build up a dependency network but, with this approach, the resulting graph is not directional. That is, the graph is truly a declarative statement of the relationships between the design parameters and not a procedural statement about how to evaluate certain parameters. Furthermore, since this graph is not constrained to be directional, it is not constrained to be acyclic either and so iterative equations can be included within it. This has great benefits for the designer but at the expense of much heavier computation when evaluating parameters and, if one single factor is guaranteed to dissuade designers from using a system, it is a poor response time.

3.3.2. Design Automation Approaches

The main objective of these approaches is to understand and simulate the design process in a manner which will permit many of the decisions which must normally be taken by the designer to be taken by a computer system instead.

The Configuration Approach

The system variously called XSEL, XCON and R1 developed by McDermott (1982) is an expert system which configures VAX computer systems. Armed with a set of components, a requirements specification and a large knowledge base of rules it can configure a computer system to meet a customer's requirements. The approach has much in common with planning systems due to the fixed and finite number of components which can be configured into a system. Unfortunately, designs are not always derivable from a set of pre-determined components and this approach is only mentioned in order to indicate that it is already possible to achieve design automation within a restricted field.

The Redesign Approach

Dixon and Simmons (1983) point out that much real design work is in fact concerned with re-designing a previously designed artifact to remove flaws or amend it in some minor way to suit new circumstances. Their approach utilises a trial-and-error approach in which the original design is repeatedly modified and its various characteristics are measured against some target set until an acceptable result is obtained.

The Design Refinement Approach

Brown and Chandrasekaran (1983) have reported work based on the idea of refining a design. By examining what they call *Class 3* or *routine* design tasks they can produce designs automatically in some cases (Brown & Chandrasekaran 1985a & 1985b). This is possible when the constraints on the design are sufficient to reduce the design options to such an extent that the final choices become trivial or arbitrary. On less routine tasks the design system might be semi-automatic and require designer intervention only in particular situations.

Brown and Chandrasekaran have demonstrated their approach with a system for designing air-cylinders called AIR-CYL. The overall design is broken down into sub-tasks which are handled by *design agents*. These agents take a number of forms ranging from design specialists, which attempt to design a particular part of the overall artifact, to design steps which make a single design decision based on the current state of the design.

The Design Synthesis Approach

Maher (Westerberg et al. 1989) has developed a shell called EDESYN which is based on a design synthesis paradigm and has been used to develop a number of expert systems for engineering design. This approach involves searching through a tree of design combinations until each possible design which satisfies a set of design constraints has been identified. A method for evaluating the alternatives and making a selection is apparently under development. Unfortunately very few details are available about this approach at the moment.

3.3.3. Functional Design Approaches

Medland's (1986) concept is concerned with utilising the power of the computer to tighten up the process of designing complex artifacts. Currently an analysis of any major design will reveal many critical points where important information is lost, needless assumptions are made, or where the use of common sense is thwarted due to designers adopting too narrow a view of the overall design. A good example of the need for such an approach is the design of a gear-box for a motor vehicle. Whilst this is clearly a specialised task requiring the skills of a specific individual or team, the final design will have ramifications for the design of the passenger compartment and the transmission which will also be designed by specialists. Many problems result from this delegation of roles and the functional approach seeks to reduce them by clearly identifying the existence of semi-independent design specialists and, hence, facing up to the problems of communication between them. Medland has examined many aspects of this approach in a theoretical manner but has not yet developed a prototype which demonstrates them.

3.3.4. Summary

The two main initiatives with regard to the application of expert systems to design are distinguished by the degree of automation which they seek to achieve. Both of these initiatives are based on the assumption that there exists a complete knowledge base for the type of design they are considering. In the case of the design assistant approaches this knowledge base is expected to be distributed

between the expert system and the human designer. In the case of the design automation approaches it is intended to be held solely within the expert system.

Latombe and Medland have both emphasised that the assumption of a complete and closed knowledge domain (no matter how large the human contribution may be) is unrealistic. In practice, design requires inputs from many different specialisms and no single human or expert system designer can be expected to be conversant with all of them. Furthermore, the cross-fertilisation of ideas between these specialists is absolutely essential to the creation of a satisfactory end-product.

The functional design approach, which represents a departure from the conventional hierarchical representation of design, provides a framework which could be employed to 'open up' the closed world assumption of current systems and allow unforeseen expertise to influence a design. This idea is as valuable when applied to human-human communication in design teams as it is to human-computer communication.

Chapter 4

SPECIFICATION OF AN AID FOR ERGONOMIC DESIGN

4.1. User Definition

Like any other tool, the expert system has to be developed with due consideration to the user population, the task in hand and the environment in which the system will be used.

The user group at which the system is aimed, whilst being experienced designers in their own field, are not expected to have any great knowledge of ergonomics. They are, however, expected to learn from their experience with the system. This latent learning will produce a more enlightened user group, at least with regard to the system's terminology and methodology if not with the discipline of ergonomics as a whole.

A major constraint on the approach of the system is that it will be just one tool employed on a subset of the problems which define the task. In other words it is impossible for the system to have a complete picture of the design being undertaken. It must, therefore, be flexible enough to allow the user to make changes to the system's model which are not explicable in terms of the system's ergonomics knowledge alone.

The system is intended to be employed in large drawing offices and engineering consultancies which will almost certainly include a means of accessing computer resources. The last consideration - the environment in which the system will be used - can, therefore, be assumed to cause no problems which have not been investigated before.

4.2. The Ergonomics Design Process

It is in the nature of design that it is an iterative process broken up by conceptual leaps, so it is essential that any design aid does not thwart the creative potential of the more experienced designer. This point may not be obvious in the context of expert systems - which are generally held to be able to take over the more intellectual of human abilities. Indeed, it is far from clear that electronics design, for instance, which tends to boil down to the problem of optimising the links between black boxes, requires any creative input from the designer at all (Begg 1984). Similarly, R1, which configures computer systems, is able to impose so much structure on its problem domain that the need for user interaction is negligible (McDermott 1982). However, in systems where the design options are not so constrained, such as DESIGNER (MacCallum 1982, MacCallum & Duffy 1985) for ship hull design, the creativity of the user is of the utmost importance in pruning the search space of design possibilities. Ergonomics design falls into the latter category. There are no clear optimisation procedures and a dearth of constraints compared to the numerous options available.

Ergonomic design is a large and complex problem. As such, it is necessary to break ergonomics down into subfields. Some of these subfields alone are sufficiently complex to warrant the application of expert systems. See, for example, the work on manual materials handling (Karwowski & Ayoub 1984). We must not forget, however, that whilst breaking a problem down into subproblems is a valid approach, as Stefik et al (1982) point out, the interaction between the subproblems cannot be ignored.

When a designer sits down at a drawing board or computer terminal to start work on a new design he will have a number of constraints to work within. These initial constraints are likely to be fairly broad - consisting of the design specification and any inferences which his experience may permit him to make from the specification. We can consider the designer's requirements at this stage as consisting of a number of sets which represent the possible options for the major design parameters.

Warman (1978) has identified four processes which may act on a design in a CAD system. They are *reduction* of the design to an equivalent but more suitable form, *modularisation* of the total design into sub-units that may be used in other designs, *simulation* of the behaviour of the design under varying circumstances and *optimisation* of the design with respect to certain parameters. What all these processes have in common is their numerical nature and this invariably requires exactness.

A conventional CAD system cannot, therefore, allow the designer to reserve his options. He will be forced to make a number of, possibly arbitrary, decisions in order to present his requirements to the system in the highly specific form expected. Effectively, a large number of the sets of options will be reduced to single elements. Worse still, these restrictions will have to be applied wholesale to both those options with major effects on the design and those which are relatively insignificant. These parameters will thenceforth be treated as if they had equal importance. In Medland's terms (Medland 1986) the designer must have already completed the *primary* (conceptual) design stage and also have

investigated the *secondary* (analytic) stage before he approaches a CAD system.

Whilst the conventional CAD system can allow the designer to change individual design parameters, it cannot make this easy for him. Changing a single parameter, when everything else is fixed, is the first stage of a long and tedious process. Due to the interactions between the parameters a large number are likely to require amendment to keep them consistent with the single parameter which has been changed. This, potentially explosive, effect of change will therefore dissuade the designer from changing all but the worst of his original selections.

The motivation behind the development of CAD systems was to facilitate the process by which a designer evaluates a number of alternative solutions. By and large these systems have succeeded in providing an improvement over manual methods. Particularly in Medland's *tertiary* (production of manufacturing information) stage. What they do not do, however, is to provide the kind of improvement which computing technology currently affords them. As Spur and Krause (1985) state, *What is normally not possible with CAD systems is designing.*

This thesis presents an expert system which has been designed to provide ergonomics advice to industrial designers engaged in prelimiary non-routine design tasks. The main feature which distinguishes this system from all other developments known to the author is the importance placed on permitting the user to begin with rather vague assertions and gradually tighten them up until a final design materialises.

4.2.1. Modelling the Design Process

The actual process of design, whilst it has received much interest and generated even more discussion, has only recently been subjected to any truly rigorous analysis. As Coyne et al. (1990) point out, the *analysis - synthesis - evaluation* approach leaves far too many questions unanswered. Perhaps because the futility of attempting to formalise something as non-deterministic as *designing* has been all too obvious, those researchers concerned with its study have limited themselves to models of design which steer well clear of anything that might be called 'mathematical'.

Whilst the goal of a formal analysis to produce a mathematical model of a design which can in some sense be optimised might be unattainable, much can be learnt in the process of trying to build such a model. Coyne (1988) has examined design from a number of angles ranging from treating the artifact being designed as a theorem in predicate calculus through to generating a range of design grammars which prescribe rules governing the production and transformation of designs.

Coyne's formal approach reveals many factors which are very important in the evaluation of a design and, perhaps even more importantly, his methods provide an insight into the actual process of designing as well. However, Coyne too stops short of trying to formalise the ultimate aim of a design system, namely to optimise some objective function based on the design parameters in order to arrive at a 'best' design.

In Appendix A of this thesis a mathematical model of design is derived. This model is developed top-down with the aim of generating an objective function which permits a set of design parameters to be selected in such a way as to optimise their values collectively, rather than individually. This model, although it does not, indeed *cannot*, succeed in descending to the level of detail necessary to optimise a specific design, does cast light on a number of issues which are of great importance in optimising a design.

Firstly, it brings to the fore the problem that a 'good' design is the result of a particular mix of parameter values rather than a collection of values which are intrinsically 'good' in themselves. Such a set can only be derived in an iterative manner as the effects which each parameter value has on other parameters are successively manifested.

Secondly, it throws into sharp relief the fact well-known to designers, but rarely catered for in computer aided design, that what distinguishes different types of design is the *relative importance* attributed to each parameter. Why is a heavy goods vehicle so different from a saloon car? The answer lies solely in the requirement that load capacity is much more important with the former whilst comfort and speed are accorded more importance with the latter. This is not the same as saying that driver comfort is not a factor in the design of a heavy goods vehicle or that luggage capacity is not a factor in the design of a passenger vehicle. Clearly the contrary is the case.

Thirdly, the analysis in Appendix A identifies the one most helpful facility which a computer based design system can provide to a designer. Of all the evaluations required by the model, the most tedious for the designer, and yet the easiest to automate, is the calculation of the effects of a single change to a design parameter on all of the other parameters and the subsequent validation required to determine whether that change is acceptable.

The insight provided by this mathematical formalism has had a considerable influence on the design of the constraint refinement system described in the next section and hence on the ALFIE system which is based on it. The success of the ALFIE system is therefore offered as an empirical justification for the formalism developed in Appendix A.

4.3. The Constraint Refinement Paradigm

As described in the previous chapter, one way in which the power of the computer can be exploited to aid the design process is to use the computer to uphold the integrity of the large set of design parameters typical in industrial design applications. The paradigm proposed here is based on ensuring that a design conforms to a set of mutually consistent constraints on the design parameters. To give a very simple example, in the design of a writing desk, these constraints might include minima and maxima for the height and area of the writing surface. Any designs which did not comply with these constraints would be rejected and the user informed of the particular constraints which had been violated.

These constraints should not be identified with the constraint related approaches described in the previous chapter. Whilst there is much common ground, as will become clear later, one of the most significant aspects of the constraints described here is that they represent *vague bounds* on the values of the design parameters. For a historical pedigree the reader is referred to the field of *interval analysis* (Moore 1966, Nickel 1986) but with the caveat that, unlike the interpretation common in interval mathematics, constraints in no way imply any *error* in a parameter value. They merely assert that more than one value is acceptable and consistent with the known relationships between that parameter and the others that make up the design model.

Assertions of different types can be achieved by using fixed constraints (Eg. all lengths should be greater than zero), variable constraints (Eg. the length of a table-leg should always be less than the table-height, where both table-leg and table-height are design variables) or more exact relationships which should be maintained between the parameters of a design (Eg. physical laws). Clearly constraints have a very broad applicability - they might be drawn, for instance, from manufacturing capabilities, economic considerations or even contain an aesthetic component.

The second fundamental aspect of the constraint refinement paradigm is the concept of gradually tightening up the vague initial constraints on the design parameters as the design proceeds. Each parameter in a design will be required to comply with a number of relationships involving other design parameters. Where more than one relationship places a limiting range on the value of a parameter and

where those ranges are not identical then it is logical to conclude that it is the intersection of those two ranges which should be used to constrain the parameter.

By permitting any number of design relationships to freely participate in determining the acceptable range of values for a design parameter we can successively refine the, originally vague, values of the parameters of a design as more is learnt from the user about the design he is undertaking. Furthermore, we can also readily detect when an incompatible set of vague constraints has been requested by the user since the resultant range required for some parameters will be empty after all of the required intersections have been carried out. Nor are we limited to merely detecting inconsistencies for, by examining the relationships which led to the conflict, we can actually identify those parameter values which created the inconsistency.

For preliminary design, the constraint refinement paradigm is likely to be most profitably employed as part of a system which intervenes in the normal human directed design activity.

4.4. The ALFIE System

The ALFIE system is a hybrid created from the design assistant and functional design approaches reviewed in Chapter 3. Its design assistant element belongs to the parametric design class which was exemplified by the DESIGNER system. However, the ALFIE system differs considerably from the DESIGNER system and not solely in the non-hierarchical functional manner in which ALFIE handles its expertise (which is implemented as a network of concepts and discussed

further below and in Chapetr 6).

The ALFIE system is based on *vague* reasoning whereas the DESIGNER system employs *probabilistic* reasoning. The crucial distinction here is that ALFIE seeks to reason with ranges of values for design parameters all of which are *acceptable* whilst the DESIGNER system concentrates on reasoning with single values for design parameters all of which can have a margin of *error* (MacCallum & Duffy 1985). It is these different premises which lead the two systems to employ unique reasoning schemes and enables ALFIE to take advantage of the constraint refinement paradigm advocated in the last section.

The main objectives of the ALFIE system can be summarised as follows:

* *To guide the user to important ergonomic concepts;*
* *To constrain ergonomic factors by invoking appropriate models;*
* *To inform the user of unacceptable factor constraints;*
* *To advise the user of relationships between factors;*
* *To assist the user in making 'What if' speculations;*
* *To produce a summary of the final constraints on the factors.*

With ergonomic design we are not allowed the luxury of being completely 'in the know' with respect to the task which we shall be designing for. The dual aims of the system are therefore to provide the ergonomics novice with the detailed guidance he requires and yet allow the more experienced user the autonomy to move freely around the system examining such areas as he thinks fit when he thinks fit. In order to cater for both the expert and novice user the ALFIE system has been organised into a number of different levels.

From the user's point of view, the highest level is that of the various *concepts* into which the field of ergonomics is broken down. These concepts are connected into a *concept net* which links related concepts and provides a basis for the system to guide the user from a given concept (Eg. the thermal environment in which a task is to be performed) to more detailed concepts (Eg. the thermal comfort of the operatives carrying out the task) or more general concepts (Eg. the overall environment in which the task is to be performed) or just to other related concepts (Eg. the population characteristics of the operatives performing the task).

Upon examining a concept, the user triggers a number of *rules* which will normally engage the user in a dialogue concerning that concept. If other concepts are inextricably bound up with the concept being examined then examination of these other concepts will be initiated by the system. The user is not free to ignore relevant material although he may give very vague responses to questions arising from areas with which he is not particularly concerned. This vagueness however, will be manifested in the system's advice which may well be too general to be of much help to the user if he is exceptionally vague.

At its most detailed level, the ALFIE system contains an *influence net* which records the interdependencies of the various design parameters or *factors* as they shall be called from now on. The influence net is created by examining the relationships which are stated to exist between the factors. The most common statement of a relationship between one or more factors is presented as a *model* which states how one factor may be calculated from the values of other factors. The ALFIE system is structured to work with the intervals mentioned earlier and

the constraint refinement paradigm is employed throughout. For this reason it would not be strictly correct to identify a model with an equation since, although each model contains an equation, a model only imposes one of any number of constraints on the range of values which are permissible for the factor being calculated.

It should also be noted that the influence net is not fixed. It varies as more is determined concerning the nature of the design task. Ergonomics, as already indicated, is a patchy subject. Much of the experimental evidence used to identify the precise nature of relationships between factors cannot be generalised. For this reason, entirely different models may be required to evaluate a factor depending on the context in which the evaluation is required. For example, when attempting to determine whether an operative performing a task is likely to suffer from heat stress a number of excellent models are available. Each, however, is concerned with a very specific set of environment/operative/task characteristics. The models have very little in common and cannot, therefore, be combined into one single all-purpose model. The ALFIE system has to take account of the fact that only one of these models should be used and the one to be selected can only be identified after a certain amount of information has been gleaned from the user. Furthermore, some small amendment which the user may make to this information at a later date may well require that another model be used instead - thus necessitating a change in the network of influences.

Mediating between the concept and influence nets are the rules of the ALFIE system. Each rule consists as a number of <condition, action> pairs and behaves

like a miniature production system. The conditions of the rules are composed of logical expressions based on factors. When a condition can be shown to be true then the appropriate action is executed and, in this way the system forward chains through an investigation into a particular concept and, possibly, into further concepts which cannot be safely ignored.

When the ALFIE system attempts to evaluate a rule condition in which some of the factors are unknown, a backward chaining approach is employed in order to find some way of evaluating those factors automatically. If this fails then the user is asked for the appropriate information. In this way the number of questions which the user is required to answer is minimised.

As a result of the inferences which it can make, the system can produce a set of mutually consistent ranges on the factors pertinent to the design and also display a variety of pieces of advice concerning those ranges. The user can easily change the value of a factor or factors and receive a prompt response indicating the new set of mutually consistent ranges and a modified commentary on those ranges.

The precise workings of the ALFIE system are detailed in the following three chapters.

Chapter 5

KNOWLEDGE REPRESENTATION IN ALFIE

5.1. Overview

Within the ALFIE system we distinguish between three types of knowledge. These are *conceptual knowledge* of the potential for interaction between different subfields of ergonomics; *predicative knowledge* of the logical consequences of certain facts; and *mathematical knowledge* of the numerical relationships which must hold between certain design parameters. In order to represent this knowledge the ALFIE system is provided with four basic building blocks and two major associative networks which link these blocks. We shall examine the knowledge representation by discussing each of the basic blocks in turn.

5.2. Concepts

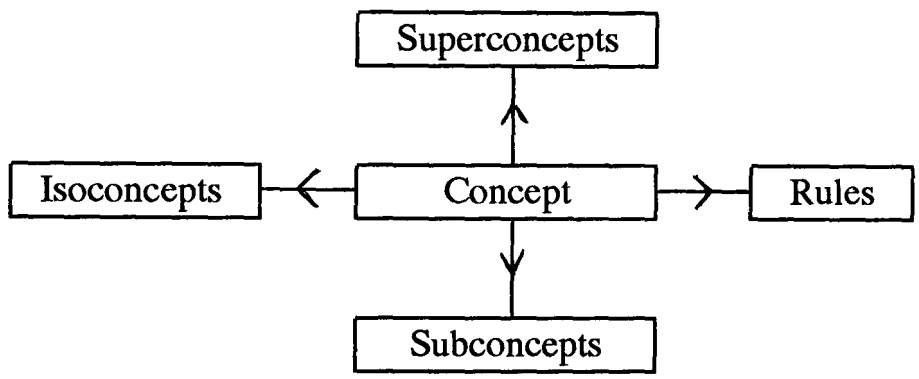

Figure 5.1 *The general structure of a concept.*

In order to break the domain of ergonomics down into manageable subfields the ALFIE system contains a network of *Concepts*. Each concept represents some body of ergonomic knowledge and is linked, through the network, to other relevant areas of ergonomics. Apart from assisting the knowledge engineer to visualise the system, these concepts permit the user to go straight to those areas

which he considers important without being forced through a chain of rules, some of which might be totally irrelevant to the current design task. The general structure of a concept is shown in Figure 5.1.

Attached to each concept are a number of rules. Rules are explained in detail below. For the time being the reader need only note that rules are used to test propositions and instigate a variety of actions. When the user decides to examine a concept, it is the associated rules which will present him with questions and determine the course which the consultation is to follow.

5.2.1. The Concept Net

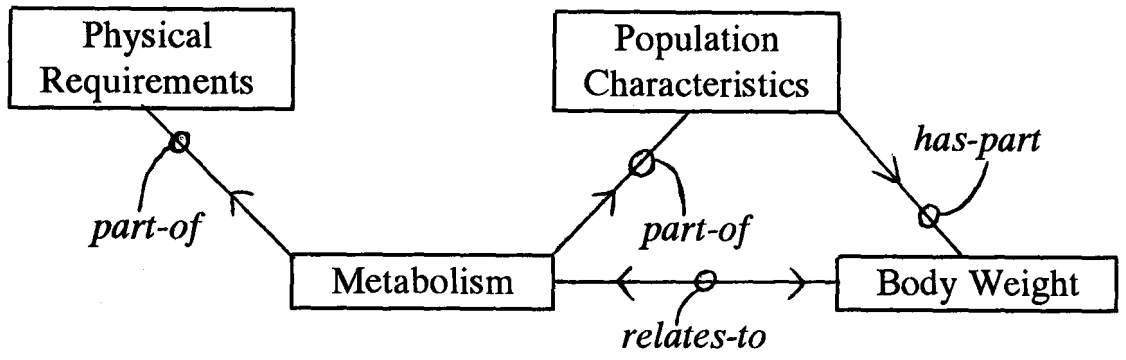

Figure 5.2 *Part of a concept net.*

The *Concept Net* is a semantic net formed by linking concepts with one of three types of connection. The *part-of* (or *super-concept*) connection is a pointer from a more specialised area of ergonomics to a more general area. For example, in Figure 5.2, the concept of "metabolism" is a part of the general area of "population characteristics". It is also a part of the general area of "physical requirements" of the task. The concept of "metabolism" would therefore be

linked to the latter pair of concepts by two *part-of* connections. The second type of link is almost a corollary of the *part-of* link. It is the *has-part* (or *sub-concept*) connection. In fact the existence of a *part-of* connection between two concepts in one direction would be sufficient for the system to create a *has-part* connection in the other direction. However, for pragmatic reasons this conclusion is initially supressed. We shall defer discussion of this point until we have dealt with the third type of link, the *relates-to* (or *iso-concept*) connection. This type of link permits related areas of ergonomics to be associated when it would be incorrect to state that either was a part of the other. For example, "metabolism" is highly dependent on "body weight" of the population but there is not really a *part-of* relationship between them in either direction.

The concept net is expected to become very large as new areas of ergonomics are added to the system. Before long it would be impractical for those adding new concepts to be fully aware of the links which exist between the concepts already present in the knowledge base. In particular, it would be quite possible for anomalous situations to arise from the manner in which new links were labelled. For this reason, any inferences which the system can make concerning *part-of* and *has-part* connections are deferred until after the user's labelling has been shown to be consistent. This will save the user from considerable confusion if his labelling ever has to be faulted. An example concept net is shown in Figure C.1 in Appendix C.

5.3. Rules

Within ALFIE, the term *rule* has a slightly different meaning to that which the reader might be familiar with. In order to assist the knowledge engineer in structuring a knowledge base, a number of <condition, action> clauses can be grouped together. It is these groups which are called rules within ALFIE and not the individual clauses. In addition to the conditional clauses, rules may also contain a default clause whose action part will be executed whenever none of the conditions in the other clauses could be shown to be true. A typical clause in a rule might be:

WHEN VIisual_angle < 0.83 AND Viewing_distance > 30 :

ADVISE The fault is too small. You had better magnify it in some way or resort to a measuring gauge of some sort.

Here **Visual_angle** and **Viewing_distance** are the names of factors (see below) and the action is to display some advice to the user. The full set of actions available for use by rule clauses can be found in the Knowledge Engineering Language Reference Guide in Appendix C under the section on defining a rule.

Associated with each rule are pointers to all concepts which trigger it. These pointers are used when the system has to take account of a change in the value of a factor which is relevant to the rule. The mechanism underlying this is left until the next chapter.

5.4. Models

Knowledge can just as easily come in the form of numerical relationships as it can in the form of rules. For this reason the ALFIE system provides a mechanism for incorporating mathematical knowledge in its knowledge base. This knowledge is accorded the same status as the predicative knowledge represented in the rules and is recorded in *models*.

Like rules, models determine some outcome based on an expression involving factors. Here the similarity ends however. The typical form of a model might be:

MODEL Calculate_visual_angle

EQUATION $Visual\_angle \leftarrow 120 \times arctan\left[\frac{Fault\_size}{20 \times Viewing\_distance}\right]$

A variety of mathematical functions are provided within ALFIE for use in model and rule expressions. In addition, it is possible for the knowledge engineer to write extra programs to perform unusual functions and these are called up in the same way as the internal functions such as *arctan* above.

As indicated in the previous chapter, it is possible for a model to be valid under a particular set of circumstances only. For this reason each model has a flag which indicates whether it is currently valid or not. Initially all models are treated as invalid and the knowledge engineer is expected to write rules which *enable* those models to be used. Subsequently, enabled models can be *disabled* if required. It is not uncommon for the knowledge engineer to wish to enable/disable a group of models *en masse* and each model may be linked to other models which are

concurrent with it for this purpose. Another common situation is for the knowledge engineer to wish to select one and only one model from a group of models. A list of *exclusive* may also be attached to each model so that if the model is enabled then all of the others can be disabled. Judicious use of concurrent and exclusive models permits complex statements about the manner in which a factor should be evaluated to be made. For instance, the AND/OR tree in Figure 5.3 can be created by identifying models *M1*, *M2* and *M3* as concurrent, models *M3*, *M4* and *M5* as exclusive, and models *M5* and *M6* as concurrent. This AND/OR tree is interpretted to mean that the factor concerned can either be evaluated by using models 1, 2 and 3; or by using model 4; or by using models 5 and 6.

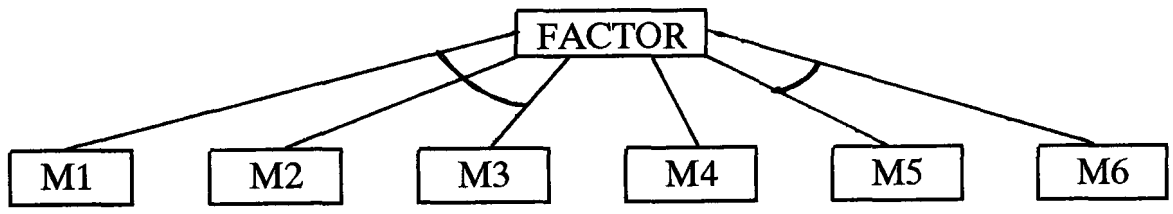

Figure 5.3 *An AND/OR tree relating models to a factor.*

5.5. Factors

The raw data of the ALFIE system is recorded in *factors*. The system provides for three different types of factor. Precise descriptions of these can be found in the Knolwedge Engineering Reference Guide in Appendix C under the section on defining a factor. For now, it will suffice to say that a factor can be *logical* in

which case it only represents boolean values; *numerical* in which case it can represent any real interval; or it can be a *set* of values selected from a superset previously determined by the knowledge engineer.

Numerical factors may, in addition, be provided with a set of overall bounds which limit the range of values which the factor may take. These bounds are displayed to the user whenever it is necessary to obtain the factor's value from him. The bounds can be general numerical expressions in other factors, thus permitting the range of a factor to be restricted differently under differing circumstances.

All factors contain references to those rules which require them in the condition part of a clause. They also contain a reference to any other factor which requires them in the determination of its overall bounds. In addition, each factor is also linked to all of the models that can possibly provide a means of evaluating it.

5.5.1. The Influence Net

The set of enabled models at any one time defines a network of influences between the factors. An *influence net* is generated every time a model is enabled or disabled. Effectively, each factor is made to point to all of the factors which are dependent on it given the current state of the models. This network permits the system to readily detect any 'knock-on' effects of a change in the value of a factor and to re-evaluate just those rules and models which might be affected by the change. Figure 5.4 depicts part of an influence net created from the following model which involves three factors:

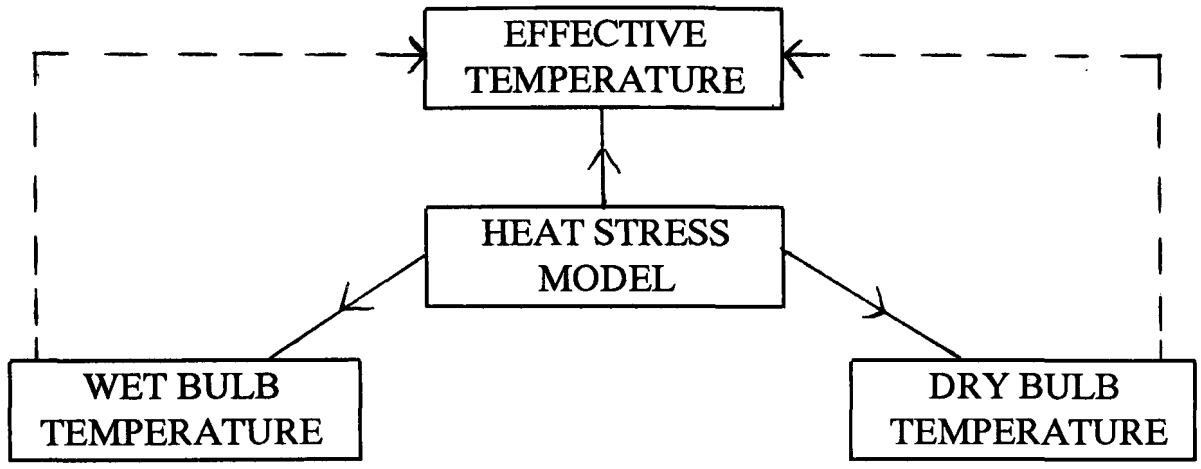

Figure 5.4 *An influence net (dashed line) between factors.*

MODEL Calculate_heat_stress

EQUATION *Effective_temp* ← 0.85 × *Wet_bulb_temp* + 0.15 × *Dry_bulb_temp*;

Chapter 6

THE ALFIE INFERENCE SYSTEM

6.1. Overview

As indicated in Chapter 4, the ALFIE system works at two main levels; that of the concept network and that of the influence network. This distinction leads to the two major inference techniques used by the system. The former employs a rule-based approach to forward chain through rules and concepts examining those areas of ergonomics which are relevant to the user's design. The second technique is based on a backward chaining approach which attempts to update the values of any factors which are unknown (or whose values have become uncertain) by invoking appropriate models. We shall call these two techniques *Rule Based* and *Model Based* respectively.

Whilst the real work of the system is accomplished by the rules and models, there is yet another (much simpler) form of inference used to guide those users who have little idea of which areas of ergonomics to examine. We shall briefly deal with this form of inference, which we shall call *Concept Based*, first.

6.2. Concept Based Inference

The ALFIE system incorporates a guidance system which uses the concept net to assist the user in selecting an area for investigation. Occasionally the system will be presented with a user who has no idea whatsoever which areas of ergonomics he should look into. Under these circumstances the guidance system will merely select the first concept defined by the knowledge engineer and instigate an examination of it. It is for this reason that special mention is made of this first or 'root' concept elsewhere in the thesis. The knowledge engineer is encouraged to

build a simple dialogue within the 'root' concept which can uncover the nature of the user's task and direct him to other appropriate concepts.

Normally the user will have some idea of which aspects of ergonomics are particularly important for him. By examining these he gives the system an idea of where his interests lie and if he subsequently requests guidance then the ALFIE system has a context in which to work. The concept net, as already described, contains links with varying semantics. A straightforward examination of the last concept investigated by the user will reveal the existence of any more detailed concepts in the same area. The guidance system will first attempt to guide the user to one of these. If none exist then any more general concepts which have not been fully examined will be tried. If this fails then other related concepts will be sought.

If no concepts directly linked to the last concept examined by the user are suitable then the system will backtrack to the concept which the user examined just prior to the last one. This whole process iterates until either a suitable concept is found or until all of the concepts examined by the user have been tested. At this point the system will start to look further afield - at concepts indirectly linked to concepts which the user has shown an interest in. Eventually every concept in the net will have been either discarded or selected.

Where more than one concept meets the search criteria (for instance, the last concept examined by the user might be related to two other unexamined concepts) the guidance system can examine a connection weight, optionally included by the knowledge engineer, in order to discriminate between them.

This is a suitable point at which to define precisely what is meant by "concept examination". A concept is considered to have been examined if all of the rules associated with that concept have been triggered and none of the factors used in the conditions of those rules have since been modified. If such a modification takes place then the rule concerned will be flagged to denote this. Additionally, any other rules which this rule triggers will be flagged as will any concepts which trigger any of the rules so flagged. In this way the system can determine whether any previously examined concepts require re-examination due to a change in circumstances.

6.3. Rule Based Inference

The forward chaining rule based inference procedure permits the system to enable and disable models, update factors, provide textual advice to the user, trigger other rules and examine concepts. Each rule will contain a number of conditional clauses and, optionally, a default clause whose action will be executed whenever *all* of the conditional clauses were found to be false. The rule based inference system takes each clause in turn and attempts to 'prove' that its condition is true. The proof employs the backward chaining model based inference system described below. For each condition which can be shown to be true, the rule based inference system will execute the associated action before proceeding to evaluate the next clause.

Whenever a rule action leads to the enablement or disablement of a model a number of tasks must be performed:

1. The concurrency and exclusivity lists associated with that model must be used to change the status of any other models linked to the current model;

2. Factors which any of those models evaluate must be flagged as having a value which is no longer certain;

3. Any factors dependent on these factors (as indicated in the influence net) must also be flagged as uncertain;

4. The influence net must be re-generated to represent the factor dependencies resulting from the new set of model statuses;

5. Any rule which references an uncertain factor must be flagged to indicate that it should be triggered again;

6. Any factors dependent on models which might be enabled or disabled by such a rule should be flagged as uncertain and task 5 should be repeated;

7. Any concepts that require a rule of the type mentioned in task 5 should be flagged as requiring re-examination.

6.4. Model Based Inference

The backward chaining model based inference system is used to update the values of factors. It is invoked whenever a value is required for an uncertain factor. This situation can arise in two ways. The first occurs when a rule condition is being proved and that condition relies on an uncertain factor. The second occurs when an uncertain factor is used in a model to evaluate another factor.

The system employs the following algorithm:

1. Find all of the models which contain the factor on their left hand sides (these might contribute to the factor's value);

2. Attempt to enable each model in turn (we shall look at this step in detail in a moment);

3. If one and only one model can be enabled then the new value of the factor is the range produced by the model intersected with the overall bounds on the factor;

4. If more than one model can be enabled then the new value of the factor is the range produced by intersecting all of the ranges produced by the models with the overall bounds on the factor;

5. If no model can be enabled or if the intersection resulting from steps 3 or 4 is null then consult the user.

Step 2 in the above requires the system to examine the rule base and search for those rules which enable or disable the model or a concurrent one or an exclusive one. The usability of these rules may be conditional on clauses in other rules (if they are only triggered by another rule and not by a concept). These other rules must be used to determine whether the current rule can be triggered. Similarly, those rules might be dependent on yet other rules, and so on. Eventually all rules will be traced back to concepts which do not trigger rules in a conditional manner and so a particular enablement or disablement can be shown to be true.

Fortunately the inference system only needs to prove one disablement or enablement in order to determine whether the model can be used. If, as a result of some inconsistency in the knowledge definition, the system finds that both enablement and disablement of a model can be proved, then the model is not used. This is the safer course within the domain of ergonomics where the applicability of the empirical evidence is continually being refined - once a model has been shown to be invalid in a given context, it should not be used no matter how often it has been supposed to be valid in the past.

Step 4 in the previous algorithm enshrines the refinement principle of ALFIE. This is the stage where a number of models may constrain the value of a factor beyond any vague value which the user might have originally specified.

Chapter 7

INTERACTING WITH ALFIE

7.1. Overview

There are two types of interaction which arise with the ALFIE system. The first occurs whilst a user is employing the system to assist in a design. Such a user communicates through a standard *user command language*. The second form of interaction occurs when a knowledge engineer imparts knowledge to the system. This is achieved by means of a *knowledge engineering language*.

7.2. The User Command Language

The user is expected to interact with the system via a simple command language whose primary purpose is to enable the user to make conjectures and refutations concerning the design he is undertaking. In order to assist the user in understanding what the system knows about, a number of commands are included which list, describe and detail the various concepts, rules, models and factors known to the system. These are fully documented in the User Command Reference Guide in Appendix C.

The user can invoke the guidance system to discover new areas of the knowledge base which he should consider if he runs out of ideas. Otherwise he can elect to examine a particular concept and respond to any subsequent questions in as vague a manner as he likes.

If the user later decides that he wants to be less vague about one or more of his answers then he can determine the name of the factor which recorded his answer and refute its value. Having refuted all those factors which he no longer accepts

he can then request the system to accommodate this fact. Any factors which he has not provided values for will then trigger requests for new values and any new advice will be displayed.

At any time the user may obtain a complete list of the contraints on those factors which are relevant to his application along with any recommendations which the system may make based on those values. At the end of a session this information may be filed for later use.

The system incorporates an explanation system which will be invoked whenever a '?' is typed in response to a question. The explanation takes a number of forms depending on which type of question was being asked. The major forms, those which explain *what* the system is doing and *why* it is doing it are detailed in Appendix C.

At any time, except when engaged in a system led dialogue, the user may save the complete knowledge base along with any information concerning the status of the current session so that he can return at a later date and continue the consultation where he left off.

The current user interface is somehwat rudimentary and potential improvements in this area are discussed in Chapter 11. One obvious improvement, that of using a hypertext approach to the user interface has, I understand, already been investigated at the University of Nottingham (Connolly 1990).

7.3. The Knowledge Engineering Language

Having examined the architecture and user interface to the ALFIE system, we shall now consider the task of knowledge engineering - ie. getting knowledge into the system. The problems inherent in actually acquiring the knowledge required for the system are general to all expert system developments and are considered at length in Chapter 9.

For now, we shall assume that the knowledge has been extracted from whatever sources were available and is in a suitable form to be added to the system. This is achieved through a knowledge engineering language developed specifically for the ALFIE system. Precise details of the language are presented in the Knowledge Engineering Guide and the Knowledge Engineering Language Reference Guide in Appendix C and an example of its use is given in Appendix E.

Recall that one of the aims of the system was to ensure that the task of creating knowledge modules was not so complicated as to inhibit a domain expert from creating them on his own. The knowledge engineering language developed for ALFIE therefore has a very simple declarative structure. There are, in fact, no more than thirty different types of declaration which the knowledge engineer can make in the language. The compiler has also been designed to be as 'unpedantic' as possible. The wrong type of punctuation or extra punctuation; upper and lower case anomalies; and synonymous objects are all taken in the compiler's stride as long as no ambiguity results.

The builder of a knowledge base must first break his domain down into a number of concepts and specify the relationships between these concepts. Later he may need to link these concepts into a larger system in which case he will need to:

a). Indicate the relationships between the new concepts and those already present;

b). Ensure that no factors or models are duplicated and that no concepts, rules, models, or factors have the same name.

Next the user should construct his rule base in a top-down fashion. That is, he need not worry about where the knowledge is going to come from. When he has done this he will be in a position to define the factors which he has used. Finally, he can now add any models which relate the factors to each other.

Having written and debugged a knowledge base the knowledge engineer may compile it by loading it into the ALFIE system and saving it straight away.

Chapter 8

IMPLEMENTATION OF ALFIE

8.1. Hardware Considerations

The reader's indulgence is requested whilst the historical context in which the system was originally developed is outlined. Today, in 1990, it is easy to forget the hardware limitations which existed in 1980 when the majority of implementation decisions concerning the ALFIE system had to be taken. Were the system to be developed from scratch today a very different implementation would result.

The average mini-computer in 1980 was equipped with no more main memory and very little extra disc memory than the average micro-computer (IBM PC, for example) has at its disposal in 1990. In 1980, workstations were only just beginning to appear and were prohibitively expensive for most computer users. Personal micro-computers in 1980 were not suitable for any large-scale software system due to severe limitations both in processing power and memory size. It was clear, however, that micro-computers were developing sufficiently fast for them to potentially provide a platform at some, not too distant, future date.

Whilst the ALFIE system was to be developed as a stand-alone system, it was always the intention that it should be possible to link it with the ergonomics CAD system, SAMMIE, (Case & Porter 1980, Bonney et al. 1989a, Porter et al. 1990) and also with the robot workplace simulation system, GRASP, (Dooner et al. 1982, Taylor et al. 1982). Both of these systems were running on a Prime 400 mini-computer under the Primos IV operating system. It seemed only logical, therefore, that the ALFIE system should be developed on the same machine. This machine has since been upgraded to a much more powerful Prime 9650 and the

ALFIE system was easily ported onto it.

Before moving on from the hardware considerations it is necessary to discuss one aspect of the architecture which had very serious consequences for the software considerations treated in the next section. This is the limited amount of main memory available on the machine (less than half a mega-byte at the commencement of the work). From the start it was obvious that the knowledge base for ALFIE would be much larger than the amount of main memory available. This clearly meant that paging and the possibility of 'thrashing' were going to require some careful thought. Note that for a knowledge base whose size is n times that available in main memory then something in the order of $\left[1 - \frac{1}{n}\right] \times 100$ % of the knowledge base accesses could generate a page fault.

8.2. Software Considerations

The major software decision to be taken was the choice of programming language. Whilst Prolog and Lisp were available, they were not well supported and, in 1980, provided no mechanism with which to control the physical organisation of the knowledge base on disc. This latter factor meant that it would be impossible to minimise the paging requirements of a large knowledge base. On top of this, the compilers available did not produce anything like optimised code for Prime machines. It was therefore decided that, in the interests of speed and efficiency, a procedural language would have to be used. Of those available, Pascal had the richest set of control and data structuring facilities.

In order to facilitate the porting of the ALFIE system to other machines a complete set of I/O facilities were needed to replace the machine dependent and non-standard ones provided with the Prime compiler. It should be pointed out that there was no standard for Pascal I/O at that time and that Prime compilers now comply with the standard.

In order to implement the interval arithmetic needed by the ALFIE system a large library of mathematical functions based on intervals was built. The functions covered are listed in the Knowledge Engineering Language Reference Guide in Appendix C.

8.3. Other Considerations

Whilst the main ALFIE system was naturally designed in a modular manner, what may not be so obvious is the reasoning behind developing the concepts in the knowledge base in a similarly modular manner. This decision was based on two considerations.

Firstly, it was appreciated that users of the system might well wish to pick and choose which areas of ergonomics they wanted assistance with. The modular approach permits their selection to be directly translated into a list of concepts which should be included in the knowledge base. The possibility of mounting the system on a micro-computer also meant that a large knowledge base would not be practical and that selectively choosing areas within the overall knowledge base would be necessary in order to implement the system on a smaller machine.

Secondly, the amount of information associated with a concept could be very large. Just as programmers employ a modular approach to assist them in the management of a large software design, so knowledge engineers would also require some such mechanism to assist in the creation of the knowledge base.

Chapter 9

KNOWLEDGE ENGINEERING

9.1. The Knowledge Acquisition Process

Probably the greatest, but least documented, of the problems encountered in the development of an expert system is that of knowledge acquisition. Whilst the validation of the information used is clearly an essential and exacting process, capturing the expertise in the first place is by no means simple and the choice of appropriate efficient techniques is not clear-cut.

It is now over twenty years since work started on the DENDRAL system for inferring plausible chemical structures of unknown compounds from mass spectrographic, nuclear magnetic resonance and other chemical data. During this time much effort has been channelled into developing a wide range of expert systems. A rather disturbing statistic, however, is the small number of such systems which are actually in use:

As of September, 1984, the number of fully operational expert system applications in regular use under field conditions is probably no more than ten.

(Reitman & Weischedel 1985)

Even allowing for the fact that this statement is now five years old, given the fever of activity in expert systems development over the past decade, we are entitled to ask the question: Where are the fruits of all the labour? Reitman and Weischedel give us a hint when they proceed to inform us that in 1984 there were between 100 and 200 other expert systems in *some stage of development*. This suggests, and confirms my own experience whilst developing the ALFIE system, that it is exceedingly common to under-estimate the resources required to build a

viable expert system. So just where are these errors of judgement being made and why?

9.1.1. Some Stumbling Blocks

The evolutionary nature of the development of an expert system is summarised by Buchanan et al. (1983) thus:

1. Identify the problem characteristics;

2. Find concepts to represent the knowledge;

3. Design a structure to organise the knowledge;

4. Formulate rules to embody the knowledge;

5. Validate the prototype system;

6. If need be start again from an earlier stage.

Stages 1 and 2, although they may take some time to accomplish, are not likely to result in any surprises so far as the amount of effort required to achieve them is concerned. Stage 3 is the point at which a proprietary expert system shell might be purchased and it is essential that such a shell is suited to the domain under consideration. A badly chosen shell may well turn stage 4 into a mammoth exercise in accommodation if the true nature of the domain is more conducive to another approach. It is probably worth noting here the existence of model-based approaches to expert systems as well as the more conventional rule-based approach. See Johnston (1986) for instance. Stage 5 should be, like stages 1 and 2, relatively problem free, even though it will require a significant proportion of the overall effort.

Stage 6 enshrines the evolution principle. It can in fact be applied at any time and will be applied quite frequently - even if the software designer, knowledge engineer and domain expert are one and the same person. The evolution principle is not unique to expert systems. It appears wherever something is being created or modified. Unfortunately, such unpredictable iterations are very hard to cost in terms of man-power requirements which may well be much greater than anticipated. The problem is compounded by the fact that a proprietary expert system shell may have to be thrown out as part of this iteration with a consequent increase in cost. Perhaps some of those systems under development have been *temporarily shelved* due to a lack of resources and are not under active development at all.

Throughout the above development process there is also the underlying problem of actually obtaining the expertise needed to 'arm' the system. Because knowledge acquisition is not mentioned explicitly, as is all too often the case in reports of expert system developments, the new-comer may well be lured into thinking that it is a relatively trivial problem compared to the others. This is far from being the case (Fox et al. 1985, Welbank 1983).

9.2. The Knowledge Acquisition Bottleneck

Buchanan et al. (op. cit.) report the existence of a knowledge acquisition bottleneck between the knowledge source (domain experts and/or definitive works) and the expert system knowledge base. This will exist for at least as long as the representations used by the knowledge source cannot readily be mapped

onto those used by the expert system. However, it is important that the knowledge engineer does not exacerbate this mapping problem through a lack of awareness of the subject matter or terminology employed.

Unless the knowledge engineer is himself the domain expert, the first stage in knowledge acquisition must be for him to familiarise himself with the domain under consideration. Grover (1983) proposes the production of a Domain Definition Handbook which should contain the following:

A general problem description;

A bibliography of reference documents;

A glossary of terms;

A list of domain experts;

A list of performance metrics;

Examples of reasoning scenarios.

As Grover points out, the production of the handbook can be speeded up by having access to an expert at this stage. However this must be set against the amount of time which the expert is prepared to devote to the project. Certainly, the expert will be invaluable in reducing the potential bibliography to those references which provide up to date, authoritative information on the domain but he cannot be expected to provide answers to each and every query which the knowledge engineer might encounter whilst studying the references. Having obtained a selective bibliography, a glossary of terms and a provisional list of domain experts should not be difficult to produce. Short communications with those people on the provisional list should enable the knowledge engineer to

produce his working list.

Before proceeding however, I would recommend that the knowledge engineer re-examines his problem description in the light of what he has learnt from studying the bibliography. The original proposal may now appear to be unfeasible given the state of the art of the domain itself. A less ambitious objective may still be a worthwhile subject for the expert system and, if necessary, this should be investigated. Once the knowledge engineer and domain experts have agreed on a feasible and useful domain for the system they can consider the performance criteria required of the system and produce some examples of its use.

9.3. Sources of Knowledge

There are two major types of source from which knowledge of a domain might be extracted. These are the literature, comprising the bibliography mentioned earlier, and human experts. In general, the literature on a subject will be more structured than a transcript compiled from an interview with a human expert. It will also allow the knowledge engineer to acquaint himself with the basics of the domain without wasting the time of an expert. The literature is therefore a more suitable starting point for the knowledge acquisition process. The main drawbacks of the literature are that it is not interactive and it may well be incomplete. In order to fill in the gaps left by the literature the knowledge engineer will have to approach human experts.

Not all developments will require the use of both types of knowledge source. If the literature is sufficiently detailed and complete (but so diverse that an expert system is still required to collate the information into a usable package) then a human expert may not be needed until the validation stage. On the other hand, if the literature is too scarce or idealised then the human expert will be required from a very early stage.

The knowledge engineer should approach a human expert with as clear a set of objectives as is possible. This may seem obvious. However, the objectives should include an interviewing technique which has been carefully thought out to ensure that the interview does not go 'off the rails' into areas which are not of immediate concern to the knowledge engineer.

A number of knowledge acquisition techniques have been documented ranging from the study of textbooks through interviewing and questioning techniques to automated learning and text understanding systems. This is not an appropriate volume in which to describe all of these techniques in detail so a summary of some fifteen techniques which contains references to sources where full descriptions may be found is provided in Table 9.1. A large proportion of the approaches cited have been employed differently by different people and have consequently been given different labels by those people. I have attempted to show how these different labels equate with each other.

With regard to some of the more esoteric techniques, it should be pointed out that sometimes the expert being interviewed suffers from not being sufficiently aware of the purpose of the interview. As Schweickert et al. (1987) demonstrate, a

	Fox et al. (1985)	Buchanan et al. (1983)	Grover (1983)	Welbank (1983)
1		Textbooks		
2	Informal Interviews		Forward Scenario Simulation	Interviews
3	Protocol Analysis		Procedural Simulation	Observing Experts
4				Critical Incidents
5				Repertory Grids
6				Distinguishing Goals
7			Pure Reclassification	Reclassification
8				Systematic Symptom-to-Fault
9				Intermediate Reasoning Steps
10			Goal Decomposition	Dividing the Domain
11	Interactive Computer Techniques	Intelligent Editors		Questioning with Shells
12				Using Half-Built Systems to Elicit Further Knowledge
13	Rule Induction	Inference Systems		Induction Systems
14	Heuristic Discovery	Automated Learning		
15		Text Understanding Systems		

Table 9.1 *A Summary of Knowledge Acquisition Techniques.*

straight-forward interview can often be the best way to generate rules from an expert since he is fully aware of the real goal of the exercise and not 'blindfolded' by some intermediate goal such as sorting cards, etc.

9.4. The Art of Knowledge Engineering

The author has collaborated with ergonomists and psychologists from the University of Nottingham and Purdue University in experimental evaluations of different knowledge acquisition techniques (Schweikert et al. 1987). As a result of this work, and also his involvement in training knowledge engineers to work on the ALFIE system, the author has identified a number of problems which suggest that the task of knowledge engineering should not be taken lightly.

In an early experiment in knowledge engineering the author and Dr Richard Schweikert of Purdue University jointly interviewed an expert on lighting techniques for inspection tasks. They then independently extracted rules from the transcript of the interview. After discussion 61 rules were agreed upon. Of these only 13 were initially identified by both interviewers. A further 40 were identified by one of the interviewers but not by the other. Furthermore, on 14 occasions the two interviewers extracted different rules from the same statements in the transcript.

Of the 61 rules agreed upon by the interviewers only 59% could be used in an ALFIE knowledge base. In a later experiment carried out in conjunction with the Department of Psychology at Nottingham University an even higher rejection rate was recorded. Following interviews with four experts, 312 rules were identified

but only 127 (41%) could be formalised for use in an ALFIE knowledge base.

These figures indicate that there are clear problems associated with knowledge engineering. Barring any major revolution in the application of automatic knowledge acquisition systems, the interviewing of experts (or reviewing of literature) is certain to remain the primary method for gaining knowledge for expert systems. The wider application of expert systems is therefore likely to be limited by two main problems.

The first problem which needs to be addressed is that of training knowledge engineers in the extraction of information from textual material. It is quite obvious from the first experiment that an untrained knowledge engineer can easily miss a large amount of information even if that knowledge engineer is fully conversant with the knowledge domain. More worrying still is the apparent ability of such a knowledge engineer to extract an incorrect rule. In the first experiment the final rule set was presented to the expert for approval. The expert modified 33% of the 61 rules derived by the interviewers before accepting them and actually turned 3% down as blatantly incorrect.

The second problem is that of rule rejection rates. In both experiments these were high. One reason for this was that, for the purposes of the experiment, every possible rule was extracted irrespective of whether it was considered useful to the domain or not. In a non-experimental situation many possible rules can be ignored since it will be obvious that they can provide no useful information to an expert system. For instance, rules of the form

Chapter 10

CASE STUDIES

PAGE
MISSING
IN
ORIGINAL

10.1. Ergonomics Design Knowledge Bases

A large number of knowledge bases have been created in the general area of ergonomics ranging from small exemplar systems to five major concepts which have been incoporated in the default knowledge base of the system which is described in Appendix C. Four of these major knowledge bases were created with minimal contributions from the author. The author claims no credit for these but will use them to present examples of the two categories of reasoning for which the ALFIE system was primarily designed.

The one ergonomics knowledge base created solely by the author after consultation with two experts, one from Purdue University and one from Technica Ltd, is concerned with selecting lighting techniques for inspection tasks. The knowledge acquisition for this knowledge base became part of an experiment investigating the appropriateness of different techniques for knowledge elicitation which was carried out in conjunction with members of the Psychology departments of Nottingham and Purdue Universities (Schweikert et al. 1987).

Some areas of ergonomics are amenable to a considerable amount of mathematical formalism. The majority, however, are not. Where numerical models are sparse a *qualitative reasoning* approach is required and a large proportion of the knowledge is recorded in rules. Where sufficient experimentation has resulted in a high degree of formalism, on the other hand, much of the knowledge can be represented as models and a *quantitative reasoning* approach can be adopted. We shall now examine two ergonomics knowledge bases in detail; one from each of these two classes.

10.1.1. Qualitative Reasoning

We shall use the inspection lighting knowledge base, whose definition can be found in Appendix E, as an exemplar of a knowledge base which employs predominantly qualitative reasoning.

The object of this knowledge base is to form a list of suitable lighting techniques for a user-described inspection task. A browse through the definition of this knowledge base will reveal that models can be extremely useful, even in qualitative domains, for collating sets of results.

This knowledge base results in a somewhat 'chatty' dialogue with the user due to the personality of the expert who provided the knowledge. Odd pieces of advice are given to the user all the way through the dialogue as and when they are appropriate. No attempt has been made to collate all this advice and present it in one piece at the end of the consultation. This is not too important since the ALFIE system will always produce a summary of all the advice that the system ever gives when the user terminates the session.

Eighteen different lighting techniques which could be used in inspection tasks were identified by one of the experts who also provided some rules governing their selection. Most of the rules, however, came from a second expert. The approach adopted was to initially accept that all eighteen techniques were appropriate and then to start reducing the list by eliminating those found to be unsuitable. For this reason there are many models of the form *eliminate_X* in the knowledge base where *X* is some form of lighting technique. The function of each

of these models is to remove one possibility from the list of options.

The system adopts a forward chaining approach, exploring as many aspects of the inspection task as could be extracted from the expert and enabling models to eliminate lighting techniques as it goes. Appendix D gives an example of a typical dialogue with this knowledge base. The reader will note that the list of possibilities is eventually examined in a rule called RECOMMENDATIONS and a brief description of each suitable technique is given to the user.

10.1.2. Quantitative Reasoning

We shall use the thermal comfort knowledge base as an exemplar of a knowledge base which employs predominantly quantitative reasoning.

This knowledge base is built around a well-formed set of equations derived by an acknowledged expert in the field of thermal comfort, P.O.Fanger (1970). There is little of a heuristic nature in this knowledge base, the main problem for the knowledge engineer being to devise a scheme which would permit the user to take advantage of the vagueness implicit within the ALFIE shell and yet still gain useful advice from Fanger's equations. Much of the rule based aspect of this knowledge base is concerned with checking that the user has actually given values to factors which are indispensable to the equations.

One other problem which this knowledge base presented was that Fanger's equations are iterative. The ALFIE system does not provide for iteration in the solution of equations for obvious reasons. This knowledge base therefore employs a very useful feature of the system which enables the knowledge engineer to write

a separate program in any suitable high-level language to perform such a calculation. This program communicates with ALFIE through files and is invoked by treating it as if it was a standard ALFIE function (just like sin, cos, tan, etc.). As long as a run-file with the same name as the function is found in a specific directory then the function will return the required result. See the System Administrator's Guide in Appendix C for more details.

10.2. Another Type of Knowledge Base - Diagnosis

Finally, we shall examine a knowledge base created by the author from a manual providing advice on television reception (Department of Trade and Industry 1985). The aim of this knowledge base is to diagnose faults in television receiver circuits.

Whilst ALFIE was developed as a design expert system, its structure is sufficiently general to permit the system to capture and reason with knowledge from other types of domains. The TV reception diagnosis system employs the rule based inference approach within the ALFIE system to progressively prompt the user to examine the aerial, feeder cable, and splitter boxes connected to a TV set before digging deeper into the circuitry of the tuner, etc.

The following is a typical rule from this knowledge base:

RULE CHECK_VOLTAGE

DESCRIPTION Check the terminal voltage. If it is OK check the receiver.@

WHEN NOT GIVEN(TERMINAL_VOLTAGE) : COMPLY CHECK_RECEIVER

WHEN TERMINAL_VOLTAGE>=50 AND TERMINAL_VOLTAGE<=60 : COMPLY CHECK_RECEIVER

WHEN TERMINAL_VOLTAGE>60 : ADVISE The terminal voltage is too high - fit an attenuator to bring it down to below 60 dBuV.@

WHEN TERMINAL_VOLTAGE<50 : ADVISE The terminal voltage is too low - it should be at least 50 dBuV. Either,

a). Change the position of the aerial, or

b). Fit a larger (higher gain) aerial, or

c). Fit an aerial pre-amp.@

This knowledge base actually includes no models at all. This is due to the fact that the knowledge base is tracing the fault from the aerial to as deep a level as necessary. At each stage the system asks the user to physically examine particular parts of the receiver. Everything which the system needs to know must come from the user and there is no point in trying to calculate things which the user claims not to be able to measure.

Chapter 11

EVALUATION OF THE ALFIE SYSTEM

11.1. Benefits

By employing the ALFIE concept net to permit a functional approach to design it has proved possible for a number of different knowledge engineers to develop interacting knowledge bases (Pemberton 1985, Simpson 1987a, 1987b & 1987c, Connolly 1988, Simpson 1988a & 1988b) without any major problems. As Table 11.1 indicates, it has also been possible for knowledge engineers to draw on and enhance earlier work in the same area. This would have been much less common if a more rigid approach to design had been adopted.

Knowledge Base	**Knowledge Engineer**	**Year**
Strength Analysis	I Pemberton	1985
Heat Stress	G J Boggs	1985
Heat Stress	M R Simpson	1987
Lighting Levels	G J Boggs	1985
Lighting Levels	N P Milner	1986
Lighting Levels	A Connolly	1988
Metabolic Loads	G J Boggs	1985
Metabolic Loads	M R Simpson	1987
Inspection Lighting	R Schweickert	1986
Inspection Lighting	N K Taylor	1987
Inspection Lighting	M R Simpson	1987
Thermal Comfort	M R Simpson	1988

Table 11.1 *Knowledge Bases and Knowledge Engineers.*

The provision of model based as well as rule based reasoning within the ALFIE system has also reaped many rewards. A number of the knowledge bases would have been very difficult to implement without the model based reasoning facility. In particular the Heat Stress, Thermal Comfort and Metabolic Loads evaluations required the application of continuous functions which would have been

impossible to implement within a conventional rule based system.

The vague reasoning system employed by ALFIE has been shown to identify solutions to design problems which could easily have been missed if a less flexible approach had been used. Pemberton (1985) felt that the ability of the ALFIE system to maintain this flexibility until the last possible moment held much promise for ergonomic design. The constraint refinement paradigm, based on vague reasoning, was seen as having an applicability well beyond ergonomic design by a consultant commissioned by the ACME Directorate of SERC to evaluate the system (BYG Systems Ltd 1986).

The development of the ALFIE system also benefits the conventional discipline of ergonomics. In constructing knowledge bases in specific areas, gaps in the ergonomics data base have been uncovered. An ALFIE knowledge base attempts to be both complete and sound in some small specialist domain of ergonomics. On numerous occasions the *state-of-the-art* in ergonomics has been found to be incomplete. Given the uncoordinated manner in which ergonomics has developed it is not surprising that gaps in the theory exist. These gaps are difficult to detect and by identifying them ALFIE performs an unexpected service to the ergonomics community. Furthermore, the soundness requirement of ALFIE knowledge bases has also uncovered areas where the standard ergonomics wisdom is inconsistent. For example in the pre-requisites for the application of different heat stress models.

11.2. Drawbacks

Vague reasoning can lead to an explosion in the space of design solutions. By its very nature vague reasoning works to increase the bounds on variables. This tendency towards a combinatorial explosion is worrying and, whilst the constraint refinement paradigm mediates against it (very successfully in the author's experience), it is quite clear that there could well exist situations when vague reasoning cannot be employed. Fortunately, single-valued algebra is a proper subset of interval algebra. The ALFIE system can therefore be reduced to deterministic reasoning in a sound manner if necessary although the benefits which would derive from employing the system in this way are highly questionable.

The lack of integration with a geometric modeller and graphics display has been mentioned by a number of people who have used the system. This is acknowledged as a major drawback by the author and is seen as the next major initiative in the development of the system. Further discussion of this issue is left until Chapter 12.

The user interface of ALFIE is primitive. The structure of the concept network has much in common with the developing field of hypertext. The manner in which a user interacts with ALFIE could be significantly improved by adding a hypertext front end to the system. Johnstone (1990) has undertaken a pilot study into the design of a hypertext interface for the complete interaction with the inspection lighting and illumination knowledge bases which employs approximately 150 screens of hypertext and the results appear to be very

promising.

The knowledge engineer's interface with the system could also be enhanced through the provision of a rule editor and an analogous model editor. Ideas could also be borrowed from rule induction systems which are becoming more realistic alternatives to the conventional literature and expert based approaches.

11.3. Summary

The ALFIE system has been used and evaluated by more than a dozen people over a period of five years. The majority of these people have been ergonomists and they have often gained a greater insight into their discipline as a result of working with ALFIE. The general feeling is that the approach adopted holds much promise but that interfacing with a CAD system and an improved user interface will be necessary if the system is to find a niche in the market place.

Chapter 12

FUTURE WORK

12.1. Applications of the ALFIE Shell

In the previous chapter it was shown that the ALFIE system can be used in domains other than design. The author intends to examine other domain types (planning for instance) with a view to building a picture of just what the ALFIE shell can handle.

It would be wrong to assume however, that the shell can even handle all types of design problems. It was originally hoped that the system would be useful for industrial design problems in general and not simply those concerning ergonomics. Whilst ergonomics design does have much in common with other forms of industrial design, much more effort will be required in order to assess its applicability to, say, mechanical design.

The early decision that a probabilistic component would not be suitable for the user group being considered has strongly influenced the type of knowledge which the system can handle. There are some aspects of ergonomics, risk assessment for instance, which are just as much a branch of statistics as they are of human factors. Such fields can only be treated in a probabilistic manner. Many other applications could be considered if a suitably educated user group (recall the warning offered in Chapter 3) and an inference system with a probabilistic nature were available.

The author has already begun to derive a mechanism for dealing with probabilities in conjunction with interval variables and the initial results are quite promising. The *probabilistic intervals* which result have much in common with

confidence intervals. However, whereas a confidence interval is a statement of the form *"it is 95% certain that the value of design parameter Z will lie in the range [x,y]"*, a probabilistic interval will make a statement of the form *"it is 95% certain that a range of [x,y] is acceptable for design parameter Z"*. I trust that the reader will agree that the latter is a much more useful predicate for somebody designing a bridge, for instance.

12.2. Integration of Expert and CAD Systems

The author has investigated the possibilities for truly integrating knowledge based and CAD systems since completing the ALFIE shell. Bonney, et al. (1989b) considered the mechanism which would be required to link the ALFIE system with the ergonomics CAD system SAMMIE. From developing and using the ALFIE system it has become clear to the author that access to a geometric model of the design being undertaken would be of great benefit to both the knowledge engineer and the user of the system. It is quite clear that some form of shared database would be required to achieve this but the exact details of how much information needs to be shared by the two systems are still being investigated. In order to accommodate the vagueness permitted in the ALFIE system some mechanism for handling maximally and minimally enveloping convex hulls would have to be developed for the CAD system. This will not be easy but the author feels that advantages of visualisation in the CAD system and vagueness in the expert system during preliminary design are important enough to warrant an attempt at it.

Good (1990), under the supervision of the author, has made some excellent progress in linking a Prolog based expert system with the CAM-X InfoSOLID modeller at Heriot-Watt University. Also at Heriot-Watt, a team has recently completed a project which develops process plans by interrogating a geometric model held within the CAM-X InfoCAD system (Willis et al. 1989).

One of the major problems encountered in the research projects cited is that of turning geometrical data into predicates which can be employed in an expert system. Kapur and Mundy (1988) and Woodwark (1989) have set the scene for the advent of a new discipline of *geometric reasoning*. This new discipline is of interest not only to those concerned with integrating CAD and expert systems but also to researchers working on computer vision, geometric modelling and robotics. This variety of backgrounds should ensure that geometric reasoning becomes a very exciting field of study in the next decade.

The work of Medland (1986) on functional design is particularly relevant to future work. If truly integrated systems are to emerge then they are likely to blur some very deep-rooted dividing lines in the product design departments of those companies which use them. Much more is going to be required of them than reasoning about geometry. This is a field of study which will generate much research and development in the future.

Chapter 13

CONCLUSION

This thesis has outlined the potential benefits to be gained from using expert systems to assist in ergonomic design. It has demonstrated these benefits using a number of knowledge bases containing ergonomics expertise in conjuction with the ALFIE expert system shell. The structure of this shell has been described and its generality has been demonstrated - to the extent that it has even been possible to develop a diagnosis system with it.

A new paradigm for computer based design has been proposed and demonstrated. This paradigm adopts a model-based as well as a rule-based approach to inference. Furthermore, it has demonstrated that the tentative ideas which a designer normally brings to preliminary design do not have to be sacrificed when he sits down at a computer terminal. It is possible for a computer based design system to accept these vague ideas and reason with them whilst assisting and encouraging the user to develop a more concrete view.

The work described covers the complete process of building an expert system from specification to implementation. One particular problem stood out as being much more time consuming than anticipated during the development. This was the task of acquiring the expertise needed by the system. The thesis has attempted to highlight the problems involved and has made some small contribution to analysing the skills required of a knowledge engineer.

Finally, this thesis has indicated that the future of knowledge based design will be constrained by developments in the fields of, initially, geometric reasoning and, secondly, functional design.

REFERENCES

Allan, J.J., (ed), 1977, *CAD Systems*, North-Holland Publishing Company, Amsterdam.

Amram, F.M., 1984, 'Designing a Social Environment for Human-Robot Co-operation', in Lupton (ed), 1984.

Arbab, F., & Wang, B., 1989, 'A Constraint-Based Design System based on Operational Transformation Planning', in Gero (ed), 1989.

Banda, P., & Kuo, C., (eds), 1985, *Computer Applications in the Automation of Shipyard Operation and Ship Design V*, Elsevier Science Publishers B.V.

Barson, R.J., 1982, *Computer-Aided Mesh Generation for Finite Element Analysis*, PhD Thesis, University of Leicester.

Begg, V., 1984, *Developing Expert CAD Systems*, Kogan Page, London.

Besant, C.B., & Lui, C.W.K., 1986, *Computer-Aided Design and Manufacture*, 3rd edition, Ellis Horwood Ltd, Chichester.

Bonney, M.C., Case, K., & Porter, J.M., 1989a, 'Applications of SAMMIE and the Development of Man Modelling', in McMillan et al. (eds), 1989.

Bonney, M.C., Taylor, N.K., & Case, K., 1989b, 'Using Computer Aided Design and Expert Systems for Human Workplace Design', in Woodwark (ed), 1989.

Bower, A.J., 1988, *Expert Systems*, MEng Project Dissertation, University of Nottingham.

Bramer, M.A., (ed), 1987, *Research and Development in Expert Systems III*, Cambridge University Press, Cambridge.

Brown, D.C., & Chandrasekaran, B., 1983, 'An Approach to Expert Systems for Mechanical Design', *Proceedings of the IEEE Trends and Applications*, pp.173-180.

Brown, D.C., & Chandrasekaran, B., 1985a, 'Plan Selection in Design Problem-Solving', *Proceedings of AISB 85*, pp.108-124.

Brown, D.C., & Chandrasekaran, B., 1985b, 'Expert Systems for a Class of Mechanical Design Activity', in Gero (ed), 1985.

Buchanan, B.G., Barstow, D., Bechtal, R., Bennett, J., Clancey, W., Kulikowski, C., Mitchell, T., & Waterman, D.A., 1983, 'Constructing an Expert System', in Hayes-Roth, et al. (eds), 1983a.

Buffa, E.S., 1983, *Modern Production/Operations Management*, 7th edition, John Wiley & Sons, New York & Toronto.

Bullock, A., & Stallybrass, O., (eds), 1979, *The Fontana Dictionary of Modern Thought*, 5th impression, Fontana/Collins, London.

BYG Systems Ltd, 1986, *Investigation into the Commercial Viability of an Ergonomic Expert System*, Market Survey, BYG Systems Ltd, Nottingham.

Carney, S.P., & Brown, D.C., 1989, 'A Qualitative Model for Reasoning about Shape and Fit', in Gero (ed), 1989.

Case K., & Porter, J.M., 1980, 'SAMMIE: A Computer-Aided Ergonomics Design System', *Engineering*, January, pp.21-25.

Chan, W.T., & Paulson, B.C., 'Exploratory Design Using Constraints', *(AI EDAM)*, **1** (1), pp.59-71.

Childs, D., 1986, *Britain Since 1945*, 2nd edition, Methuen & Company Ltd, London.

Christensen, J.M., 1976, 'Ergonomics : Where have we been and where are we going?', *Proceedings of the 6th Congress of the International Ergonomics Association*, pp.25-33.

Clark, T.S., & Corlett, E.N., 1984, *The Ergonomics of Workspaces and Machines : A Design Manual*, Taylor & Francis, London.

Connolly, A., 1988, 'Lighting for Visual Performance, Comfort and Fatigue: A Knowledge Base for the ALFIE Expert System', Technical Report ES.14, Department of Production Engineering and Production Management, University of Nottingham.

Connolly, A., 1990, Private Communication, Swansea.

Conservative Party Manifesto, 1987, *The Next Moves Forward*, Conservative Central Office, Smith Square, London.

Corlett, E.N., 1985, Private Communication, Nottingham.

Corlett, E.N., 1988, 'Where have we come from and where are we going?', in Megaw (ed), 1988.

Coyne, R.D., 1988, *Logic Models of Design*, Pitman, London.

Coyne, R.D., Rosenman, M.A., Radford, A.D., Balachandran, M., & Gero, J.S., 1990, *Knowledge-Based Design Systems*, Addison-Wesley Publishing Company, Reading MA.

Davis, R., 1977, 'Interactive Transfer of Expertise I : Acquisition of New Inference Rules', *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pp.321-328.

Department of Trade and Industry, 1985, *How to Improve Television and Radio Reception*, DoTI booklet.

Department of Trade and Industry, 1990, *Expert System Opportunities*, DoTI brochure.

Dewar, J.K.S., 1968, *Implementation of an Interval Arithmetic on an ICL (ELLIOTT) 4130 System*, MSc Thesis, University of Dundee.

Dixon, J.R. & Simmons, M.K., 1983, 'Computers that Design: Expert Systems for Mechanical Engineers', *Computers in Mechanical Engineering*, November, pp.10-18.

Dooner, M., Taylor, N.K., & Bonney, M.C., 1982, 'Planning Robot Installations by CAD', *Proceedings of the 5th International Conference on Computers in Design Engineering - CAD82.*

Doyle, J., 1979, 'A Truth Maintenance System', *Artificial Intelligence*, **12**, pp.231-272.

Duda, R.O., Gashnig, J.E., & Hart, P.E., 1979, 'Model Design in the PROSPECTOR Consultant System for Mineral Exploration', in Michie (ed), 1979.

Edwards, P.R., 1987, *The Computer Aided Design and Manufacture of Cutting Patterns on Crystal Glassware*, PhD Thesis, University of Nottingham.

Eklund, J.A.E., 1988, 'Organisation of Assembly Work : Recent Swedish Examples', in Megaw (ed), 1988.

Erman, L.D., Hayes-Roth, F., Lesser, V., & Reddy, D., 1980, 'The HEARSAY-II Speech-Understanding System : Integrating Knowledge to Resolve Uncertainty', *Computing Surveys*, **12** (2), pp.213-253.

Evans, S.M., 1985, 'Ergonomics in Workspace Design : Current Practices and an Alternative Computer-Aided Approach', *Human Factors Society 29th Meeting*, Baltimore MD.

Fanger, P.O., 1970, *Thermal Comfort, Analysis and Applications in Environmental Engineering*, Danish Technical Press, Copenhagn.

Feigenbaum, E.A., 1979, 'Themes and Case Studies of Knowledge Engineering', in Michie (ed), 1979.

Feigenbaum, E.A., Buchanan, B.G., & Lederberg, J., 1971, 'On Generality and Problem Solving : A Case Study Using the DENDRAL Program', in Meltzer & Michie (eds), 1971.

Feigenbaum, E.A., & Feldman, J., (eds), 1963, *Computers and Thought*, McGraw-Hill, New York.

Foley, J.D., & Van Dam, A., 1984, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley Publishing Company, Reading MA.

Forsyth, R., (ed), 1984a, *Expert Systems - Principles and Case Studies*, Chapman & Hall, London.

Forsyth, R., 1984b, 'The Architecture of Expert Systems', in Forsyth (ed), 1984a.

Fox, J., Myers, C.D., Greaves, M.F., & Pegram, S., 1985, 'Knowledge Acquisition for Expert Systems : Experience in Leukaemia Diagnosis', *Methods of Information in Medicine*, **24**, pp.65-72.

Gero, J.S., (ed), 1985, *Knowledge Engineering and Computer-Aided Design*, North-Holland, Amsterdam.

Gero, J.S., (ed), 1989, *Artificial Intelligence in Design*, Computational Mechanics Publications with Springer-Verlag.

Gillan, W.J., 1987, 'The Japanese Secret - Are They Winning?' Report issued by the Overseas Technical Information Unit, Department of Trade and Industry.

Good, S.N., 1990, *The Design and Implementation of a Solid Model Query Language*, BSc Project Dissertation, Heriot-Watt University, Edinburgh.

Grandjean, E., 1980, *Fitting the Task to the Man*, 3rd edition, Taylor & Francis, London.

Green, S., 1987, 'SPACES - A System for the Representation of Commonsense Knowledge about Space for Design', in Bramer (ed), 1987.

Groover, M.P., & Zimmers, E.W., 1984, *CAD/CAM : Computer-Aided Design and Manufacturing*, Prentice-Hall, Englewood Cliffs NJ.

Grover, M.D., 1983, 'A Pragmatic Knowledge Acquisition Methodology', *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pp.426-438.

Guilfoyle, C., 1986a, 'At a Glance, a Choice of Euro Micro Shells', *Expert Systems User*, March, pp.14-15.

Guilfoyle, C., 1986b, 'A Table Load Full of Micro Shells', *Expert Systems User*, April , pp.18-20.

Gummer, J.S., 1984, Opening Address at the 1st International Conference on Human Factors in Manufacturing, in Lupton (ed), 1984.

Harbury, C., & Lipsey, R.G., 1983, *An Introduction to the UK Economy*, Pitman, London.

Harris, D.G., Freathy, P., Dawson, J.A., & Davies, B.K., 1990, *A Report on a Computer Board Initiative on Knowledge Based Systems,* Institute for Retail Studies, University of Stirling.

Hayes, P.J., 1979, 'The Naive Physics Manifesto', in Michie (ed), 1979.

Hayes-Roth, F., Waterman, D.A., & Lenat, D.B., (eds), 1983a, *Building Expert Systems,* Addison-Wesley Publishing Company, Reading MA.

Hayes-Roth, F., Waterman, D.A., & Lenat, D.B., 1983b, 'An Overview of Expert Systems', in Hayes-Roth, et al. (eds), 1983a.

Herzberg, F., Mausner, B., & Synderman, B., 1959, *The Motivation to Work,* John Wiley & Sons.

Hume, D., 1739, *A Treatise of Human Nature,* 2nd edition. Edited by L.A.Selby-Bigge (1888), revised by P.H.Nidditch (1978), Clarendon Press, Oxford.

Jackson, P.C., 1974, *Introduction to Artificial Intelligence,* Petrocelli/Charter, New York.

John, P.A., 1988, 'The Ergonomics of Computer Aided Design within Advanced Manufacturing Technology', *Applied Ergonomics,* **19** (1), pp.40-48.

Johnson, L., & Keravnou, E.T., 1985, *Expert Systems Technology,* Abacus Press, Tunbridge Wells.

Johnston, R., 1986, 'Breaking the Rules', *Expert Systems User,* May, pp.23-26.

Johnstone, P., 1990, *Expert Systems and their User Interfaces*, BSc Project Dissertation, University of Nottingham.

Kapur, D., & Mundy, J.L., 1988, 'Geometric Reasoning and Artificial Intelligence: Introduction to the Special Volume', *Artificial Intelligence*, **37** (1), pp.1-11.

Karwowski, W., & Ayoub, M.M., 1984, 'Fuzzy Modelling of Stresses in Manual Lifting Tasks', *Ergonomics*, **27** (6), pp.641-649.

Kidd, A.L., & Sharpe, W.P., 1988, 'Goals for Expert Systems Research: An Analysis of Tasks and Domains', in Moralee (ed), 1988.

de Kleer, J., 1986, 'An Assumption-based TMS', *Artificial Intelligence*, **28**, pp.127-162.

Labour Party Manifesto, 1987, *Britain Will Win*, The Labour Party, Walworth Road, London.

Larsson, K-A., 1984, 'Car Assembly at the Volvo Kalmar Plant', in Lupton (ed), 1984.

Latombe, J-C., 1977, 'Artificial Intelligence in Computer-Aided Design : The TROPIC System', in Allan (ed), 1977.

Latombe, J-C., (ed), 1978, *Artificial Intelligence and Pattern Recognition in Computer Aided Design*, North-Holland Publishing Company, Amsterdam.

Latombe, J-C., 1979, 'Failure Processing in a System for Designing Complex Assemblies', *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, pp.508-515.

Lawlor, M., 1984, 'Participation - Involvement : The Guinness (Dublin) Experience', in Lupton (ed), 1984.

Lupton, T., (ed), 1984, *HUMAN 1*, IFS (Publications) Ltd, Bedford.

MacCallum, K.J., 1982, 'Understanding Relationships in Marine Systems Design', *International Marine Systems Design Conference*, RINA, London.

MacCallum, K.J., & Duffy, A.H.B., 1985, 'Approximate Calculations in Preliminary Design', in Banda & Kuo (eds), 1985.

MacCallum, K.J., & Duffy, A.H.B., 1987, 'An Expert System for Preliminary Numerical Design Modelling', *Design Studies*, **8** (4).

MacCallum, K.J., Duffy, A.H.B., & Green, S., 1985, 'An Intelligent Concept Design Assistant', Technical Note ESG/23/P, University of Strathclyde.

McCarthy, W., 1988, *The Future of Industrial Democracy*, Fabian Society Tract 526, The Fabian Society, London.

McCormick, E.J., & Sanders, M.S., 1983, *Human Factors in Engineering and Design*, 5th edition, McGraw-Hill, Tokyo.

McDermott, J., 1982, 'R1 : A Rule-Based Configurer of Computer Systems', *Artificial Intelligence*, **19** (1), pp.39-88.

McMillan, G.R., et al., (eds), 1989, *Applications of Human Performance Models to Systems Design*, Plenum Press, London.

Martin, W.A., & Fateman, R.J., 1971, 'The MACSYMA System', *Proceedings of the 2nd Symposium on Symbolic and Algebraic Manipulation*, pp.59-75.

Medland, A.J., 1986, *The Computer Based Design Process*, Kogan Page, London.

Megaw, E.D., (ed), 1987, *Contemporary Ergonomics 1987*, Taylor & Francis, London.

Megaw, E.D., (ed), 1988, *Contemporary Ergonomics 1988*, Taylor & Francis, London.

Meltzer, B., & Michie, D., (eds), 1971, *Machine Intelligence 6*, American Elsevier, New York.

Michie, D., (ed), 1979, *Expert Systems in the Micro-Electronic Age*, Edinburgh University Press, Edinburgh.

Michie, D., 1982, 'Experiments on the Mechanisation of Game-Learning', *Computer Journal*, **25** (1), pp.105-113.

Moore, R.E., 1966, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, NJ.

Moralee, D.S., (ed), 1988, *Research and Development in Expert Systems IV*, Cambridge University Press, Cambridge.

Moses, J., 1967, *Symbolic Integration*, PhD Thesis, Massachusetts Institute of Technology.

Newell, A., & Simon, H.A., 1963, 'GPS, A Program that Simulates Human Thought', in Feigenbaum & Feldman (eds), 1963.

Nickel, K., (ed), 1986, *Interval Mathematics 1985*, Springer-Verlag, Berlin.

Oakley, K.P., 1975, *Man the Tool-Maker*, 6th edition, British Museum, London.

Pemberton, I., 1985, *The Utilisation of Ergonomic Data in Industrial Design*, MPhil Dissertation, Loughborough University of Technology.

Pople, H.E., 1984, contribution to 'Expert Systems: A Discussion', in Winston & Prendergast (eds), 1984.

Popper, K.R., 1933, 'Ein Kriterium des empirischen Charakters theoretischer Systeme', Erkenntnis, **3**, pp.426f.

Popper, K.R., 1977, *The Logic of Scientific Discovery*, 9th impression, Hutchinson & Co Ltd, London.

Popper, K.R., 1983, *Objective Knowledge*, 2nd edition, Clarendon Press, Oxford.

Popper, K.R., 1989, *Conjectures and Refutations*, 5th edition, Routledge, London & New York.

Popplestone, R.J., 1987, 'The Edinburgh Designer System as a Framework for Robotics: The Design of Behaviour', *(AI EDAM)*, **1** (1), pp.25-36.

Porter, J.M., Case, K., & Bonney, M.C., 1990, 'Computer Workspace Modelling', in Wilson & Corlett (eds), 1990.

Raphael, B., 1976, *The Thinking Computer*, W H Freeman & Company, San Francisco CA.

Reitman, W., & Weischedel, R., 1985, 'Automated Information Management Technology : Considerations for a Technology Investment Strategy', Air Force Aerospace Medical Research Laboratory Report TR-85-042, pp.27-48.

Rumelhart, D.E., McClelland, J.L., & The PDP Research Group, 1986, *Parallel Distributed Processing*, Vols 1 & 2, MIT Press, Cambridge MA.

Sabin, M., (ed), 1984, *Report on the Workshop on Expert Systems in Manufacturing Engineering*, ACME Directorate, SERC, Swindon.

Sapossnek, M., 1989, 'Research on Constraint-Based Design Systems', in Gero (ed), 1989.

Schweickert, R., Burton, A.M., Taylor, N.K., Corlett, E.N., Shadbolt, N.R., & Hedgecock, A.P., 1987, 'Comparing Knowledge Elicitation Techniques : A Case Study', *Artificial Intelligence Review*, **1** (4), pp.245-253.

SDP/Liberal Alliance Manifesto, 1987, *Britain United - The Time Has Come*, SDP, Cowley Street, London.

Shafer, G., & Tversky, A., 1985, 'Languages and Designs for Probability Judgement', *Cognitive Science*, **9**, pp.309-339.

Shah, J.J., & Rogers, M.T., 1988a, 'Functional Requirements and Conceptual Design of the Feature Based Modelling System', *Computer Aided Engineering Journal*, February, pp.9-15.

Shah, J.J., & Rogers, M.T., 1988b, 'Expert Form Feature Modelling Shell', *Computer Aided Design*, **20** (9), pp.515-522.

Shortliffe, E., 1976, *Computer-based Medical Consultations : MYCIN*, Elsevier, New York.

Simpson, M.R., 1987a, 'Report on the Construction of an Intelligent Knowledge Based System to Assess Heat Stress', Technical Report ES.11, Department of Production Engineering and Production Management, University of Nottingham.

Simpson, M.R., 1987b, 'Overview of the ALFIE Heat Stress Knowledge Base', Technical Report ES.12, Department of Production Engineering and Production Management, University of Nottingham.

Simpson, M.R., 1987c, 'Estimation of Energy Expenditure and its Application in ALFIE', Technical Report ES.13, Department of Production Engineering and Production Management, University of Nottingham.

Simpson, M.R., 1988a, 'Documentation for the ALFIE Thermal Comfort Knowledge Base', Technical Report ES.15, Department of Production Engineering and Production Management, University of Nottingham.

Simpson, M.R., 1988b, 'The possibilities of Expansion of the current Thermal Comfort Knowledge Base', Technical Report ES.16, Department of Production Engineering and Production Management, University of Nottingham.

Slagle, J.R., 1963, 'A Heuristic Program that Solves Symbolic Integration Problems in Freshman Calculus', in Feigenbaum & Feldman (eds), 1963.

Spur, G., & Krause, F-L., 1985, 'Research Problems of CAD Systems', *Annals of the CIRP*, **34** (1), pp.183-186.

Stefik, M., Aikins, J., Balzer, R., Benoit, J., Birnbaum, L., Hayes-Roth, F., & Sacerdoti, E., 1982, 'The Organisation of Expert Systems - A Tutorial', *Artificial Intelligence*, **18** (2), pp.135-173.

Sutherland, I.E., 1963, *SKETCHPAD : A Man-Machine Graphical Communication System*, Spartan Books, Baltimore MD.

Taylor, N.K., 1979, *A Computer Model of the Hippocampus*, MSc Project Dissertation, University of London.

Taylor, N.K., Bonney, M.C., & Yong, Y.F., 1982, 'CAD for Planning Safe Robot Systems', *Proceedings of the 1st Robot Safety Seminar*, University of Nottingham.

Taylor, N.K., & Corlett, E.N., 1984, 'Report on the Development of an Expert Aid for Ergonomic Design', in Sabin (ed), 1984.

Taylor, N.K., & Corlett, E.N., 1987a, 'ALFIE - Auxiliary Logistics For Industrial Engineers', *Industrial Ergonomics*, **2** (1), pp.15-25.

Taylor, N.K., & Corlett, E.N., 1987b, 'An Expert System which Constrains Designs', *Artificial Intelligence in Engineering*, **2** (2), pp.72-75.

Taylor, N.K., & Corlett, E.N., 1987c, 'The Development of an Expert Aid for Ergonomic Design', *Proceedings of the ACME 1987 Research Conference*, ACME Directorate, SERC, Swindon.

Taylor, N.K., Corlett, E.N., Boggs, G.J., & Freivalds, A., 1985, 'Aiding Ergonomic Design Using an Expert System', *Proceedings of the 9th Congress of the International Ergonomics Association.*

Taylor, N.K., Corlett, E.N., & Simpson, M.R., 1987, 'Problems of Knowledge Acquisition for Expert Systems', in Megaw (ed), 1987.

Tricker, R.I., 1979, 'Scientific Management', in Bullock & Stallybrass (eds), 1979.

Tynan, O., 1984, 'People are Assets!', in Lupton (ed), 1984.

Warman, E.A., 1978, 'Computer Aided Design : An Intersection of Ideas', in Latombe (ed), 1978.

Welbank, M., 1983, 'A Review of Knowledge Acquisition Techniques for Expert Systems', British Telecom Research Laboratories Report.

Westerberg, A., Grossman, I., Talukdar, S., Prinz, F., Fenves, S., & Maher, M.L., 1989, 'Applications of AI in Design Research at Carnegie Mellon University's EDRC', in Gero (ed), 1989.

Willis, D., Donaldson, I.A., Ramage, A.D., Murray, J.L., & Williams, M.H., 1989, 'A Knowledge-Based System for Process Planning based on a Solid Modeller', *Computer-Aided Engineering Journal*, February, pp.21-26.

Wilson, J.R., & Corlett, E.N., (eds), 1990, *Evaluation of Human Work*, Taylor & Francis, London.

Winston, P.H., 1984, *Artificial Intelligence*, 2nd edition, Addison-Wesley Publishing Company, Reading MA.

Winston, P.H., & Prendergast, K.A., (eds), 1984, *The AI Business: The Commercial Uses of Artificial Intelligence*, MIT Press, Cambridge MA.

Woodwark, J.R., (ed), 1989, *Geometric Reasoning*, Clarendon Press, Oxford.

Appendix A

A MATHEMATICAL MODEL OF DESIGN

Every design consists of a compromise between a number of factors or design variables. The objective of the design process is to optimise the values assigned to these variables.

We can consider the design as a function, $D(V)$, of a set of n design variables, $V = \{ V_1, ..., V_n \}$.

A 'best' design (and there might be more than one in a given application) might then be defined to be one which maximises this function in some manner.

This can be written as,

$$\{ V : D(V) \geq D(A), \forall A \} \qquad (1)$$

That is, those sets of values of the design variables which maximise $D(V)$ will constitute 'best' designs.

Clearly, each variable is going to be constrained to lie within a certain range - in the final analysis there will always be practical and financial constraints if nothing else so we can form sets of constraints, C_i, such that,

$$V_i \in C_i, \forall i \quad where \; C_i = \{ C_{i_1}, ..., C_{i_m} \}$$

Given limits on the values of each V_i we can form a set of normalising functions, $N_i(V_i)$, which map the V_i onto the interval [-1,1]. In fact we can further insist that each function, $N_i(V_i)$, returns unity when, and only when, we have the most favourable value of the variable concerned. Similarly we can ensure that each $N_i(V_i)$ returns minus one only when the variable, V_i, takes the worst value we can imagine. Between these two extremes we can have a gradation of the values of $N_i(V_i)$ which is governed by the favourability of each value of V_i.

Now we can tentatively formulate an initial model, or function $D(V)$,

$$D(V) = \sum_{i=1}^{n} N_i(V_i) \tag{2}$$

Our model will return a value for $D(V)$ in the range $[-n, n]$. The larger the value the 'better' the design is considered to be.

The next aspect we need to consider is that some variables will be considered to be of greater importance than others. That is to say, some variables will be expected to have a much more significant effect on the overall merits of the design than others. To take account of this we need to introduce the concept of weighting each term in the summation according to its importance.

If we define W_i to be the contribution made by each V_i to the design we can re-formulate our initial model (2) as,

$$D(V) = \sum_{i=1}^{n} W_i N_i(V_i) \tag{3}$$

If the W_i are constrained by the inequality,

$$0 \le W_i \le W_{\max}, \forall\, i$$

then our model, $D(V)$, will be constrained by the inequality,

$$-n.W_{\max} \le D(V) \le n.W_{\max}$$

So far we have considered each design variable in isolation. In practice we will often find that we need to consider the interaction between variables. In terms of our model (3) we might find that a value for V_i of k is considered good when some other V_j takes the value l but poor when it takes the value m. In fact, we may find that some of the design variables have no intrinsically good or poor

values at all but are wholly dependent on the values of other variables.

What we need now is an extension to our functions, $N_i(V_i)$, which takes account of the dependence of each V_i on the values of other variables.

Consider the set of functions,

$$\{ N_i(V_i \mid V_j, \forall j \neq i) \}$$

This set of functions, N_i, will return a value based upon a certain value of V_i given prior values for all the other variables, V_j.

In practice some of the N_i may return the same value for a given V_i irrespective of the values of the other V_j (ie. the V_i is independent of the other V_j) but by formulating our N_i in this way we are ensuring that we have a completely general model.

Our model (3) now becomes :

$$D(V) = \sum_{i=1}^{n} W_i N_i(V_i \mid V_j, \forall j \neq i) \qquad (4)$$

This model will serve us very well for deciding between two or more given designs (values for the V_i) but it will not easily allow us to produce a 'best' design. In order to optimise our design - maximise $D(V)$ - we will need to evaluate the function, $D(V)$, for all possible combinations of the design variables, V_i.

When all the sets of constraints, C_i, are finite this may be a viable approach but when some are infinite or exceedingly large then it clearly is not. In these circumstances an iterative approach will be required. This method would involve

optimising each V_i with respect to the other V_j and then using that optimum to optimise another V_j and so on. The iteration is unlikely to converge to a maximum for $D(V)$ and would need to be implemented by iterating a fixed number of times and recording the maximum $D(V)$ obtained over the whole iteration.

We have developed a mathematical model for the design process. This is all well and good but the mathematics hides a very big problem - namely, determining the set of functions, N_i, and the weights, W_i. These functions and weights will be entirely application dependent. They cannot, by their nature, be general and at this point we have to consider specific design areas. Moreover, it is very unlikely, even in a very restricted set of design problems, that we will be able to determine the N_i with any degree of accuracy or confidence without resorting to empirical techniques. We are now moving from the global, deterministic mathematical model to the less determinate and more heuristic details of the design process.

There are a number of questions that spring to mind when we consider the mathematical model :

a). Precisely what are the V_i in the application under consideration ?

b). What are the W_i in the application under consideration ?

c). What are the N_i in the application under consideration ?

d). Given $N_i(V_i \mid V_j)$ and $N_i(V_i \mid V_k)$ can we approximate to $N_i(V_i \mid V_j \& V_k)$?

e). What do we do if we don't know an $N_i(V_i \mid V_j)$?

We should be able to form an initial set of V_i and W_i by questioning practitioners in the design area under consideration. However, it is important to remember that these will be working assumptions and subject to continual improvement. For each general class of design we will need to know which factors are considered relevant and how much importance each is to be given.

The $\{N_i\}$ will be a collection of relations producing, from a set of values for the design variables, a measure of the worth of that set. Since we cannot hope to determine all the N_i empirically we shall now look at how we might generate these functions automatically.

The N_i are functions which will have to be formulated from a consideration of all of the variables in the design. It would be of great value to us to generate a set of relations governing the formulation of these functions if possible.

Given $N_i(V_i \mid V_j)$ and $N_i(V_i \mid V_k)$ we could simply average these two functions to produce the following relation,

$$N_i(V_i \mid V_j \& V_k) = \frac{N_i(V_i \mid V_j) + N_i(V_i \mid V_k)}{2} \tag{5}$$

However, since we have a measure of the importance of each of the variables, V_j and V_k, we can form a better relation by using the weighted average,

$$N_i(V_i \mid V_j \& V_k) = \frac{W_j N_i(V_i \mid V_j) + W_k N_i(V_i \mid V_k)}{W_j + W_k} \tag{6}$$

We can generalise this to form the relation,

$$N_i(V_i \mid V_j, \forall\, j \neq i) = \frac{\sum_{j \neq i} W_j N_i(V_i \mid V_j)}{\sum_{j \neq i} W_j} \qquad (7)$$

So far the relations we have derived have only been concerned with the contribution of the V_i under consideration given one other V_j at a time. From this we have generalised to formulate functions for the contribution of the V_i given values for all the other V_j. However, it might not be necessary for us to derive all these functions theoretically. If we find that empirical rules exist for certain combinations of the V_i and V_j we should use these in preference to our (possibly idealised) derivations.

For instance, if we know the relationship between, say, three factors, V_i, V_j and V_k, we should use this instead of deriving the three equivalent functions: $N_i(V_i \mid V_j \& V_k)$; $N_j(V_j \mid V_i \& V_k)$ and $N_k(V_k \mid V_i \& V_j)$ from the equations given in (6).

When we come to the equations (7) we should treat the subset $\{V_i, V_j, V_k\}$ as distinct from the rest of the set $\{V\}$,

$$N_i(V_i \mid V_m, \forall\, m \neq i) = \frac{(W_j + W_k) N_i(V_i \mid V_j \& V_k) + \sum_{l \neq i,j,k} W_l N_i(V_i \mid V_l)}{\sum_{m \neq i} W_m} \qquad (8)$$

We can apply this method to all the disjoint subsets of $\{V\}$ that we have empirical knowledge of by removing them from the summation in the numerator of (8) and treating them separately, as we have with $\{V_i, V_j, V_k\}$. However, we still need to consider the case when some of the subsets are not disjoint, ie. when two or more subsets are concerned with the same variable. If we do not treat non-disjoint subsets in a special manner we will end up with a disproportionate

reliance on the common variables in our final equation. We can solve this problem by insisting that all subsets with factors in common be merged into a single subset in which the weight of the common factors is averaged out,

$$N_i(V_i \mid V_j \& V_k \& V_l) = \frac{\frac{W_j+W_k}{2} N_i(V_i \mid V_j \& V_k) + \frac{W_l+W_k}{2} N_i(V_i \mid V_l \& V_k)}{W_j+W_k+W_l}$$

If we let D_i represent the contribution of the i-th disjoint subset, S_i, and consider each variable that does not belong to a subset as a single element subset we can re-formulate our equation (8) in terms of the contributions of subsets,

$$N_i(V_i \mid V_j, \forall\, j \neq i) = \frac{\sum_k D_k}{\sum_{l \neq i} W_l} \tag{9}$$

Where each D_k would be of the form,

$$D_k = (W_x + \cdots + W_z) N_i(V_i \mid V_x \& \cdots \& V_z)$$

and each S_k would be a disjoint subset, $S_k = \{ V_x, ..., V_z \}$, so that,

$$\bigcup_k S_k = \{ V \} \setminus V_i \quad \text{and} \quad S_i \bigcap S_j = \varnothing, \forall\, i \neq j$$

Until now, we have assumed that the V_j are mutually independent unless we have an empirical relation between them. In the absence of any evidence to the contrary we can assume independence of some V_j but when we know that a subset of $\{ V \}$ is mutually dependent we must take advantage of this knowledge and incorporate it in our model, (9). We can do this by ensuring that the subset be represented as an S_k.

Our final question was concerned with incomplete data. If we find a situation in which an unspecified N_i is required in the evaluation we have two options:

a). We could request information from the user sufficient to form an approximation to the N_i;

b). We could merely assign a default value to the N_i.

Both these approaches have their advantages. Option (a) will probably provide a more realistic input whereas option (b) would relieve the user of performing what might turn out to be an arduous task. The relative merits of each would best be determined by trial and error.

Appendix B

AN ALGEBRA OF INTERVALS

Definition 1

Let $[A]$ represent a variable whose value is constrained to lie within the closed interval $[A_{min}, A_{max}]$. We shall call $[A]$ an **interval variable**.

The fundamental rules of arithmetic for independently constrained, or *independent*, interval variables can be formulated as follows,

Axiom 1

$$[A] + [B] = [A + B]$$

That is, $(A + B)_{min} = A_{min} + B_{min}$

$$(A + B)_{max} = A_{max} + B_{max}$$

Axiom 2

$$[A] - [B] = [C]$$

where $C_{min} = A_{min} - B_{max}$

$$C_{max} = A_{max} - B_{min}$$

Axiom 3

$$[A] \times [B] = [C]$$

where $C_{min} = Min\ \{A_{min} \times B_{min}, A_{max} \times B_{max}, A_{min} \times B_{max}, A_{max} \times B_{min}\}$

$$C_{max} = Max\ \{A_{min} \times B_{min}, A_{max} \times B_{max}, A_{min} \times B_{max}, A_{max} \times B_{min}\}$$

Axiom 4

$$[A]/[B] = [C] \quad when \ 0 \notin [B_{min}, B_{max}]$$

$$where \quad C_{min} = Min\ \{A_{min}/B_{min}, A_{max}/B_{max}, A_{min}/B_{max}, A_{max}/B_{min}\}$$

$$C_{max} = Max\ \{A_{min}/B_{min}, A_{max}/B_{max}, A_{min}/B_{max}, A_{max}/B_{min}\}$$

Note that the algebraic structure defined above does not constitute a *group* under any of the operations defined in the axioms due to the lack of additive and multiplicative inverses. Whilst the purist might therefore be tempted to question its usefulness, the ALFIE system provides a clear counter example to such a suggestion.

Lemma 1

We can represent any single-valued or *point* variable, x, as an interval variable, $[X]$, with $X_{min} = X_{max} = x$. Proof trivial.

Lemma 1 and axioms 1 to 4 yield the following, trivially provable, theorems,

Theorem 1

$$x + [A] = [A] + x = [A + x]$$

$$where \quad (A+x)_{min} = A_{min} + x$$

$$(A+x)_{max} = A_{max} + x$$

Theorem 2a

$$[A] - x = [A - x]$$

Theorem 2b

$$x - [A] = [C]$$

where $C_{\min} = x - A_{\max}$

$C_{\max} = x - A_{\min}$

Theorem 3

$$x \times [A] = [A] \times x = [A \times x] \quad \textit{when } x > 0$$

$$= [C] \quad \textit{when } x < 0$$

where $C_{\min} = A_{\max} \times x$

$C_{\max} = A_{\min} \times x$

Theorem 4a

$$x / [A] = [x / A] \quad \textit{when } x < 0$$

$$= [C] \quad \textit{when } x > 0$$

where $C_{\min} = x / A_{\max}$

$C_{\max} = x / A_{\min}$

Undefined when $0 \in [A_{\min}, A_{\max}]$

Theorem 4b

$$[A]/x = [A/x] \quad when\ x > 0$$

$$= [C] \quad when\ x < 0$$

$$where\ C_{min} = A_{max}/x$$

$$C_{max} = A_{min}/x$$

Undefined when $x = 0$

Non-Independent Interval Variables

If we take two interval variables which are not independent and combine them using axioms 2 to 4 we will not produce a maximally constrained result. We can employ knowledge of the extra constraints (dependencies) to further constrain the resultant variable. Consider the case where $[A]$ and $[B]$ are the same variable and recall that, although $[A]$ is an interval, it represents a degree of vagueness concerning a measure which has a *single* value. We can, therefore, identify three special cases,

$$[A]-[A] = [0] = 0$$

$$[A]/[A] = [1] = 1$$

$$[A]\times[A] = [C]$$

$$where\ \ C_{min} = 0 \ \ when\ A_{min} \times A_{max} < 0$$

else C_{min}, C_{max} *defined as in axiom* 3

The first two cases are obvious. The third however, is an example of a higher order function in which turning points have to be considered. In general, a little thought applied to functions with turning points (sine, cosine, tangent, etc.) will

permit the derivation of a correctly and maximally constrained result.

Relational Operators

The standard inequality relations are defined as follows,

$[A] = [B]$ *iff* $A_{\min} = B_{\min} \,\&\, A_{\max} = B_{\max}$

$[A] \neq [B]$ *iff* $A_{\min} \neq B_{\min}$ *or* $A_{\max} \neq B_{\max}$

$[A] < [B]$ *iff* $A_{\max} < B_{\min}$

$[A] > [B]$ *iff* $A_{\min} > B_{\max}$

$[A] \leq [B]$ *iff* $A_{\min} \leq B_{\min} \,\&\, A_{\max} \leq B_{\max}$

$[A] \geq [B]$ *iff* $A_{\min} \geq B_{\min} \,\&\, A_{\max} \geq B_{\max}$

The definitions of ≤ and ≥ used here differ from those commonly employed in interval analysis. The above definitions are in keeping with the convention that two intervals which are equal will also be judged to be ≤ and ≥. More importantly however, these definitions permit overlapping intervals to be detected no matter what the size of the overlap whilst the usual definitions can only detect 'touching' intervals.

Appendix C

ALFIE MANUALS

This appendix contains the complete set of manuals for ALFIE version 5.4. There are six manuals altogether:

User Guide - A brief description of the system and how it works aimed at users of the system who will not want much detail;

Knowledge Engineering Guide - A fuller coverage of the system aimed at those who intend to write knowledge bases for it;

User Command Reference Guide - A complete catalogue of all the commands and options available to a user of the system;

Knowledge Engineering Language Reference Guide - A complete exposition of the knowledge engineering language provided with the system;

Default Knowledge Base Reference Guide - A description of each of the *concepts* in the default knowledge base provided with the system;

System Administrator's Reference Guide - A description of the implementation and management routines required by the system.

Interaction with ALFIE proceeds through the examination of *Concepts*. These are modules of domain knowledge. They provide a means for reducing the problem area to manageable sub-problems as well as a means of identifying what has been done and what remains to be done.

The concepts are linked together in such a manner that the system can guide the user to related areas of the problem domain if he runs out of ideas. One concept is nominated as the 'root' concept and this is where the user is guided if he gets stuck and has not given the system any idea of what he is interested in.

Once the user or the guidance system has commenced examination of a concept a number of decisions are made. These decisions are recorded in *Rules*. In order for the system to make its decision it will normally have to request a certain amount of information from the user.

This information is recorded in *Factors*. The conditions of the rules are based on logical combinations of these factors and expressions formed from them. Apart from asking the user, the system has another manner in which to get values for factors.

The system can represent *Models* which relate factors to one another and it is therefore capable of evaluating certain factors from other factors. The system's models are not regarded as eternal truths but are selected according to the prevailing situation. Under differing circumstances different sets of models will be in use or *enabled* by the system. Those not in use at any particular time are said to be *disabled.*

If more than one model can be used to evaluate a factor at any time the system will quite happily use them all. It can do this because most models are used to produce numerical results and the system treats numerical factors as ranges, not fixed values. The value assigned to a factor when more than one model can be used is simply the range of results common to all of the models.

Having got the information on which to base its decision, the system will proceed to examine further concepts, fire further rules, enable and disable models and report its advice back to the user. The user is then free to refute any of the information which he gave to the system and instruct the system to accommodate any new information he may care to give.

An explanation facility is provided through which the user is able to question the system on its actions and advice at any time during a session.

At the end of a session the user can request a hard-copy of the constraints on the important factors which have been considered along with the advice which the system gave as a result of those constraints.

System Architecture

The overall problem domain is represented as a number of *Concepts* linked together in a network. These concepts permit the problem area to be broken down into interacting sub-problems in a modular manner.

Concepts provide the user with a means of moving around the system both providing and eliciting information as he goes. The links between concepts are of three types:

Superconcept links link a concept to more general concepts;

Subconcept links link a concept to more detailed concepts;

Isoconcept links link a concept to other related concepts.

On electing to examine a certain concept the user will activate the appropriate node in the system network. Attached to each concept in the network are a number of *Rules*. These take the form of <condition,action> pairs and an optional default action. When a concept is activated the system examines the condition parts of each rule and performs the associated action for every true condition. A default action may be specified to cater for the case when all the conditions of a rule are false.

There are two other types of object within the system which need to be discussed before we can take a more detailed look at rules. The first is a *Factor*. Factors represent logical predicates, which are either true or false; numerical ranges for quantitative information, which are constrained to lie within a certain range of values; and sets of objects, which provide for a multiple-choice on the part of the

user.

The final type of object in the system is a *Model*. Models are equations which state the relationships that should hold between factors. An example of a numerical model would be:

$$V \leftarrow U + A * T \tag{1}$$

When this model is invoked V will be amended to ensure that its maximum permissible value does not exceed the maximum value of the right hand side and that its minimum permissible value is not less than the minimum value of the right hand side. If this is not possible the user is warned of the conflicting interests. This process of constraint refinement until either a contradiction or a consistent set of factor values is reached is a natural manner in which to represent the design process since it provides for the cumulative acquisition of constraints on the design parameters over a period of time.

As well as being linked to their constituent factors, models are also used to build an influence net between the factors. Thus, in (1), U, A and T are linked to V since any change in the former could change the latter. Were such a change to occur, V and any factors which it influences would be flagged so that all future references to them on the right hand side of a model would automatically lead to their re-evaluation.

Under differing circumstances it may be necessary to use different models to constrain factors. For instance, when calculating velocities we can use a number of different models depending on the information we have available to us. The choice is made by accepting and rejecting models on the basis of factor

constraints.

We are now in a position to look at the general form of a rule:

WHEN <condition 1> : <action 1>

.

.

WHEN <condition n> : <action n>

DEFAULT <action n+1>

Examples of conditions are :

(V > 50) AND (V < 100)

UNIFORM_MOTION AND NOT WIND_RESISTANCE

The action parts of a rule instruct the system to perform such operations as examining concepts, complying with rules, enabling or disabling models, and displaying advice.

System Knowledge

The domain knowledge is imparted to the system by means of a declarative knowledge engineering language which describes the concepts, rules, models and factors which are relevant to the domain. It also contains textual descriptions of these objects, default values and the relationships between the objects.

The first concept defined by the knowledge engineer acts as a 'root' concept for the knowledge base. It will be investigated by the guidance system if the user requests guidance immediately after starting a session. It should therefore contain a general description of the problem domain and direct the user, through a question and answer session, to those sub-problems which are found to be relevant.

As more knowledge is added the system will become more autonomous and less will be required of the user. It is therefore important that rules which lead from one concept to another do not force an unnatural approach on the user.

The manner in which the models are formulated directs the constraint refinement system. The knowledge engineer has to ensure that the models he employs in the system definition do not contain invariant factors on their left hand sides.

Note also that if certain algorithmic constructs are required in a knowledge base but cannot be implemented using the knowledge engineering language - because they need iteration, for instance - separate programs can be created and used as if they were normal ALFIE functions. More details on this can be found in the Knowledge Engineering Language Reference Guide and System Administrator's Reference Guide.

Conventions

In the following sections a number of syntactical conventions are employed:

Anything that appears in square brackets ([...]) is optional;

Anything that appears in diamond brackets (<...>) is determined by the user - the user leaves the diamond brackets out;

A vertical bar (|) means alternation (OR).

Thus,

<Concept> means the name of a concept,

<Rule> | <Model> means the name of a rule or model,

[ON | OFF] means ON or OFF or neither,

[<Command>] means the name of a command or nothing.

Note that all the commands and object names can be abbreviated. For object names to be abbreviated a '.' must be used at the end. The system will select the first match found. When two or more objects have the same name (eg. a concept and a rule might be called by the same name) and it is not clear from the context which is required, the system will question the user. Also, whenever the name of an object is optionally omitted the system assumes that the user means the last object, of the correct type, referred to by the user.

ACCOMMODATE

This command instructs the system to take account of any new information which has been imparted by the user (using CONSTRAIN or REFUTE). Accommodation involves re-examination of previously examined concepts which are now flagged as unvisited.

ADVISE <Lines of text> @

This command allows the user to send a message of unlimited length to the local system manager. It should be used for fault reporting and general feedback on the user's reaction to the system.

COMPLY [<Rule>]

This command instructs the system to perform the actions specified in the named rule when the conditions are true. If the conditions of the rule have not changed since it was last complied with, no action is taken, otherwise all the actions associated with true conditions are performed and where no condition is true an optional default action may be performed. When the rule has been complied with it is flagged as visited.

CONSTRAIN [<Factor>]

This command instructs the system to update the named factor with values elicited from the user. Any dependent factors are flagged as NOT RELIABLE and all of the dependent rules and their triggering concepts are flagged as unvisited (for use by the ACCOMMODATE and GUIDE commands).

DESCRIBE [<Concept> | <Rule> [-<Clause>] | <Model> | <Factor>]

This command instructs the system to display the textual description of the named object on the screen. If a supplementary description exists, the user is asked if he wants it.

Optionally, in the case of a rule, the user may specify a clause number in the format <Rule>-<Clause>. This will instruct the system to display the description associated with that particular WHEN part of the rule. If no description exists, the overall rule description will be delivered. By convention, the default clause of a rule is clause 0.

DETAIL [<Concept> | <Rule> [-<Clause>] | <Model> | <Factor>]

This command instructs the system to display the details of the named object on the screen. This includes both its defining characteristics and its relationship with other objects.

For a concept the details are a list of the related concepts (with their importance weights) and a list of the rules triggered from the concept.

For a rule the details are all of the <condition,action> clauses, the default clause and a list of the concepts which trigger the rule. Optionally, in the case of a rule, the user may specify a clause number in the format <Rule>-<Clause>. This will instruct the system to display the details of that particular WHEN part of the rule. By convention, the default clause of a rule is clause 0.

For a model the details are the defining expression and its status (ENABLED or DISABLED).

For a factor the details are its overall bounds (for a numerical factor), its current value and its status (DEFAULT VALUE, NOT RELIABLE, INFERRED BY ALFIE or GIVEN BY <User>) and a list of the factors currently dependent on it.

DISABLE [<Model>]

This command instructs the system to flag the named model as invalid in the current situation. This consists of disabling any models which are concurrent with the named model. Any factors derived from the models which are so disabled are given status NOT RELIABLE. This command should be used with care since it will, temporarily at least, overide the system's own selection of valid and invalid models.

ENABLE [<Model>]

This command instructs the system to flag the named model as valid in the current situation. This consists of enabling any models which are concurrent with the named model and disabling any models which are exclusive to the named model. Any factors derived from the models so enabled or disabled are given status NOT RELIABLE. This command should be used with care since it will, temporarily at least, overide the system's own selection of valid and invalid models.

EXAMINE [<Concept>]

This command instructs the system to examine the named concept. This means that the concept name is displayed followed by a verbal description of it and examination proceeds by complying with any rules associated with the concept. When this has been accomplished the concept is flagged as visited.

FINISH

This command instructs the system to terminate the session. The user will be asked if he wants a summary of the factor values and recommendations given in the most recent pieces of system advice. If the user answers in the affirmative the information is sent to a file specified by the user.

GUIDE

This command instructs the system to select and examine the concept which it considers most relevant to the design and which has not yet been examined. The most relevant concept is found by starting at the last concept which the user showed an interest in (or the 'root' concept if the user has not indicated his area of interest). Firstly, the guidance system looks for unvisited subconcepts, then for unvisited superconcepts and finally for unvisited isoconcepts. In the event of more than one concept from one of these categories being found the system uses the weights between the concepts to select the most relevant concept. If no concepts are found the system then selects the concept from all of the above categories with the highest weight as its new starting point and repeats the above procedure. The guidance system can thus be used repeatedly until it has examined every concept in the network.

HELP [<Command>]

This command, without a parameter, will display on the screen a resumee of the available commands. With a parameter, the format and function of the named command will be displayed on the screen.

INVOKE [<Model>]

This command instructs the system to attempt to execute the named model. This includes updating any factors required by the model. If the model is currently disabled the user is asked to verify the request. The system does not rely on the user to invoke the correct models for factor updates. It will place the new value into the factor if it can but will set the factor's status to DEFAULT VALUE so that it is re-evaluated using all of the valid models if it is needed later.

LIST [CONCEPTS | RULES | MODELS | FACTORS | ALL]

This command instructs the system to display a list of the concepts, rules, models and/or factors with a short description of each. If no parameter is given ALL is assumed.

REFUTE [<Factor>]

This command instructs the system to ignore the current value of the named factor and treat it as if it had its default value and status. As with the CONSTRAIN command, any dependent factors are flagged as NOT RELIABLE and all of the rules which use the factor along with their triggering concepts are flagged as unvisited (for use by the ACCOMMODATE and GUIDE commands).

RESET

This command instructs the system to completely reset itself. This involves refuting all the factors, disabling all the models, flagging all the concepts and rules as unvisited and resetting the current concept to the initial 'root' concept.

SAVE <File-name>

This command instructs the system to save a copy of the current state of the knowledge base and consultation in a file named <File-name>.ALFBIN. The information is saved in a binary form and may be used as a knowledge base definition file at the start of a subsequent session. Binary files can be loaded faster than normal definition files and are distinguished by the ALFBIN suffix.

SUMMARY

This command instructs the system to display on the screen a summary of all the pertinent factors with their values. Pertinent factors are those which the system has used and which are stated to be REPORTABLE in the knowledge definition file.

TRACE [ON | OFF]

This command instructs the system to turn tracing on or off. If no parameter is supplied the opposite of the current mode is selected. Initially tracing is turned off and the trace information is suppressed. When tracing is turned on the system lists its actions to the screen.

UPDATE [<Factor>]

This command instructs the system to attempt to re-evaluate the named factor. To do this, the system tries to enable all of the appropriate models by examining the validity of the rules which decree their enablement. It then invokes all of those models. Should no models be available the system will query the user for a value, if this is possible - ie. if the factor has an associated question.

?

This command performs two main functions depending on when it is issued. It can be issued anywhere - in response to questions, in response to continuation prompts and as a normal command. In this latter case it acts like the HELP command. When issued in response to questions and prompts from the system it

can elicit explanations of factor meanings and the system's line of reasoning.

This process is normally started when the system is asking for a value for a factor. If the user queries the system at this point there are obviously two questions which he may have in mind - *Can you explain the question in more detail?* or *Can you explain why the question is being asked?*.

The system therefore adopts the approach of first informing the user of what it is currently trying to do - in this case, obtain a value for a factor, which it names. It next informs of the user of the current value associated with the factor and its status - which will always be either DEFAULT or NOT RELIABLE. Finally it provides the description of the factor.

The system then offers the user the opportunity of a further explanation. This will be the reason for the request for the factor value. There are two possibilities here - the factor must have been referred to in a model or in a rule.

In the case of a model reference, the system will identify the name of the model and give its description. The system will then offer an explanation of why the model was used - which will be in order to update another factor, which it will name.

In the case of a rule reference, the system will identify the name and clause number of the rule and give either a description of the clause or, if that is not possible, the overall description of the rule. The system will then offer an explanation of why the rule was used - this could be as a result of it being necessary to validate a model or a direct consequence of a concept being

examined or some other rule being complied with.

When explaining why a concept was being investigated the possible reasons are the same as those for rules with the additional possibilities that either the guidance or accommodation system might have instigated the action.

Eventually the system, if continually pressed, will reach the point where a user instruction becomes the reason. At this point it will report that this was the case and cease to offer further explanations.

This same process is initiated when the user acknowledges a piece of advice with the '?' character.

Conventions

In the following sections a number of syntactical conventions are employed:

Anything that appears in square brackets ([...]) is optional;

Anything that appears in diamond brackets (<...>) is decided by the user - the user leaves the diamond brackets out;

<Log exp> means an expression which evaluates to logical true or false;

<Num exp> means an expression which evaluates to a numerical value;

A vertical bar (|) means alternation (OR).

Thus,

<Concept> means the name of a concept,

[(<Weight>)] means an optional weight enclosed within parentheses,

<Log exp> | <Num exp> means a logical expression or a numerical expression.

Defining a Concept

```
CONCEPT <Concept>
   [ DESCRIPTION <Lines of text> @ ]
   [ SUPPLEMENTARY <Further text> @ ]
   [ SUPERCONCEPT <Concept1> [ (<Weight1>) ] ]
   [ ISOCONCEPT   <Concept2> [ (<Weight2>) ] ]
   [ SUBCONCEPT   <Concept3> [ (<Weight3>) ] ]
   [ SELECT <Rule> ]
```

The CONCEPT keyword creates a concept called <Concept>. The subsequent keywords define it.

The DESCRIPTION keyword should be followed by a short description of the concept and the SUPPLEMENTARY keyword should give a full description which can be as long as necessary. The <Lines of text>, if present, will be used by the DESCRIBE command and the user will be asked if he requires the <Further text>.

The SUPERCONCEPT, ISOCONCEPT and SUBCONCEPT keywords define links to other concepts. There may be any number of each type including none. A weight is associated with each link. This permits the relative strengths of the relationships between concepts to be recorded. The weight should be between 0 and 1 and the default is 1. As well as providing a means for informing the user of the strength of the relationships between concepts, this weight is also used by the guidance system in its selection of the most relevant concept to direct the user to.

Note that the system will infer links based on the information that it has and if link information is duplicated under two concepts by the knowledge engineer the system will check it for consistency. For instance, if a concept Fred is defined to have a subconcept called Joe, the system will infer that the concept Joe has a superconcept called Fred and if the knowledge engineer attempts to state that the concept Joe has a subconcept or an isoconcept called Fred the inconsistency will be reported to the user.

The SELECT keyword, which can also be used as many times as required, links the concept to a rule which should be complied with when the concept is examined. These rules are complied with in the order in which they are SELECTed.

Defining a Rule

```
RULE <Rule>

   [ DESCRIPTION <Lines of text> @ ]

   [ SUPPLEMENTARY <Further text> @ ]

   [ WHEN <Log exp> : <Action> ]

   [ DESCRIPTION <Lines of text> @ ]

   [ DEFAULT <Action-d> ]

   [ DESCRIPTION <Lines of text> @ ]
```

The RULE keyword creates a rule called <Rule>. The subsequent keywords define it.

The first DESCRIPTION keyword should be followed by a short description of the rule and the SUPPLEMENTARY keyword should give a full description which can be as long as necessary. The <Lines of text>, if present, will be used by the DESCRIBE command and the user will be asked if he requires the <Further text>. Subsequent DESCRIPTIONs are used to describe individual clauses which can be used very profitably by the explanation facility as well as by the DESCRIBE command.

The WHEN keyword introduces a condition-action pair. There may be any number of these. When the rule is complied with, all the actions associated with true conditions are performed. When no condition is true a default action (<Action-d>) will be performed if the DEFAULT keyword has been used. For a description of the conditions (<Log exp>) see Logical Expressions below. Note

that any factor whose status is DEFAULT or NOT RELIABLE referred to in a rule condition is automatically updated and this is the recommended way in which to handle factor updates.

There are a number of types of <Action> that can be used:

EXAMINE <Concept> will lead to the examination of the concept <Concept> and the subsequent triggering of any associated rules;

COMPLY <Rule> will lead directly to triggering the rule <Rule>;

INVOKE <Model> will lead to the execution of model <Model> - Note that this is not recommended for general use since other models may be required to update a factor properly;

ENABLE <Model> will flag model <Model> as valid in the current situation for factor updates (see also Defining Sets of Models below);

DISABLE <Model> will flag model <Model> as invalid in the current situation for factor updates (see also Defining Sets of Models below);

UPDATE <Factor> will update factor <Factor> by invoking all the enabled models which contain <Factor> on their left hand sides. If there are no such models the system will ask the user. Note that this is not the recommended manner in which to update factors since the system can update factors automatically if they are placed in rule conditions;

ADVISE <Lines of text> @ will display the advice, <Lines of text>, to the user. If the lines include a string of the form [<Factor>], where <Factor> is a valid factor name then the system will output its value. If the factor is numerical and its minimum and maximum are the same, just one of the bounds is output as a number. If the factor is a set, its value is output in wiggly parentheses. If the factor is logical, its truth value is output.

Defining a Model

```
MODEL <Model>

   [ DESCRIPTION <Lines of text> @ ]

   [ SUPPLEMENTARY <Further text> @ ]

   EQUATION <Factor>  <-  <Log exp> | <Num exp> ;
```

The MODEL keyword creates a model called <Model>. The subsequent keywords define it.

The DESCRIPTION keyword should be followed by a short description of the model and the SUPPLEMENTARY keyword should give a full description which can be as long as necessary. The <Lines of text>, if present, will be used by the DESCRIBE command and the user will be asked if he requires the <Further text>.

The EQUATION keyword introduces the relationship that defines the model. When the model is invoked the factor <Factor> will be refined by the expression on the right hand side. In other words, the new value for factor <Factor> will be the intersection of its old value and the value evaluated from the expression on the right hand side. If the intersection is the null set the system will report an inconsistency in the constraints on the factor <Factor> when the model is invoked. For a definition of <Log exp> and <Num exp> see Logical Expressions and Numerical Expressions below.

NB. Models must not be recursive (ie. the factor on the left hand side of the model must not appear in the <Log exp> or <Num exp> on the right. Nor should mutually recursive models be enabled simultaneously - they should be made

exclusive (see Defining Sets of Models below).

Defining a Factor

The three types of factor have different definitions:

```
FACTOR <Factor> : LOGICAL [ (Default) ]
   [ QUESTION <Requesting text> @ ]
   [ DESCRIPTION <Lines of text> @ ]
   [ SUPPLEMENTARY <Further text> @ ]
   [ REPORTABLE ]

FACTOR <Factor> : NUMERICAL
   [ QUESTION <Requesting text> @ ]
   [ DESCRIPTION <Lines of text> @ ]
   [ SUPPLEMENTARY <Further text> @ ]
   [ REPORTABLE ]
   [ BOUNDS <Num exp1> : <Num exp2> ; ]

FACTOR <Factor> : SET
   [ QUESTION <Requesting text> @ ]
   [ DESCRIPTION <Lines of text> @ ]
   [ SUPPLEMENTARY <Further text> @ ]
   [ REPORTABLE ]
   [ MAXIMUM <m> OF {<Element1> <Element2> ... <Elementn>} ]
```

The FACTOR keyword creates a factor called <Factor>. The subsequent keywords define it.

The QUESTION keyword provides the text of the question that should be asked of the user if the user is needed to provide a value for the factor.

The DESCRIPTION keyword should be followed by a short description of the factor and the SUPPLEMENTARY keyword should give a full description which can be as long as necessary. The <Lines of text>, if present, will be used by the DESCRIBE command and the user will be asked if he requires the <Further text>. The <Lines of text> are output when the user asks for help during a request for a new value for the factor.

The REPORTABLE keyword, if present, informs the system that the factor is meaningful to the user (ie. not an intermediate variable in a calculation) and should be included in any summaries requested by the user.

The logical factor may contain an optional default value (true or false) and if this is not specified the factor is assumed to be false at startup.

The numerical factor may contain default bounds on its permitted range. These bounds are general numerical expressions, a description of which can be found under Numerical Expressions below. If no bounds are specified the factor is initially considered to range between negative and positive infinity.

The set factor comes in two forms depending on whether the MAXIMUM keyword is present.

If the MAXIMUM keyword is present then the set factor is limited to taking at most <m> values at any one time from a range of elements, <Element1>, <Element2>, etc. The default value for this form of the set factor is the universal

set of all its possibilities.

If the MAXIMUM keyword is **NOT** present then the factor can take as many strings as the user requires. This is a very general type of factor which can be used to hold lists generated by the user for assorted purposes. The default value for this form of the set factor is the empty set or list.

Defining Sets of Models

There are two keywords which define the relationships between models:

EXCLUSIVE <Model1> <Model2> ... <Modeln> @ This statement informs the system that only one of <Model1> ... <Modeln> should be active or enabled at any one time. Thus, if one of these models is ever enabled the others are all disabled automatically.

CONCURRENT <Model1> <Model2> ... <Modeln> @ This statement informs the system that the models <Model1> ... <Modeln> should be treated as a group - whenever one of them is enabled the others will be enabled automatically and, conversely, whenever one of them is disabled the others are automatically disabled as well.

Other Keywords

COMMENT <Lines of text> @ This statement allows the knowledge engineer to put explanatory comments in the knowledge definition file.

ECHO & NOECHO These two keywords turn echoing of the knowledge definition file during compilation on and off respectively. They are useful for debugging parts of a file.

TRACE <Factor> This statement instructs the system to keep an eye on the named factor when the system is being used. This takes the form of displaying the factor's details every time it is amended. This should only be done during system development.

Logical Expressions

These are expressions which return a logical result (true or false). They will therefore consist of logical values and logical operators, set values and set operators, numerical expressions and numerical relationships or a combination thereof.

```
Example: (P OR Q) AND (A IN B) AND NOT (X > Y)
Where P and Q are logical factors
      A and B are set factors
      X and Y are numerical factors
```

Numerical Expressions

These are expressions which return a numerical result. They will therefore consist of numerical values and numerical operators and set values and set operators.

```
Example: R * COS(X) + CARD(A)
Where R and X are numerical factors
      A is a set factor
```

Operators and Relationships

The operators and relationships supported within the system for use on factors in logical and numerical expressions are as follows:

MONADIC OPERATORS (Parameters must be placed in parentheses).

GIVEN Returns true if the factor status is, or can be made to be, GIVEN;

CURRENT Returns true if the factor status is INFERRED or GIVEN, else false - note that false will only be returned if no suitable model could be used and the user has been asked for, and declined to give, a value;

MIN Returns the lower bound of a numerical factor;

MAX Returns the upper bound of a numerical factor;

NOT Returns the logical negation of a logical factor;

CARD Returns the cardinality of the current set contained in a set factor;

LOG Returns the log (base 10) of its parameter;

ANTILOG Returns the antilog (base 10) of its parameter;

LN Returns the natural log of its parameter;

EXP Returns the exponential of its parameter;

-	Returns the negative of its parameter;
FACT	Returns the factorial of its parameter;
ABS	Returns the absolute value of its parameter;
SIN	Returns the sine of its parameter;
ARCSIN	Returns the principal value of the arcsine of its parameter;
COS	Returns the cosine of its parameter;
ARCCOS	Returns the principal value of the arccosine of its parameter;
TAN	Returns the tangent of its parameter;
ARCTAN	Returns the principal value of the arctangent of its parameter;

DIADIC SET OPERATOR

IN	Returns true if the first set is a subset of the second set - ie. every element of the first is an element of the second;

DIADIC NUMERICAL OPERATORS

^ or **	Returns the value of the first operand raised to the power of the second operand;
*	Returns the value of the first operand multiplied by the second operand;

/ Returns the value of the first operand divided by the second operand;

+ Returns the value of the first operand summed with the second operand;

- Returns the value of the first operand minus the second operand;

DIADIC LOGICAL OPERATORS

AND Returns the logical AND of its operands;

OR Returns the logical OR of its operands;

NUMERICAL RELATIONSHIPS

= Returns true if its operands are identical, else false;

<> Returns true if its operands are not identical, else false;

> Returns true if its first operand is strictly greater than its second operand, else false;

< Returns true if its first operand is strictly less than its second operand, else false;

>= Returns true if its first operand is greater than or equal to its second operand, else false;

<= Returns true if its first operand is less than or equal to its second operand, else false.

Expressions may also contain parentheses as required.

Programmable Functions

In addition to the operators already described, more complex algorithms can be employed by ALFIE. These should be compiled as separate programs and placed in a special directory by the system administrator. They will then be available for execution by ALFIE and will behave in the same way as internal functions such as SIN, although they may have any number of input parameters. For further information on setting up programmable functions the System Administrator's Reference Guide.

Operator Priorities

(GIVEN CURRENT NOT CARD MIN MAX)

(LOG ANTILOG LN EXP - FACT ABS)

(SIN ARCSIN COS ARCCOS TAN ARCTAN)

(<Programmable Functions>)

IN

^ **

* /

+ -

= <> > < >= <=

AND

OR

Values

The values used in expressions come in a number of forms. All types of value can be expressed by the appropriate type of factor variable. In addition constants can be used as shown in the following examples:

Logical constants : TRUE, FALSE

Numerical constants : 42, 1.1, [0,1], [3.14,9.81]

Set constants : {RED,GREEN,BLUE}

Names and Lines of Text

Names are limited to 64 characters in length. They must commence with a letter and contain only letters, digits and the '_' character.

Lines of text are limited to 80 characters in length.

The default knowledge base currently consists of the concept net displayed in Figure C.1. The major links between the concepts are indicated in this figure but to include all of the links would make the diagram unintelligible. The complete knowledge base contains 18 concepts; 121 rules; 166 models; 132 factors and requires slightly in excess of 700K bytes of storage when loaded into the system. The rest of this appendix is devoted to a brief description of each of the concepts. Appendix E contains a listing of the definition of the *INSPECTION* concept.

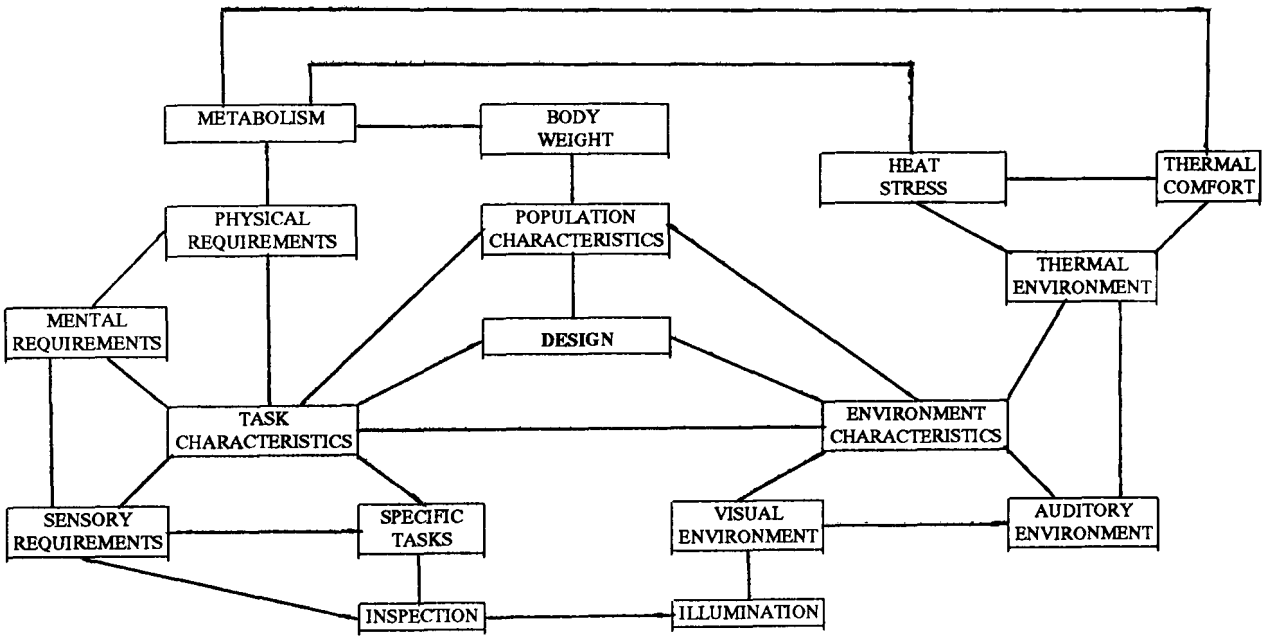

Figure C.1 *The Default Knowledge Base within ALFIE.*

Design

This is the 'root' concept of the knowledge base. It has associated rules which lead off in an exploration of the population, task and environmental characteristics of the operation being designed or assessed.

Population Characteristics

This concept leads to an examination of relevant population characteristics. Currently, only body weight is required for assessment criteria and so there is a single subconcept associated with this concept.

Body Weight

This concept permits an estimation of the operatives' body weight based on other, more easily gleaned, characteristics such as gender.

Task Characteristics

The knowledge base treats tasks in two modes: either by considering the types of action required of the operatives - the sensory, mental and physical characteristics of the task - or by considering specific tasks in a more holistic fashion - inspection for instance. This concept therefore leads off to an investigation of specific tasks or general task characteristics.

Specific Tasks

Since only one specific task can be handled currently - inspection - this concept checks whether the task is of an inspection type and investigates that concept if it is appropriate.

Inspection

This concept investigates the task of inspection. It is almost entirely concerned with selecting the right kind of lighting technique since this is a major factor in such tasks. The knowledge base specifically associated with this concept contains 10 rules; 29 models; 30 factors and occupies 71K bytes when loaded into the system. A complete definition of the concept may be found in Appendix E.

Sensory Requirements

This concept is included as a flag to future knowledge base developers. No work has been carried out in this particular area but it forms a fundamental part of the task breakdown when a specific task cannot be identified.

Mental Requirements

This concept is included as a flag to future knowledge base developers. No work has been carried out in this particular area but it forms a fundamental part of the task breakdown when a specific task cannot be identified.

Physical Requirements

The most important physical requirement of the knowledge base is the metabolic load imposed on the operatives as a result of the task. This concept therefore leads to an assessment of this parameter under its appropriate concept.

Metabolism

This concept can perform an estimation of the metablic load on the operatives. It first checks that this information is not directly available however. The knowledge base specifically associated with this concept contains 15 rules; 72 models; 16 factors and occupies 289K bytes of storage when loaded into the system.

Environment Characteristics

This concept instigates examination of the three major areas of environmental control. Rules lead the user to an awareness of auditory, visual and thermal considerations.

Auditory Environment

No work has been carried out so far in this area but the concept has been included as a flag for future workers to warn them that it should be considered as soon as possible.

Visual Environment

This concept leads to the investigation of various topics relevant to the visual environment as a whole. Particularly important amongst these are the illumination levels required for the operatives.

Illumination

This concept examines in detail those factors which affect the illumination requirements of the task. When possible actual lux levels are recommended, otherwise the user is advised to change the work situation in some manner. The knowledge base specifically associated with this concept contains 12 rules; 21 models; 13 factors and occupies 38K bytes of storage when loaded into the system.

Thermal Environment

This concept leads to an investigation of the problems posed by extreme thermal conditions and the problems of comfort in 'moderate' conditions.

Heat Stress

Potentially dangerous situations are investigated under this concept which draws information from many of the other concepts in the knowledge base. Recommendations are made on the suitability of the influencing factors. The knowledge base specifically associated with this concept contains 16 rules; 24 models; 17 factors and occupies 276K bytes of storage when loaded into the system.

Thermal Comfort

This concept is concerned with assessing the comfort, as a predicted mean vote of the operatives. Again, information is drawn from all over the default knowledge base. The knowledge base specifically associated with this concept contains 14 rules; 15 models; 25 factors and occupies 62K bytes of storage when loaded into the system.

This guide refers to the implementation on the PR1ME 9650 in the Department of Production Engineering and Production Management at the University of Nottingham.

The main directory, at MFD level, is called ALFIE*. Within this directory are five sub-directories:

.ALFIE

This directory contains two runfiles derived from PASCAL source and built using BIND. No extra libraries are required to create them - just the normal PASLIB and system libraries.

The ALFIE.RUN file is the main ALFIE program. A piece of CPL in CMDNC0 refers to the treename of this file so that all users can call up the program simply by typing ALFIE. This runfile, and therefore the users that invoke it, needs access to the sub-directories .DATA and .PROGS.

The USERMAN.RUN file is a system administration program which permits editing of users' details. Perhaps most useful are the facilities to change a user's experience level (downwards, since it increments automatically), change a user's privileges (normal users should have a low permission level), assign a user a 'friendly' name, and remove old users from the record. This runfile needs access to the sub-directory .DATA.

.DATA

This directory contains the default knowledge base files. Knowledge base files have two types of suffix: ALFDEF files are source files which can be edited and loaded into the system through the compilation sub-system; ALFBIN files are binary files which are created with the SAVE command. The latter will be loaded into the ALFIE system much quicker than the former but only files with the ALFBIN suffix will be treated as binary files - all others with be loaded through the compilation sub-system.

This directory also contains two data files: USERS.DATA contains information on the currently known users; UNITS.DATA holds information on conversions between a variety of common units and their SI equivalents. It is not currently used - so don't be lead astray by its existence.

.PROGS

This directory contains a number of source and runfiles which are the current complement of externally programmed functions which are described in the *INFO file contained in that directory. These programs use a very unsatisfactory method of communication with the main ALFIE program : data is passed from ALFIE through the file PARAMS.ALFIE and back to ALFIE through the file RESULT.ALFIE. Note that this **WILL** lead to problems if more than one user attempts to invoke an external function simultaneously.

Additional functions can be added by creating them with BIND and placing them in this directory. The prefix part of the name becomes the finction name in ALFIE. Remember that inputs to these programs should be drawn from the file PARAMS.ALFIE and the result of each function should be placed in the file RESULT.ALFIE.

.MANUALS

This directory contains the manuals associated with ALFIE version 5.4. They are contained in the files:

GUIDE.FRONT.5.4	Frontispiece and disclaimer
GUIDE.CONTENTS.5.4	Contents of all guides
GUIDE.UG.5.4	User Guide
GUIDE.KEG.5.4	Knowledge Engineering Guide
GUIDE.UCRG.5.4	User Command Reference Guide
GUIDE.KELRG.5.4	Knowledge Engineering Language Reference Guide
GUIDE.DKBRG.5.4	Default Knowledge Base Reference Guide
GUIDE.SARG.5.4	System Administrator's Reference Guide
GUIDE.BIB.5.4	Bibliography

.CODE

This directory contains the source files for the main ALFIE program and for the USERMAN administration program. There is currently a bug in the PASCAL compiler which leads to a severity 4 error in the compilation of EXPR_LIB. This can be avoided by compiling this module with Optimisation level 1.

The files with suffixes of EXTERN and GLOBAL are insert files - only the PASCAL files need compiling. The two programs are formed as follows:

ALFIE

PASCAL ALFIE

PASCAL SYSDEFN -EXT

PASCAL COMPSYS -EXT

PASCAL LOADSYS -EXT

PASCAL CONSULT -EXT

PASCAL CMND_LIB -EXT

PASCAL ERRS_LIB -EXT

PASCAL EXPR_LIB -EXT -OPT 1

PASCAL FILE_LIB -EXT

PASCAL INIT_LIB -EXT

PASCAL MATH_LIB -EXT

PASCAL MCDP_LIB -EXT

PASCAL NODE_LIB -EXT

PASCAL TEXT_LIB -EXT

PASCAL UNIT_LIB -EXT

PASCAL WORK_LIB -EXT

Then BIND them together with the PASLIB and system libraries.

USERMAN

PASCAL USERMAN

PASCAL USERFIX -EXT

PASCAL CMND_LIB -EXT

PASCAL ERRS_LIB -EXT

PASCAL EXPR_LIB -EXT -OPT 1

PASCAL FILE_LIB -EXT

PASCAL INIT_LIB -EXT

PASCAL MATH_LIB -EXT

PASCAL MCDP_LIB -EXT

PASCAL NODE_LIB -EXT

PASCAL TEXT_LIB -EXT

PASCAL UNIT_LIB -EXT

PASCAL WORK_LIB -EXT

Then BIND them together with the PASLIB and system libraries.

Appendix D

EXAMPLE ALFIE SESSION

This appendix contains details of a session with the ALFIE system using the knowledge base defined in Appendix E. This appendix has three parts.

The first part consists of a hard copy of an example dialogue with the ALFIE system. The purpose of the consultation was to identify lighting techniques for inspection tasks. The user's inputs are set in **boldface**.

At the end of the consultation the user requested a print-out of the constraints on the factors and the system's recommendations. This summary file forms the second part of the appendix.

During a consultation the ALFIE system keeps a trace of what is going on. This is stored in a file so that the user or, more likely, the knowledge engineer can follow the system's reasoning. The third part of the appendix contains the trace file for the example dialogue.

ALFIE

ALFIE Version 5.4 (31.03.88)
Department of Production Engineering and Production Management
University of Nottingham

The quality of the advice produced by this software is dependent on both the quality of the knowledge base employed and the validity of the responses given. The University of Nottingham therefore accepts no liability for any adverse effects which may result from its use.

Good afternoon Nick.

Please enter the name of the knowledge base which you wish to use.
To use the default knowledge base just hit RETURN : **INSPECTION**

The knowledge base contains : 1 concept

10 rules

29 models

30 factors

and uses 71K bytes of storage.

 # GUIDE

INSPECTION

This concept examines the problem of detecting a fault in an object.

ADVICE : A note about contrast and glare.
In order to maximise the contrast between the centre of interest and the background whilst ensuring that glare does not become a problem the centre of interest should ideally be three times brighter than the background or vice versa.
Diffuse light sources or diffusing screens can be used to reduce glare and unwanted surface reflections.

Enter '?' for an explanation or any other key to continue : **C**

Can the fault be detected with a measuring gauge of some sort ?

Please enter Yes or No.
If you do not know just hit RETURN : **NO**

Please select from the following categories the one that best describes
the type of fault being inspected for :
1. A chip in the object;
2. A scratch on the surface of the object;
3. A dent in the object;
4. A bump or lump on the surface of the object;
5. A bend in the object;
6. A colour difference (Eg. Matching materials or dyes);
7. Stresses and strains (Eg. In a piece of glass);
8. None of the above.

Please enter the number of the best category.
If you do not know just hit RETURN : **4**

ADVICE : A directional light can be used to produce highlighting and shadowing.

Enter '?' for an explanation or any other key to continue : **C**

ADVICE : A low angled directional light can be used to make the fault cast a shadow. This will increase the effective size of the fault.

Enter '?' for an explanation or any other key to continue : **C**

ADVICE : Note that the inspector may try to use touch to identify the fault.

Enter '?' for an explanation or any other key to continue : **C**

Is the fault being inspected for fluorescent ?

Please enter Yes or No.
If you do not know just hit RETURN : **NO**

Is the object being inspected fluorescent ?

Please enter Yes or No.
If you do not know just hit RETURN : **NO**

Is the object being inspected in motion ?

Please enter Yes or No.
If you do not know just hit RETURN : **YES**

Is the object being inspected transparent ?

Please enter Yes or No.
If you do not know just hit RETURN : **NO**

Is the fault being inspected for transparent ?

Please enter Yes or No.
If you do not know just hit RETURN : **NO**

Is the object being inspected shiny ?

Please enter Yes or No.
If you do not know just hit RETURN : **YES**

Is the fault which is being inspected for shiny ?

Please enter Yes or No.
If you do not know just hit RETURN : **YES**

What size is the object being inspected ?
Please select one of the following :
1. Small (No larger than a coin);
2. Medium (The size of a book);
3. Large (The size of a desk top);
4. Very Large (Larger than a desk top).

Please enter the number of the best category.
If you do not know just hit RETURN : **4**

ADVICE : Note that it may not be possible to employ a small localised light source.

Enter '?' for an explanation or any other key to continue : **C**

How small could the largest dimension of the fault be ?
Please give your answer in millimetres.

This must be no less than 0.000 and no more than 1000.000.
To specify a minimum and a maximum value enter them
in either order separated by spaces.
You may specify a single value if you wish.
If you do not know just hit RETURN : **10**

How far from the fault could the inspector's eye be ?
Please give your answer in centimetres.

This must be no less than 0.000 and no more than 1000.000.
To specify a minimum and a maximum value enter them
in either order separated by spaces.
You may specify a single value if you wish.
If you do not know just hit RETURN : **180 200**

ADVICE : Consider spotlighting at a low angle.
Bumps or dents in a surface will appear as points of light.
Scratches in a surface will appear as lines of light.
Caution: Spotlighting may produce glare.

Enter '?' for an explanation or any other key to continue : **C**

ADVICE : Consider stroboscopic light synchronised to light the object only at those times when a defect might occur.
Caution: Vertigo and hypnotic effects may occur with stroboscopic light.

Enter '?' for an explanation or any other key to continue : **C**

ADVICE : Consider diffuse reflection. That is, reflecting a white large area diffuse source of light off the surface of the object.
Changes in thickness in an object made of flat transparent layers will appear as a rainbow.

Enter '?' for an explanation or any other key to continue : **C**

Details of concept INSPECTION :

There are no more general concepts associated with INSPECTION.

There are no more detailed concepts associated with INSPECTION.

There are no other concepts associated with INSPECTION.

 # LIST FACTORS

FAULT_MEASURABLE
This factor records whether the fault can be measured with some sort of instrument.
OUTCOMES
This factor records the technique or techniques considered suitable for the inspection task described.

FAULT_TYPE
This factor records the type of fault being inspected for.
UNEVEN_SURFACE
This factor records whether the fault causes surface unevenness or not.
SAME
This is a dummy factor which records whether the fault and the object have the same characteristics or not.

Hit any key to continue : **C**

SHINY_FAULT
This factor records whether the fault is shiny or not.
SHINY_OBJECT
This factor records whether the object is shiny or not.
LIGHT_FAULT
This factor records whether the fault is light coloured or not.
LIGHT_OBJECT
This factor records whether the object is light coloured or not.
TRANSPARENT_FAULT
This factor records whether the fault is transparent or not.

Hit any key to continue : **C**

TRANSPARENT_OBJECT
This factor records whether the object is transparent or not.
FLUORESCENT_FAULT
This factor records whether the fault is fluorescent or not.
FLUORESCENT_OBJECT
This factor records whether the object is fluorescent or not.
MOVING_OBJECT
This factor records whether the object is moving or not.
OBJECT_SMALL
This factor records the fact that the object is small.

Hit any key to continue : **C**

OBJECT_MEDIUM
This factor records the fact that the object is medium sized.
OBJECT_VERY_LARGE
This factor records the fact that the object is very large.
OBJECT_LARGE
This factor records the fact that the object is large.
VISANG_MINUTE
This factor records the fact that the size of the visual angle is minute.
VISANG_VERY_FINE
This factor records the fact that the size of the visual angle is very fine.

Hit any key to continue : **C**

VISANG_FINE
This factor records the fact that the size of the visual angle is fine.
VISANG_MEDIUM_FINE
This factor records the fact that the size of the visual angle is medium fine.
VISANG_TOO_SMALL
This factor records the fact that the size of the visual angle is too small.
VIEWING_DISTANCE
This factor records the distance from the fault to the
inspector's eye, in centimetres.
DEEP_DENT
This factor records whether the dent is deep or not.

Hit any key to continue : **C**

VISUAL_ANGLE
This factor records the visual angle subtended at the eye
by the fault, in minutes of arc.
FAULT_SIZE
This factor records the smallest value which the largest
dimension of the fault might take, in millimetres.
VISANG_MEDIUM
This factor records the fact that the size of the visual angle is medium.
VISANG_LARGE
This factor records the fact that the size of the visual angle is large.
OBJECT_SIZE
This factor records the size of the object being inspected.

Hit any key to continue : **C**

 # CONSTRAIN FAULT_TYPE

Please select from the following categories the one that best describes
the type of fault being inspected for :
1. A chip in the object;
2. A scratch on the surface of the object;
3. A dent in the object;
4. A bump or lump on the surface of the object;
5. A bend in the object;
6. A colour difference (Eg. Matching materials or dyes);
7. Stresses and strains (Eg. In a piece of glass);
8. None of the above.

Please enter the number of the best category.

If you do not know just hit RETURN : **2**

 # ACCOMMODATE

ADVICE : The fault will alter the surface texture of the object and therefore affect its reflective properties.

Enter '?' for an explanation or any other key to continue : **C**

ADVICE : A low angled directional light will make the fault appear light on a dark background.
A high angled directional light will make the fault appear dark on a light background.
If there are no reasons why the latter should not be used then making the fault appear dark on a light background is preferable.

Enter '?' for an explanation or any other key to continue : **C**

ADVICE : Consider spotlighting at a low angle.
Bumps or dents in a surface will appear as points of light.
Scratches in a surface will appear as lines of light.
Caution: Spotlighting may produce glare.

Enter '?' for an explanation or any other key to continue : **C**

ADVICE : Consider stroboscopic light synchronised to light the object only at those times when a defect might occur.
Caution: Vertigo and hypnotic effects may occur with stroboscopic light.

Enter '?' for an explanation or any other key to continue : **C**

ADVICE : Consider diffuse reflection. That is, reflecting a white large area diffuse source of light off the surface of the object.
Changes in thickness in an object made of flat transparent layers will appear as a rainbow.

Enter '?' for an explanation or any other key to continue : **C**

 # CONSTRAIN FAULT_TYPE

Please select from the following categories the one that best describes the type of fault being inspected for :
1. A chip in the object;
2. A scratch on the surface of the object;
3. A dent in the object;

4. A bump or lump on the surface of the object;
5. A bend in the object;
6. A colour difference (Eg. Matching materials or dyes);
7. Stresses and strains (Eg. In a piece of glass);
8. None of the above.

Please enter the number of the best category.
If you do not know just hit RETURN : **3**

 # ACCOMMODATE

Is the dent very deep ?

Please enter Yes or No.
If you do not know just hit RETURN : **YES**

ADVICE : A directional light can be used to produce highlighting and shadowing.

Enter '?' for an explanation or any other key to continue : **?**

This advice is a consequence of clause 8 of the rule CHECK_FAULT_TYPE.

8. If the fault is a bump or a dent sufficient to produce
surface uneveness then a directional light can be used
to produce highlighting and shadowing (R13, R14, R46, R47, R90).

Do you want a further explanation ? **YES**

The rule CHECK_FAULT_TYPE was complied with as a consequence of clause 2 of the rule OUTCOME_LIGHTING.

2. The lighting requirements are dependent on the fault type.

Do you want a further explanation ? **NO**

ADVICE : Consider spotlighting at a low angle.
Bumps or dents in a surface will appear as points of light.
Scratches in a surface will appear as lines of light.
Caution: Spotlighting may produce glare.

Enter '?' for an explanation or any other key to continue : **C**

ADVICE : Consider stroboscopic light synchronised to light the object only at those times when a defect might occur.
Caution: Vertigo and hypnotic effects may occur with
stroboscopic light.

Enter '?' for an explanation or any other key to continue : **C**

ADVICE : Consider diffuse reflection. That is, reflecting a white
large area diffuse source of light off the surface of the object.
Changes in thickness in an object made of flat transparent layers
will appear as a rainbow.

Enter '?' for an explanation or any other key to continue : **C**

 # FINISH

Do you wish to file the summary of factor constraints and system advice ? **YES**

Please enter a file name for the summary : **SUMMARY**

Summary of ALFIE session for Nick at 15:21 on 07.09.90

Summary of factor FAULT_MEASURABLE :

This factor records whether the fault can be measured with some sort of instrument.

This value must be either True or False.
The current value is False.
This was stated by Nick.

Summary of factor OUTCOMES :

This factor records the technique or techniques considered suitable for the inspection task described.

This value should be no more than 18 of
{DAYLIGHT,BLACK_LIGHT,BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,
CROSSED_POLARISATION,DARK_FIELD_ILLUMINATION,
DIFFUSE_REFLECTION,EDGE_LIGHTING,FLUORESCENT,
MEASURING_GAUGE,MOIRE_PATTERNS,MOVING_IMAGES,
POLARISED_LIGHT,SHADOW_GRAPHING,SILHOUETTE,SPOTLIGHTING,
STROBOSCOPIC_LIGHTING,SURFACE_GRAZING,TRANSILLUMINATION}.
The current value is {DIFFUSE_REFLECTION,FLUORESCENT,
SILHOUETTE,SPOTLIGHTING,STROBOSCOPIC_LIGHTING}.
This was inferred by ALFIE.

Summary of factor FAULT_TYPE :

This factor records the type of fault being inspected for.

This value should be no more than 1 of
{CHIP,SCRATCH,DENT,BUMP,BEND,COLOUR,STRESSES,NONE}.
The current value is {DENT}.
This was stated by Nick.

Summary of factor SHINY_FAULT :

This factor records whether the fault is shiny or not.

This value must be either True or False.
The current value is True.
This was inferred by ALFIE.

Summary of factor SHINY_OBJECT :

This factor records whether the object is shiny or not.

This value must be either True or False.
The current value is True.
This was stated by Nick.

Summary of factor TRANSPARENT_FAULT :

This factor records whether the fault is transparent or not.

This value must be either True or False.
The current value is False.
This was inferred by ALFIE.

Summary of factor TRANSPARENT_OBJECT :

This factor records whether the object is transparent or not.

This value must be either True or False.
The current value is False.
This was stated by Nick.

Summary of factor FLUORESCENT_FAULT :

This factor records whether the fault is fluorescent or not.

This value must be either True or False.
The current value is False.
This was inferred by ALFIE.

Summary of factor FLUORESCENT_OBJECT :

This factor records whether the object is fluorescent or not.

This value must be either True or False.
The current value is False.
This was stated by Nick.

Summary of factor MOVING_OBJECT :

This factor records whether the object is moving or not.

This value must be either True or False.
The current value is True.
This was stated by Nick.

Summary of factor VIEWING_DISTANCE :

This factor records the distance from the fault to the
inspector's eye, in centimetres.

This value must be no less than 0.000 and no more than 1000.000.
The current value is between 180.000 and 200.000.
This was stated by Nick.

Summary of factor DEEP_DENT :

This factor records whether the dent is deep or not.

This value must be either True or False.
The current value is True.
This was stated by Nick.

Summary of factor VISUAL_ANGLE :

This factor records the visual angle subtended at the eye
by the fault, in minutes of arc.

This value must be no less than 0.000 and no more than 7200.000.
The current value is between 17.189 and 19.099.
This was inferred by ALFIE.

Summary of factor FAULT_SIZE :

This factor records the smallest value which the largest
dimension of the fault might take, in millimetres.

This value must be no less than 0.000 and no more than 1000.000.
The current value is 10.000.
This was stated by Nick.

Summary of factor OBJECT_SIZE :

This factor records the size of the object being inspected.

This value should be no more than 1 of
{SMALL,MEDIUM,LARGE,VERY_LARGE}.
The current value is {VERY_LARGE}.
This was stated by Nick.

Recommendations

Advice from rule GLARE_WARNING :

A note about contrast and glare.
In order to maximise the contrast between the centre of interest and the background whilst ensuring that glare does not become a problem the centre of interest should ideally be three times brighter than the background or vice versa.
Diffuse light sources or diffusing screens can be used to reduce glare and unwanted surface reflections.

Advice from rule RECOMMENDATIONS :

Consider spotlighting at a low angle.
Bumps or dents in a surface will appear as points of light.
Scratches in a surface will appear as lines of light.
Caution: Spotlighting may produce glare.

Advice from rule RECOMMENDATIONS :

Consider stroboscopic light synchronised to light the object only at those times when a defect might occur.
Caution: Vertigo and hypnotic effects may occur with stroboscopic light.

Advice from rule RECOMMENDATIONS :

Consider diffuse reflection. That is, reflecting a white large area diffuse source of light off the surface of the object.
Changes in thickness in an object made of flat transparent layers will appear as a rainbow.

Advice from rule CHECK_FAULT_TYPE :

A directional light can be used to produce highlighting and shadowing.

Advice from rule CHECK_SIZES :

Note that it may not be possible to employ a small localised light source.

End of summary.

Trace of ALFIE Session for Nick at 15:17 on 07.09.90

0 Nick invoked the guidance system.

1 Guidance system initiated investigation of INSPECTION.

2 Investigation of INSPECTION led to compliance with GLARE_WARNING.

3 Complying with clause 1 of GLARE_WARNING led to advising Nick.

2 Investigation of INSPECTION led to compliance with MEASURE_OR_INSPECT.

3 Evaluating condition in clause 1 of MEASURE_OR_INSPECT required updating FAULT_MEASURABLE.

4 Updating FAULT_MEASURABLE required an input from Nick.

4 Nick constrained FAULT_MEASURABLE to be False.

3 Complying with clause 0 of MEASURE_OR_INSPECT led to compliance with OUTCOME_LIGHTING.

4 Complying with clause 1 of OUTCOME_LIGHTING led to enabling LIGHTING_OUTCOME.

4 Complying with clause 2 of OUTCOME_LIGHTING led to compliance with CHECK_FAULT_TYPE.

5 Evaluating condition in clause 1 of CHECK_FAULT_TYPE required updating FAULT_TYPE.

6 Updating FAULT_TYPE required an input from Nick.

6 Nick constrained FAULT_TYPE to be {BUMP}.

5 Complying with clause 5 of CHECK_FAULT_TYPE led to enabling SURFACE_UNEVEN.

5 Evaluating condition in clause 8 of CHECK_FAULT_TYPE required updating UNEVEN_SURFACE.

6 Updating UNEVEN_SURFACE led to invocation of SURFACE_UNEVEN.

6 SURFACE_UNEVEN constrained UNEVEN_SURFACE to be True.

5 Complying with clause 8 of CHECK_FAULT_TYPE led to advising Nick.

5 Complying with clause 9 of CHECK_FAULT_TYPE led to advising Nick.

5 Complying with clause 11 of CHECK_FAULT_TYPE led to enabling ELIMINATE_DAYLIGHT.

5 Complying with clause 12 of CHECK_FAULT_TYPE led to advising Nick.

4 Complying with clause 3 of OUTCOME_LIGHTING led to compliance with CHECK_CHARACTERISTICS.

5 Evaluating condition in clause 1 of CHECK_CHARACTERISTICS required updating FLUORESCENT_FAULT.

6 Updating FLUORESCENT_FAULT required an input from Nick.

6 Nick constrained FLUORESCENT_FAULT to be False.

5 Evaluating condition in clause 1 of CHECK_CHARACTERISTICS required updating FLUORESCENT_OBJECT.

6 Updating FLUORESCENT_OBJECT required an input from Nick.

6 Nick constrained FLUORESCENT_OBJECT to be False.

5 Complying with clause 1 of CHECK_CHARACTERISTICS led to enabling ELIMINATE_BLACK_LIGHT.

5 Evaluating condition in clause 3 of CHECK_CHARACTERISTICS required updating MOVING_OBJECT.

6 Updating MOVING_OBJECT required an input from Nick.

6 Nick constrained MOVING_OBJECT to be True.

5 Evaluating condition in clause 5 of CHECK_CHARACTERISTICS required updating TRANSPARENT_OBJECT.

6 Updating TRANSPARENT_OBJECT required an input from Nick.

6 Nick constrained TRANSPARENT_OBJECT to be False.

5 Complying with clause 5 of CHECK_CHARACTERISTICS led to enabling NOT_TRANSPARENT_OBJECT.

5 Evaluating condition in clause 6 of CHECK_CHARACTERISTICS required updating TRANSPARENT_FAULT.

6 Updating TRANSPARENT_FAULT required an input from Nick.

6 Nick constrained TRANSPARENT_FAULT to be False.

5 Complying with clause 8 of CHECK_CHARACTERISTICS led to enabling ELIMINATE_TRANSILLUMINATION.

5 Evaluating condition in clause 13 of CHECK_CHARACTERISTICS required updating SHINY_OBJECT.

6 Updating SHINY_OBJECT required an input from Nick.

6 Nick constrained SHINY_OBJECT to be True.

5 Evaluating condition in clause 13 of CHECK_CHARACTERISTICS required updating SHINY_FAULT.

6 Updating SHINY_FAULT required an input from Nick.

6 Nick constrained SHINY_FAULT to be True.

4 Complying with clause 4 of OUTCOME_LIGHTING led to compliance with CHECK_SIZES.

5 Complying with clause 1 of CHECK_SIZES led to enabling CALCULATE_VISUAL_ANGLE.

6 Enabling CALCULATE_VISUAL_ANGLE led to enabling VERY_LARGE_OBJECT.

6 Enabling CALCULATE_VISUAL_ANGLE led to enabling LARGE_OBJECT.

6 Enabling CALCULATE_VISUAL_ANGLE led to enabling MEDIUM_OBJECT.

6 Enabling CALCULATE_VISUAL_ANGLE led to enabling SMALL_OBJECT.

6 Enabling CALCULATE_VISUAL_ANGLE led to enabling IMPOSSIBLE_VISANG.

6 Enabling CALCULATE_VISUAL_ANGLE led to enabling LARGE_VISANG.

6 Enabling CALCULATE_VISUAL_ANGLE led to enabling MEDIUM_VISANG.

6 Enabling CALCULATE_VISUAL_ANGLE led to enabling MEDIUM_FINE_VISANG.

6 Enabling CALCULATE_VISUAL_ANGLE led to enabling FINE_VISANG.

6 Enabling CALCULATE_VISUAL_ANGLE led to enabling VERY_FINE_VISANG.

6 Enabling CALCULATE_VISUAL_ANGLE led to enabling MINUTE_VISANG.

5 Evaluating condition in clause 2 of CHECK_SIZES required updating OBJECT_SMALL.

6 Updating OBJECT_SMALL led to invocation of SMALL_OBJECT.

7 Invocation of SMALL_OBJECT required updating OBJECT_SIZE.

8 Updating OBJECT_SIZE required an input from Nick.

8 Nick constrained OBJECT_SIZE to be {VERY_LARGE}.

6 SMALL_OBJECT constrained OBJECT_SMALL to be False.

5 Evaluating condition in clause 2 of CHECK_SIZES required updating OBJECT_MEDIUM.

6 Updating OBJECT_MEDIUM led to invocation of MEDIUM_OBJECT.

6 MEDIUM_OBJECT constrained OBJECT_MEDIUM to be False.

5 Evaluating condition in clause 3 of CHECK_SIZES required updating OBJECT_VERY_LARGE.

6 Updating OBJECT_VERY_LARGE led to invocation of VERY_LARGE_OBJECT.

6 VERY_LARGE_OBJECT constrained OBJECT_VERY_LARGE to be True.

5 Complying with clause 3 of CHECK_SIZES led to advising Nick.

5 Evaluating condition in clause 4 of CHECK_SIZES required updating OBJECT_LARGE.

6 Updating OBJECT_LARGE led to invocation of LARGE_OBJECT.

6 LARGE_OBJECT constrained OBJECT_LARGE to be False.

5 Evaluating condition in clause 6 of CHECK_SIZES required updating VISANG_MINUTE.

6 Updating VISANG_MINUTE led to invocation of MINUTE_VISANG.

7 Invocation of MINUTE_VISANG required updating VISUAL_ANGLE.

8 Updating VISUAL_ANGLE led to invocation of CALCULATE_VISUAL_ANGLE.

9 Invocation of CALCULATE_VISUAL_ANGLE required updating FAULT_SIZE.

10 Updating FAULT_SIZE required an input from Nick.

10 Nick constrained FAULT_SIZE to be 10.000.

9 Invocation of CALCULATE_VISUAL_ANGLE required updating VIEWING_DISTANCE.

10 Updating VIEWING_DISTANCE required an input from Nick.

10 Nick constrained VIEWING_DISTANCE to be between 180.000 and 200.000.

8 CALCULATE_VISUAL_ANGLE constrained VISUAL_ANGLE to be between 17.189 and 19.099.

6 MINUTE_VISANG constrained VISANG_MINUTE to be False.

5 Evaluating condition in clause 6 of CHECK_SIZES required updating VISANG_VERY_FINE.

6 Updating VISANG_VERY_FINE led to invocation of VERY_FINE_VISANG.

6 VERY_FINE_VISANG constrained VISANG_VERY_FINE to be False.

5 Evaluating condition in clause 6 of CHECK_SIZES required updating VISANG_FINE.

6 Updating VISANG_FINE led to invocation of FINE_VISANG.

6 FINE_VISANG constrained VISANG_FINE to be False.

5 Evaluating condition in clause 6 of CHECK_SIZES required updating VISANG_MEDIUM_FINE.

6 Updating VISANG_MEDIUM_FINE led to invocation of MEDIUM_FINE_VISANG.

6 MEDIUM_FINE_VISANG constrained VISANG_MEDIUM_FINE to be False.

5 Evaluating condition in clause 8 of CHECK_SIZES required updating VISANG_TOO_SMALL.

6 Updating VISANG_TOO_SMALL led to invocation of IMPOSSIBLE_VISANG.

6 IMPOSSIBLE_VISANG constrained VISANG_TOO_SMALL to be False.

2 Investigation of INSPECTION led to compliance with RECOMMENDATIONS.

3 Complying with clause 1 of RECOMMENDATIONS led to enabling EXPERTS_OUTCOMES.

3 Evaluating condition in clause 2 of RECOMMENDATIONS required updating OUTCOMES.

4 Updating OUTCOMES led to invocation of LIGHTING_OUTCOME.

4 LIGHTING_OUTCOME constrained OUTCOMES to be
{DAYLIGHT,BLACK_LIGHT,BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,
CROSSED_POLARISATION,DARK_FIELD_ILLUMINATION,DIFFUSE_REFLECTION,EDGE_LIGHTING,
FLUORESCENT,MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,SHADOW_GRAPHING,
SILHOUETTE,SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING,
TRANSILLUMINATION}.

4 Updating OUTCOMES led to invocation of ELIMINATE_DAYLIGHT.

4 ELIMINATE_DAYLIGHT constrained OUTCOMES to be
{BLACK_LIGHT,BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,
CROSSED_POLARISATION,DARK_FIELD_ILLUMINATION,DIFFUSE_REFLECTION,EDGE_LIGHTING,
FLUORESCENT,MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,SHADOW_GRAPHING,
SILHOUETTE,SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING,
TRANSILLUMINATION}.

4 Updating OUTCOMES led to invocation of ELIMINATE_BLACK_LIGHT.

4 ELIMINATE_BLACK_LIGHT constrained OUTCOMES to be
{BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,CROSSED_POLARISATION,
DARK_FIELD_ILLUMINATION,DIFFUSE_REFLECTION,EDGE_LIGHTING,FLUORESCENT,
MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,SHADOW_GRAPHING,SILHOUETTE,
SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING,TRANSILLUMINATION}.

4 Updating OUTCOMES led to invocation of NOT_TRANSPARENT_OBJECT.

4 NOT_TRANSPARENT_OBJECT constrained OUTCOMES to be
{BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,DARK_FIELD_ILLUMINATION,
DIFFUSE_REFLECTION,FLUORESCENT,MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,
SILHOUETTE,SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING,
TRANSILLUMINATION}.

4 Updating OUTCOMES led to invocation of ELIMINATE_TRANSILLUMINATION.

4 ELIMINATE_TRANSILLUMINATION constrained OUTCOMES to be
{BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,DARK_FIELD_ILLUMINATION,
DIFFUSE_REFLECTION,FLUORESCENT,MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,
SILHOUETTE,SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING}.

4 Updating OUTCOMES led to invocation of EXPERTS_OUTCOMES.

4 EXPERTS_OUTCOMES constrained OUTCOMES to be
{DIFFUSE_REFLECTION,FLUORESCENT,SILHOUETTE,SPOTLIGHTING,
STROBOSCOPIC_LIGHTING}.

3 Complying with clause 6 of RECOMMENDATIONS led to advising Nick.

3 Complying with clause 8 of RECOMMENDATIONS led to advising Nick.

3 Complying with clause 14 of RECOMMENDATIONS led to advising Nick.

0 Nick initiated constrainment of FAULT_TYPE.

0 Nick constrained FAULT_TYPE to be {SCRATCH}.

0 Nick requested the system to accommodate all new information.

1 Accommodation system found it necessary to investigate INSPECTION.

2 Investigation of INSPECTION led to compliance with MEASURE_OR_INSPECT.

3 Complying with clause 0 of MEASURE_OR_INSPECT led to compliance with OUTCOME_LIGHTING.

4 Complying with clause 2 of OUTCOME_LIGHTING led to compliance with CHECK_FAULT_TYPE.

5 Complying with clause 3 of CHECK_FAULT_TYPE led to compliance with SCRATCH_OR_CHIP.

6 Complying with clause 1 of SCRATCH_OR_CHIP led to enabling FAULT_IS_MATT.

6 Complying with clause 2 of SCRATCH_OR_CHIP led to advising Nick.

5 Evaluating condition in clause 8 of CHECK_FAULT_TYPE required updating UNEVEN_SURFACE.

6 Updating UNEVEN_SURFACE led to invocation of SURFACE_UNEVEN.

6 SURFACE_UNEVEN constrained UNEVEN_SURFACE to be True.

4 Complying with clause 3 of OUTCOME_LIGHTING led to compliance with CHECK_CHARACTERISTICS.

5 Evaluating condition in clause 13 of CHECK_CHARACTERISTICS required updating SHINY_FAULT.

6 Updating SHINY_FAULT led to invocation of FAULT_IS_MATT.

6 FAULT_IS_MATT constrained SHINY_FAULT to be False.

5 Complying with clause 13 of CHECK_CHARACTERISTICS led to advising Nick.

2 Investigation of INSPECTION led to compliance with RECOMMENDATIONS.

3 Evaluating condition in clause 2 of RECOMMENDATIONS required updating OUTCOMES.

4 Updating OUTCOMES led to invocation of LIGHTING_OUTCOME.

4 LIGHTING_OUTCOME constrained OUTCOMES to be
{DAYLIGHT,BLACK_LIGHT,BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,
CROSSED_POLARISATION,DARK_FIELD_ILLUMINATION,DIFFUSE_REFLECTION,EDGE_LIGHTING,
FLUORESCENT,MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,SHADOW_GRAPHING,
SILHOUETTE,SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING,
TRANSILLUMINATION}.

4 Updating OUTCOMES led to invocation of ELIMINATE_DAYLIGHT.

4 ELIMINATE_DAYLIGHT constrained OUTCOMES to be
{BLACK_LIGHT,BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,
CROSSED_POLARISATION,DARK_FIELD_ILLUMINATION,DIFFUSE_REFLECTION,EDGE_LIGHTING,
FLUORESCENT,MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,SHADOW_GRAPHING,
SILHOUETTE,SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING,
TRANSILLUMINATION}.

4 Updating OUTCOMES led to invocation of ELIMINATE_BLACK_LIGHT.

4 ELIMINATE_BLACK_LIGHT constrained OUTCOMES to be
{BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,CROSSED_POLARISATION,
DARK_FIELD_ILLUMINATION,DIFFUSE_REFLECTION,EDGE_LIGHTING,FLUORESCENT,
MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,SHADOW_GRAPHING,SILHOUETTE,
SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING,TRANSILLUMINATION}.

4 Updating OUTCOMES led to invocation of NOT_TRANSPARENT_OBJECT.

4 NOT_TRANSPARENT_OBJECT constrained OUTCOMES to be
{BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,DARK_FIELD_ILLUMINATION,
DIFFUSE_REFLECTION,FLUORESCENT,MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,
SILHOUETTE,SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING,
TRANSILLUMINATION}.

4 Updating OUTCOMES led to invocation of ELIMINATE_TRANSILLUMINATION.

4 ELIMINATE_TRANSILLUMINATION constrained OUTCOMES to be
{BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,DARK_FIELD_ILLUMINATION,
DIFFUSE_REFLECTION,FLUORESCENT,MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,
SILHOUETTE,SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING}.

4 Updating OUTCOMES led to invocation of EXPERTS_OUTCOMES.

4 EXPERTS_OUTCOMES constrained OUTCOMES to be
{DIFFUSE_REFLECTION,FLUORESCENT,SILHOUETTE,SPOTLIGHTING,
STROBOSCOPIC_LIGHTING}.

3 Complying with clause 6 of RECOMMENDATIONS led to advising Nick.

3 Complying with clause 8 of RECOMMENDATIONS led to advising Nick.

3 Complying with clause 14 of RECOMMENDATIONS led to advising Nick.

0 Nick initiated constrainment of FAULT_TYPE.

0 Nick constrained FAULT_TYPE to be {DENT}.

0 Nick requested the system to accommodate all new information.

1 Accommodation system found it necessary to investigate INSPECTION.

2 Investigation of INSPECTION led to compliance with MEASURE_OR_INSPECT.

3 Complying with clause 0 of MEASURE_OR_INSPECT led to compliance with OUTCOME_LIGHTING.

4 Complying with clause 2 of OUTCOME_LIGHTING led to compliance with CHECK_FAULT_TYPE.

5 Complying with clause 2 of CHECK_FAULT_TYPE led to enabling SAME_CHARACTERISTICS.

6 Enabling SAME_CHARACTERISTICS led to enabling SAME_FLUORESCENT.

6 Enabling SAME_CHARACTERISTICS led to enabling SAME_TRANSPARENT.

6 Enabling SAME_CHARACTERISTICS led to enabling SAME_LIGHT.

6 Enabling SAME_CHARACTERISTICS led to enabling SAME_SHINY.

5 Complying with clause 4 of CHECK_FAULT_TYPE led to disabling FAULT_IS_MATT.

5 Complying with clause 6 of CHECK_FAULT_TYPE led to enabling SURFACE_EVEN.

6 Enabling SURFACE_EVEN led to disabling SURFACE_UNEVEN.

5 Complying with clause 7 of CHECK_FAULT_TYPE led to compliance with CHECK_DEPTH_OF_FAULT.

6 Evaluating condition in clause 1 of CHECK_DEPTH_OF_FAULT required updating DEEP_DENT.

7 Updating DEEP_DENT required an input from Nick.

7 Nick constrained DEEP_DENT to be True.

6 Complying with clause 1 of CHECK_DEPTH_OF_FAULT led to enabling SURFACE_UNEVEN.

7 Enabling SURFACE_UNEVEN led to disabling SURFACE_EVEN.

5 Evaluating condition in clause 8 of CHECK_FAULT_TYPE required updating UNEVEN_SURFACE.

6 Updating UNEVEN_SURFACE led to invocation of SURFACE_UNEVEN.

6 SURFACE_UNEVEN constrained UNEVEN_SURFACE to be True.

5 Complying with clause 8 of CHECK_FAULT_TYPE led to advising Nick.

4 Complying with clause 3 of OUTCOME_LIGHTING led to compliance with CHECK_CHARACTERISTICS.

5 Evaluating condition in clause 1 of CHECK_CHARACTERISTICS required updating FLUORESCENT_FAULT.

6 Updating FLUORESCENT_FAULT led to invocation of SAME_FLUORESCENT.

6 SAME_FLUORESCENT constrained FLUORESCENT_FAULT to be False.

5 Evaluating condition in clause 6 of CHECK_CHARACTERISTICS required updating TRANSPARENT_FAULT.

6 Updating TRANSPARENT_FAULT led to invocation of SAME_TRANSPARENT.

6 SAME_TRANSPARENT constrained TRANSPARENT_FAULT to be False.

5 Evaluating condition in clause 13 of CHECK_CHARACTERISTICS required updating SHINY_FAULT.

6 Updating SHINY_FAULT led to invocation of SAME_SHINY.

6 SAME_SHINY constrained SHINY_FAULT to be True.

2 Investigation of INSPECTION led to compliance with RECOMMENDATIONS.

3 Evaluating condition in clause 2 of RECOMMENDATIONS required updating OUTCOMES.

4 Updating OUTCOMES led to invocation of LIGHTING_OUTCOME.

4 LIGHTING_OUTCOME constrained OUTCOMES to be
{DAYLIGHT,BLACK_LIGHT,BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,
CROSSED_POLARISATION,DARK_FIELD_ILLUMINATION,DIFFUSE_REFLECTION,EDGE_LIGHTING,
FLUORESCENT,MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,SHADOW_GRAPHING,

SILHOUETTE,SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING,
TRANSILLUMINATION}.

4 Updating OUTCOMES led to invocation of ELIMINATE_DAYLIGHT.

4 ELIMINATE_DAYLIGHT constrained OUTCOMES to be
{BLACK_LIGHT,BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,
CROSSED_POLARISATION,DARK_FIELD_ILLUMINATION,DIFFUSE_REFLECTION,EDGE_LIGHTING,
FLUORESCENT,MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,SHADOW_GRAPHING,
SILHOUETTE,SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING,
TRANSILLUMINATION}.

4 Updating OUTCOMES led to invocation of ELIMINATE_BLACK_LIGHT.

4 ELIMINATE_BLACK_LIGHT constrained OUTCOMES to be
{BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,CROSSED_POLARISATION,
DARK_FIELD_ILLUMINATION,DIFFUSE_REFLECTION,EDGE_LIGHTING,FLUORESCENT,
MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,SHADOW_GRAPHING,SILHOUETTE,
SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING,TRANSILLUMINATION}.

4 Updating OUTCOMES led to invocation of NOT_TRANSPARENT_OBJECT.

4 NOT_TRANSPARENT_OBJECT constrained OUTCOMES to be
{BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,DARK_FIELD_ILLUMINATION,
DIFFUSE_REFLECTION,FLUORESCENT,MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,
SILHOUETTE,SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING,
TRANSILLUMINATION}.

4 Updating OUTCOMES led to invocation of ELIMINATE_TRANSILLUMINATION.

4 ELIMINATE_TRANSILLUMINATION constrained OUTCOMES to be
{BRIGHTNESS_PATTERNS,CONVERGENT_LIGHT,DARK_FIELD_ILLUMINATION,
DIFFUSE_REFLECTION,FLUORESCENT,MOIRE_PATTERNS,MOVING_IMAGES,POLARISED_LIGHT,
SILHOUETTE,SPOTLIGHTING,STROBOSCOPIC_LIGHTING,SURFACE_GRAZING}.

4 Updating OUTCOMES led to invocation of EXPERTS_OUTCOMES.

4 EXPERTS_OUTCOMES constrained OUTCOMES to be
{DIFFUSE_REFLECTION,FLUORESCENT,SILHOUETTE,SPOTLIGHTING,
STROBOSCOPIC_LIGHTING}.

3 Complying with clause 6 of RECOMMENDATIONS led to advising Nick.

3 Complying with clause 8 of RECOMMENDATIONS led to advising Nick.

3 Complying with clause 14 of RECOMMENDATIONS led to advising Nick.

Session ended at 15:21 on 07.09.90.

Appendix E

EXAMPLE KNOWLEDGE BASE DEFINITION

This appendix contains the knowledge base definition file for the knowledge base consulted in Appendix D. It was written by the author of the thesis with the aid of an expert practitioner. Throughout the file numerical references have been made to the rules which the expert provided. This was done to assist the expert in identifying rules which she was not happy with when she validated the knowledge base.

```
COMMENT INSPECTION LIGHTING KNOWLEDGE BASE DERIVED FROM EXPERT LB @

CONCEPT INSPECTION

DESCRIPTION This concept examines the problem of detecting a fault in an object.@

  SELECT GLARE_WARNING

  SELECT MEASURE_OR_INSPECT

  SELECT RECOMMENDATIONS

RULE MEASURE_OR_INSPECT

DESCRIPTION This rule determines whether the fault can be measured or not.@

  SUPPLEMENTARY The first rule is derived from pilot expert rule 7.@

  WHEN FAULT_MEASURABLE : ENABLE MEASURE_OUTCOME

  DESCRIPTION If the fault can be measured do not worry about special

        lighting (R7).@

  DEFAULT COMPLY OUTCOME_LIGHTING

  DESCRIPTION If the fault cannot be measured examine the various forms of

        illumination which may aid the inspection task.@

MODEL MEASURE_OUTCOME

DESCRIPTION This model restricts the possible techniques to some

      form of measuring instrument.@

  EQUATION OUTCOMES <- {MEASURING_GAUGE};

RULE OUTCOME_LIGHTING

DESCRIPTION Record the fact that a measuring gauge is not suitable

      and investigate the object and fault characteristics.@

  WHEN TRUE : ENABLE LIGHTING_OUTCOME

  DESCRIPTION Since the fault cannot be measured consider some sort
```

of special lighting.@

WHEN TRUE : COMPLY CHECK_FAULT_TYPE

DESCRIPTION The lighting requirements are dependent on the fault type.@

WHEN TRUE : COMPLY CHECK_CHARACTERISTICS

DESCRIPTION The lighting requirements are dependent on the various

characteristics of the fault and the object.@

WHEN TRUE : COMPLY CHECK_SIZES

DESCRIPTION The lighting requirements are dependent on the sizes of

both the fault and the object.@

MODEL LIGHTING_OUTCOME

DESCRIPTION This model eliminates measuring gauges from the possible

outcomes.@

EQUATION OUTCOMES <-

{DAYLIGHT, BLACK_LIGHT, BRIGHTNESS_PATTERNS, CONVERGENT_LIGHT,

CROSSED_POLARISATION, DARK_FIELD_ILLUMINATION, DIFFUSE_REFLECTION,

EDGE_LIGHTING, FLUORESCENT, MOIRE_PATTERNS, MOVING_IMAGES,

POLARISED_LIGHT, SHADOW_GRAPHING, SILHOUETTE, SPOTLIGHTING,

STROBOSCOPIC_LIGHTING, SURFACE_GRAZING, TRANSILLUMINATION};

RULE CHECK_FAULT_TYPE

DESCRIPTION Examine the nature of the fault with a view to determining

any special lighting requirements.@

SUPPLEMENTARY These rules are derived from pilot expert rules 9, 10, 11, 13, 14,

45, 46, 47, 88, 90, 101, 102, 107 and 122.@

WHEN NOT (FAULT_TYPE IN {DENT,BEND,STRESSES}) : DISABLE SAME_CHARACTERISTICS

DESCRIPTION If the fault is not a bend or a dent or stresses and strains

then the fault characteristics cannot be assumed to be the

same as the object characteristics (R102).@

WHEN FAULT_TYPE IN {DENT,BEND,STRESSES} : ENABLE SAME_CHARACTERISTICS

DESCRIPTION If the fault is a bend or a dent or stresses and strains

then the fault characteristics can be assumed to be the

same as the object characteristics (R102).@

WHEN FAULT_TYPE IN {SCRATCH, CHIP} : COMPLY SCRATCH_OR_CHIP

WHEN NOT (FAULT_TYPE IN {SCRATCH, CHIP}) : DISABLE FAULT_IS_MATT

DESCRIPTION If the fault is not a scratch or a chip it cannot be

assumed to be matt.@

WHEN FAULT_TYPE IN {SCRATCH, CHIP, BUMP} : ENABLE SURFACE_UNEVEN

DESCRIPTION If the fault is a scratch, chip or bump it will produce

surface uneveness (R45).@

WHEN NOT (FAULT_TYPE IN {SCRATCH, CHIP, BUMP}) : ENABLE SURFACE_EVEN

DESCRIPTION If the fault is not a scratch, chip or bump then it cannot

be assumed to produce surface uneveness.@

WHEN FAULT_TYPE IN {DENT} : COMPLY CHECK_DEPTH_OF_FAULT

WHEN UNEVEN_SURFACE AND FAULT_TYPE IN {BUMP,DENT} : ADVISE

A directional light can be used to produce highlighting and shadowing.@

DESCRIPTION If the fault is a bump or a dent sufficient to produce

surface uneveness then a directional light can be used

to produce highlighting and shadowing (R13, R14, R46, R47, R90).@

WHEN FAULT_TYPE IN {BUMP} : ADVISE

A low angled directional light can be used to make the fault cast a

shadow. This will increase the effective size of the fault.@

DESCRIPTION If the fault is a bump then a low angled light will produce

shadowing (R9, R88).

If the fault has a shadow then the size of the fault will be

increased (R10).@

WHEN FAULT_TYPE IN {COLOUR} : COMPLY COLOUR_FAULT

WHEN NOT (FAULT_TYPE IN {COLOUR}) : ENABLE ELIMINATE_DAYLIGHT

DESCRIPTION If colour is not important then do not consider

daylight or an approximation to it (R122).@

WHEN FAULT_TYPE IN {BUMP} : ADVISE

Note that the inspector may try to use touch to identify the fault.@

DESCRIPTION If the fault is raised the inspector may use tactile

sensing (R11).@

WHEN FAULT_TYPE IN {BEND} : ADVISE

If the object is not flat then it is advisable to allow the inspector

to manipulate the object if at all possible.

It is also worth considering the use of template.@

DESCRIPTION If the fault is a bend and the object is 3-D consider letting

the inspector manipulate the object (R101).

Also, if the fault is a bend, a template might be used (R107).@

MODEL SAME_CHARACTERISTICS

DESCRIPTION This is a dummy model whose concurrency with other models

allows the fault characteristics to be set to the object

characteristics.@

EQUATION SAME <- TRUE;

FACTOR SAME : LOGICAL

DESCRIPTION This is a dummy factor which records whether the fault and the

object have the same characteristics or not.@

MODEL SAME_SHINY

DESCRIPTION This model states that if the object is shiny then so is the fault.@

EQUATION SHINY_FAULT <- SHINY_OBJECT;

MODEL SAME_LIGHT

DESCRIPTION This model states that if the object is light coloured then so is the fault.@

EQUATION LIGHT_FAULT <- LIGHT_OBJECT;

MODEL SAME_TRANSPARENT

DESCRIPTION This model states that if the object is transparent then so is the fault.@

EQUATION TRANSPARENT_FAULT <- TRANSPARENT_OBJECT;

MODEL SAME_FLUORESCENT

DESCRIPTION This model states that if the object is fluorescent then so is the fault.@

EQUATION FLUORESCENT_FAULT <- FLUORESCENT_OBJECT;

RULE CHECK_CHARACTERISTICS

DESCRIPTION This rule checks the object and fault characteristics.@

SUPPLEMENTARY These rules are derived from pilot expert rules 19, 23, 39, 49,

50, 51, 54, 59, 93, 94, 96, 114, 117, 118, 119, 120 and 121.@

WHEN NOT FLUORESCENT_FAULT AND NOT FLUORESCENT_OBJECT :

ENABLE ELIMINATE_BLACK_LIGHT

DESCRIPTION If neither the fault nor the object are fluorescent then black

light can be ruled out (R121).@

WHEN FLUORESCENT_FAULT OR FLUORESCENT_OBJECT : DISABLE ELIMINATE_BLACK_LIGHT

DESCRIPTION If either the fault or the object is fluorescent then black

light cannot be ruled out.@

WHEN NOT MOVING_OBJECT : ENABLE NOT_MOVING_OBJECT

DESCRIPTION If the object is not moving then stroboscopic light and

moving light images can be ruled out (R120).@

WHEN MOVING_OBJECT : DISABLE NOT_MOVING_OBJECT

DESCRIPTION If the object is moving then stroboscopic light and

moving light images cannot be ruled out.@

WHEN NOT TRANSPARENT_OBJECT : ENABLE NOT_TRANSPARENT_OBJECT

DESCRIPTION If the object is opaque then crossed polarisation,

edge lighting and shadow graphing can be ruled out (R118).@

WHEN TRANSPARENT_FAULT AND NOT TRANSPARENT_OBJECT : ADVISE

A diffusing screen over a number of incandescent light sources could

be used but glare may prove to be a problem.@

DESCRIPTION If the fault is transparent and the object is opaque then

a diffusing screen over a number of incandescent sources

should be considered and glare should be examined (R39).@

WHEN TRANSPARENT_OBJECT : DISABLE NOT_TRANSPARENT_OBJECT

DESCRIPTION If the object is transparent then crossed polarisation,

edge lighting and shadow graphing cannot be ruled out.@

WHEN NOT TRANSPARENT_OBJECT AND NOT TRANSPARENT_FAULT :

ENABLE ELIMINATE_TRANSILLUMINATION

DESCRIPTION If the object and the fault are both opaque then

transillumination can be ruled out (R119).@

WHEN TRANSPARENT_OBJECT OR TRANSPARENT_FAULT :

DISABLE ELIMINATE_TRANSILLUMINATION

DESCRIPTION If either the object or the fault is transparent then

transillumination cannot be ruled out (R114, R117).@

WHEN (TRANSPARENT_FAULT AND NOT TRANSPARENT_OBJECT) OR

(TRANSPARENT_OBJECT AND NOT TRANSPARENT_FAULT) : ADVISE

Transmitting light through the object will maximise contrast.@

DESCRIPTION If the fault is transparent and the object is opaque

or the object is opaque and the fault is transparent

then transmitting light through the object will

maximise contrast (R49).@

WHEN TRANSPARENT_OBJECT AND NOT TRANSPARENT_FAULT : ADVISE

The light source should have a beam wide enough to fill the

inspector's entire field of view.@

DESCRIPTION If the object is transparent and light is being

transmitted through it then the beam of the source

should be wide enough to fill the inspector's entire

field of view (R54).@

WHEN FAULT_TYPE IN {STRESSES} AND TRANSPARENT_OBJECT : ADVISE

Transmitting a patterned image through the object will produce

distortions caused by any stresses and strains in the object.@

DESCRIPTION If the fault is stresses and strains then it will distort

light transmitted through it (R51) and transmitting a

patterned image will accentuate the distortions (R50).@

WHEN SHINY_OBJECT AND NOT SHINY_FAULT : ADVISE

A low angled directional light will make the fault appear light on a

dark background.

A high angled directional light will make the fault appear dark on a

light background.

If there are no reasons why the latter should not be used then

making the fault appear dark on a light background is preferable.@

DESCRIPTION If the fault is matt and the object is shiny then a

low angled directional light will make the fault appear

light on a dark background (R23) and a high angled directional

light will make it appear dark on a light background (R19).

The latter is the preferred approach (R59).@

WHEN SHINY_FAULT AND NOT SHINY_OBJECT : ADVISE

A directional light can be used to make the fault appear light on a

dark background if the light is angled such that it will be reflected

off the fault into the inspector's eye. A shield may be required in

order to prevent the light shining directly into the inspector's eye.

Note that if the general overhead lighting is good then special

local lighting for inspection may not be necessary.@

DESCRIPTION If the fault is shiny and the object is matt and the general

overhead lighting is good then a special light may not be

necessary (R96).

If special lighting is required then the fault can be made to

appear light on a dark background (R93) but a shield may be

required (R94).@

RULE CHECK_SIZES

DESCRIPTION This rule checks the sizes of the object and the fault.@

SUPPLEMENTARY These rules are derived from pilot expert rules 62, 64, 95, 98,

99, 100, 106, 109 and 110.@

WHEN TRUE : ENABLE CALCULATE_VISUAL_ANGLE

WHEN OBJECT_SMALL OR OBJECT_MEDIUM : ADVISE

If the lighting is not suitably located the inspector is likely

to manipulate the object. If this is done under a directional

light source the fault may produce an 'on-off' effect and hence be

more visible.@

DESCRIPTION If the light is not located properly the inspector is

likely to manipulate the object (R100) and this may

produce an 'on-off' effect in the fault if a directional

light is used (R99).@

WHEN OBJECT_VERY_LARGE : ADVISE

Note that it may not be possible to employ a small localised

light source.@

DESCRIPTION If the object is very large a small localised light source

may not be possible (R62).@

WHEN OBJECT_LARGE AND SHINY_OBJECT : ADVISE

A directional light source may lead to problems with glare.@

DESCRIPTION If the object is large and shiny then a directional

light source might cause glare (R109).@

WHEN OBJECT_LARGE : ADVISE

A directional light source is likely to produce uneven illumination

of the object.@

DESCRIPTION If the object is large then a directional light source may

produce uneven illumination (R110).@

WHEN OBJECT_SMALL OR VISANG_MINUTE OR VISANG_VERY_FINE OR VISANG_FINE

OR VISANG_MEDIUM_FINE : ADVISE

Note that a magnification aid might be necessary.@

DESCRIPTION If the object is small or the fault is small then a magnifier

might be needed (R64, R98).@

WHEN VISANG_MINUTE OR VISANG_VERY_FINE OR VISANG_FINE : ADVISE

A high level of illumination may be needed.@

DESCRIPTION If the fault is very small then a high level of illumination

may be necessary (R95).@

WHEN VISANG_TOO_SMALL AND MAX(VIEWING_DISTANCE) <= 30 : ADVISE

The fault is too small. You had better magnify it in some way

or resort to a measuring gauge of some sort.@

DESCRIPTION Nick's rule to handle this ridiculous situation (R106).@

```
  WHEN VISANG_TOO_SMALL AND MAX(VIEWING_DISTANCE) > 30 : ADVISE
  The fault is too small. Try bringing the inspector closer to the
  object or magnifying the fault in some way.@
  DESCRIPTION Nick's rule to handle this ridiculous situation (R98).@

RULE SCRATCH_OR_CHIP
DESCRIPTION If the fault is a scratch or a chip make some inferences.@
SUPPLEMENTARY These rules are derived from pilot expert rules 16, 17 and 18.@
  WHEN TRUE : ENABLE FAULT_IS_MATT
  DESCRIPTION If the fault is a scratch or a chip then it is matt (R18).@
  WHEN TRUE : ADVISE
     The fault will alter the surface texture of the object and therefore
     affect its reflective properties.@
  DESCRIPTION If the fault is a scratch or a chip then it will alter the
        surface texture of the object (R16).
        If the surface texture is altered then the reflective properties
        are altered (R17).@

MODEL FAULT_IS_MATT
DESCRIPTION This model states that the fault is not shiny.@
EQUATION SHINY_FAULT <- FALSE;

RULE COLOUR_FAULT
DESCRIPTION If the fault is a difference in colour make some inferences.@
SUPPLEMENTARY These rules are derived from pilot expert rules 3, 37 and 122.@
  WHEN TRUE : ENABLE COLOUR_RENDERING
  DESCRIPTION If colour is important a colour rendering light source should
        be considered (R3, R37, R122).@

MODEL COLOUR_RENDERING
DESCRIPTION This model restricts the possible outcomes to daylight and
      fluorescent lighting.@
EQUATION OUTCOMES <- {DAYLIGHT, FLUORESCENT};

RULE CHECK_DEPTH_OF_FAULT
DESCRIPTION This rule checks the depth of the dent.@
SUPPLEMENTARY These rules are derived from pilot expert rule 14.@
  WHEN DEEP_DENT : ENABLE SURFACE_UNEVEN
  DESCRIPTION If the dent is deep then it produces surface uneveness (R14).@
```

```
  DEFAULT ENABLE SURFACE_EVEN

  DESCRIPTION If the dent is not deep then it cannot be assumed to produce
        surface uneveness.@

FACTOR DEEP_DENT : LOGICAL

QUESTION Is the dent very deep ?@

DESCRIPTION This factor records whether the dent is deep or not.@

MODEL SURFACE_UNEVEN

DESCRIPTION This model states that the fault makes the surface of the object uneven.@

EQUATION UNEVEN_SURFACE <- TRUE;

MODEL SURFACE_EVEN

DESCRIPTION This model states that the fault does not necessarily lead
      to uneveness in the surface of the object.@

EQUATION UNEVEN_SURFACE <- FALSE;

FACTOR UNEVEN_SURFACE : LOGICAL

DESCRIPTION This factor records whether the fault causes surface
      uneveness or not.@

FACTOR FAULT_MEASURABLE : LOGICAL

REPORTABLE

QUESTION Can the fault be detected with a measuring gauge of some sort ?@

DESCRIPTION This factor records whether the fault can be measured with
      some sort of instrument.@

MODEL ELIMINATE_DAYLIGHT

DESCRIPTION This model eliminates daylight from the possible outcomes.@

EQUATION OUTCOMES <-
      {BLACK_LIGHT, BRIGHTNESS_PATTERNS, CONVERGENT_LIGHT,
      CROSSED_POLARISATION, DARK_FIELD_ILLUMINATION, DIFFUSE_REFLECTION,
      EDGE_LIGHTING, FLUORESCENT, MEASURING_GAUGE, MOIRE_PATTERNS,
      MOVING_IMAGES, POLARISED_LIGHT, SHADOW_GRAPHING, SILHOUETTE,
      SPOTLIGHTING, STROBOSCOPIC_LIGHTING, SURFACE_GRAZING,
      TRANSILLUMINATION};

MODEL ELIMINATE_BLACK_LIGHT

DESCRIPTION This model eliminates black light from the possible outcomes.@

EQUATION OUTCOMES <-
```

{DAYLIGHT, BRIGHTNESS_PATTERNS, CONVERGENT_LIGHT,
CROSSED_POLARISATION, DARK_FIELD_ILLUMINATION, DIFFUSE_REFLECTION,
EDGE_LIGHTING, FLUORESCENT, MEASURING_GAUGE, MOIRE_PATTERNS,
MOVING_IMAGES, POLARISED_LIGHT, SHADOW_GRAPHING, SILHOUETTE,
SPOTLIGHTING, STROBOSCOPIC_LIGHTING, SURFACE_GRAZING,
TRANSILLUMINATION};

MODEL NOT_MOVING_OBJECT

DESCRIPTION This model eliminates stroboscopic light and moving light images from the possible outcomes.@

EQUATION OUTCOMES <-
{DAYLIGHT, BLACK_LIGHT, BRIGHTNESS_PATTERNS, CONVERGENT_LIGHT,
CROSSED_POLARISATION, DARK_FIELD_ILLUMINATION, DIFFUSE_REFLECTION,
EDGE_LIGHTING, FLUORESCENT, MEASURING_GAUGE, MOIRE_PATTERNS,
POLARISED_LIGHT, SHADOW_GRAPHING, SILHOUETTE,
SPOTLIGHTING, SURFACE_GRAZING, TRANSILLUMINATION};

MODEL NOT_TRANSPARENT_OBJECT

DESCRIPTION This model eliminates crossed polarisation, edge lighting and shadow graphing from the possible outcomes.@

EQUATION OUTCOMES <-
{DAYLIGHT, BLACK_LIGHT, BRIGHTNESS_PATTERNS, CONVERGENT_LIGHT,
DARK_FIELD_ILLUMINATION, DIFFUSE_REFLECTION,
FLUORESCENT, MEASURING_GAUGE, MOIRE_PATTERNS,
MOVING_IMAGES, POLARISED_LIGHT, SILHOUETTE,
SPOTLIGHTING, STROBOSCOPIC_LIGHTING, SURFACE_GRAZING,
TRANSILLUMINATION};

MODEL ELIMINATE_TRANSILLUMINATION

DESCRIPTION This model eliminates transillumination from the possible outcomes.@

EQUATION OUTCOMES <-
{DAYLIGHT, BLACK_LIGHT, BRIGHTNESS_PATTERNS, CONVERGENT_LIGHT,
CROSSED_POLARISATION, DARK_FIELD_ILLUMINATION, DIFFUSE_REFLECTION,
EDGE_LIGHTING, FLUORESCENT, MEASURING_GAUGE, MOIRE_PATTERNS,
MOVING_IMAGES, POLARISED_LIGHT, SHADOW_GRAPHING, SILHOUETTE,
SPOTLIGHTING, STROBOSCOPIC_LIGHTING, SURFACE_GRAZING};

```
FACTOR SHINY_OBJECT : LOGICAL

QUESTION Is the object being inspected shiny ?@

DESCRIPTION This factor records whether the object is shiny or not.@

REPORTABLE

FACTOR LIGHT_OBJECT : LOGICAL

QUESTION Is the object being inspected light coloured ?@

DESCRIPTION This factor records whether the object is light coloured or not.@

REPORTABLE

FACTOR TRANSPARENT_OBJECT : LOGICAL

QUESTION Is the object being inspected transparent ?@

DESCRIPTION This factor records whether the object is transparent or not.@

REPORTABLE

FACTOR FLUORESCENT_OBJECT : LOGICAL

QUESTION Is the object being inspected fluorescent ?@

DESCRIPTION This factor records whether the object is fluorescent or not.@

REPORTABLE

FACTOR MOVING_OBJECT : LOGICAL

QUESTION Is the object being inspected in motion ?@

DESCRIPTION This factor records whether the object is moving or not.@

REPORTABLE

FACTOR SHINY_FAULT : LOGICAL

QUESTION Is the fault which is being inspected for shiny ?@

DESCRIPTION This factor records whether the fault is shiny or not.@

REPORTABLE

FACTOR LIGHT_FAULT : LOGICAL

QUESTION Is the fault which is being inspected for light coloured ?@

DESCRIPTION This factor records whether the fault is light coloured or not.@

REPORTABLE

FACTOR TRANSPARENT_FAULT : LOGICAL

QUESTION Is the fault being inspected for transparent ?@

DESCRIPTION This factor records whether the fault is transparent or not.@

REPORTABLE
```

FACTOR FLUORESCENT_FAULT : LOGICAL

QUESTION Is the fault being inspected for fluorescent ?@

DESCRIPTION This factor records whether the fault is fluorescent or not.@

REPORTABLE

FACTOR FAULT_TYPE : SET

REPORTABLE

QUESTION Please select from the following categories the one that best describes

the type of fault being inspected for :

1. A chip in the object;

2. A scratch on the surface of the object;

3. A dent in the object;

4. A bump or lump on the surface of the object;

5. A bend in the object;

6. A colour difference (Eg. Matching materials or dyes);

7. Stresses and strains (Eg. In a piece of glass);

8. None of the above.@

DESCRIPTION This factor records the type of fault being inspected for.@

MAXIMUM 1 FROM {CHIP, SCRATCH, DENT, BUMP, BEND, COLOUR, STRESSES, NONE}

MODEL CALCULATE_VISUAL_ANGLE

DESCRIPTION This model calculates the angle subtended at the inspector's

eye by the fault from the fault size and viewing distance.@

EQUATION VISUAL_ANGLE <- 120*ARCTAN(FAULT_SIZE/(20*VIEWING_DISTANCE));

MODEL IMPOSSIBLE_VISANG

DESCRIPTION This model determines whether the visual angle size is too small.@

EQUATION VISANG_TOO_SMALL <- MIN(VISUAL_ANGLE)<0.83;

MODEL MINUTE_VISANG

DESCRIPTION This model determines whether the visual angle size is minute.@

EQUATION VISANG_MINUTE <- MIN(VISUAL_ANGLE)>=0.83 AND MIN(VISUAL_ANGLE)<1.08;

MODEL VERY_FINE_VISANG

DESCRIPTION This model determines whether the visual angle size is very fine.@

EQUATION VISANG_VERY_FINE <- MIN(VISUAL_ANGLE)>=1.08 AND MIN(VISUAL_ANGLE)<1.42;

MODEL FINE_VISANG

DESCRIPTION This model determines whether the visual angle size is fine.@

```
EQUATION VISANG_FINE <- MIN(VISUAL_ANGLE)>=1.42 AND MIN(VISUAL_ANGLE)<1.83;

MODEL MEDIUM_FINE_VISANG

DESCRIPTION This model determines whether the visual angle size is medium fine.@

EQUATION VISANG_MEDIUM_FINE <- MIN(VISUAL_ANGLE)>=1.83 AND MIN(VISUAL_ANGLE)<2.33;

MODEL MEDIUM_VISANG

DESCRIPTION This model determines whether the visual angle size is medium.@

EQUATION VISANG_MEDIUM <- MIN(VISUAL_ANGLE)>=2.33 AND MIN(VISUAL_ANGLE)<3.00;

MODEL LARGE_VISANG

DESCRIPTION This model determines whether the visual angle size is large.@

EQUATION VISANG_LARGE <- MIN(VISUAL_ANGLE)>=3.00;

FACTOR VISUAL_ANGLE : NUMERICAL

QUESTION What is the smallest angle which the largest dimension of the

     fault could subtend at the inspector's eye ?

     Please give your answer in minutes of arc.@

DESCRIPTION This factor records the visual angle subtended at the eye

     by the fault, in minutes of arc.@

REPORTABLE

BOUNDS 0:7200;

FACTOR FAULT_SIZE : NUMERICAL

QUESTION How small could the largest dimension of the fault be ?

     Please give your answer in millimetres.@

DESCRIPTION This factor records the smallest value which the largest

     dimension of the fault might take, in millimetres.@

REPORTABLE

BOUNDS 0:1000;

FACTOR VIEWING_DISTANCE : NUMERICAL

QUESTION How far from the fault could the inspector's eye be ?

     Please give your answer in centimetres.@

DESCRIPTION This factor records the distance from the fault to the

     inspector's eye, in centimetres.@

REPORTABLE

BOUNDS 0:1000;

FACTOR VISANG_TOO_SMALL : LOGICAL
```

DESCRIPTION This factor records the fact that the size of the visual angle is too small.@

FACTOR VISANG_MINUTE : LOGICAL

DESCRIPTION This factor records the fact that the size of the visual angle is minute.@

FACTOR VISANG_VERY_FINE : LOGICAL

DESCRIPTION This factor records the fact that the size of the visual angle is very fine.@

FACTOR VISANG_FINE : LOGICAL

DESCRIPTION This factor records the fact that the size of the visual angle is fine.@

FACTOR VISANG_MEDIUM_FINE : LOGICAL

DESCRIPTION This factor records the fact that the size of the visual angle is medium fine.@

FACTOR VISANG_MEDIUM : LOGICAL

DESCRIPTION This factor records the fact that the size of the visual angle is medium.@

FACTOR VISANG_LARGE : LOGICAL

DESCRIPTION This factor records the fact that the size of the visual angle is large.@

MODEL SMALL_OBJECT

DESCRIPTION This model determines whether the object is small.@

EQUATION OBJECT_SMALL <- {SMALL} IN OBJECT_SIZE;

MODEL MEDIUM_OBJECT

DESCRIPTION This model determines whether the object is medium sized.@

EQUATION OBJECT_MEDIUM <- {MEDIUM} IN OBJECT_SIZE;

MODEL LARGE_OBJECT

DESCRIPTION This model determines whether the object is large.@

EQUATION OBJECT_LARGE <- {LARGE} IN OBJECT_SIZE;

MODEL VERY_LARGE_OBJECT

DESCRIPTION This model determines whether the object is very large.@

EQUATION OBJECT_VERY_LARGE <- {VERY_LARGE} IN OBJECT_SIZE;

FACTOR OBJECT_SIZE : SET

REPORTABLE

QUESTION What size is the object being inspected ?

Please select one of the following :

1. Small (No larger than a coin);

2. Medium (The size of a book);

3. Large (The size of a desk top);

4. Very Large (Larger than a desk top).@

DESCRIPTION This factor records the size of the object being inspected.@

MAXIMUM 1 FROM {SMALL, MEDIUM, LARGE, VERY_LARGE}

FACTOR OBJECT_SMALL : LOGICAL

DESCRIPTION This factor records the fact that the object is small.@

FACTOR OBJECT_MEDIUM : LOGICAL

DESCRIPTION This factor records the fact that the object is medium sized.@

FACTOR OBJECT_LARGE : LOGICAL

DESCRIPTION This factor records the fact that the object is large.@

FACTOR OBJECT_VERY_LARGE : LOGICAL

DESCRIPTION This factor records the fact that the object is very large.@

FACTOR OUTCOMES : SET

REPORTABLE

DESCRIPTION This factor records the technique or techniques considered

suitable for the inspection task described.@

MAXIMUM 18 FROM {DAYLIGHT, BLACK_LIGHT, BRIGHTNESS_PATTERNS, CONVERGENT_LIGHT,

CROSSED_POLARISATION, DARK_FIELD_ILLUMINATION, DIFFUSE_REFLECTION,

EDGE_LIGHTING, FLUORESCENT, MEASURING_GAUGE, MOIRE_PATTERNS,

MOVING_IMAGES, POLARISED_LIGHT, SHADOW_GRAPHING, SILHOUETTE,

SPOTLIGHTING, STROBOSCOPIC_LIGHTING, SURFACE_GRAZING,

TRANSILLUMINATION}

RULE RECOMMENDATIONS

DESCRIPTION Advise the user of the best techniques to employ in the

inspection task.@

SUPPLEMENTARY These rules and descriptions are derived from Richard

Schweickert and pilot expert rules 26, 29 and 31.@

WHEN TRUE : ENABLE EXPERTS_OUTCOMES

WHEN {DAYLIGHT} IN OUTCOMES : ADVISE

The more like normal daylight you can make the illumination the

better. Daylight itself is not a good idea because it is too variable.

A special colour rendering light source might be considered but this

will be expsensive and require high levels of illumination.

A fluorescent light with a diffuser may therefore be preferable.@

DESCRIPTION Special colour rendering light sources are expensive (R29).

They also require high levels of illumination (R31).

Fluorescent light can be used (R37) but a diffuser may be

required to reduce glare (R26).@

WHEN {MEASURING_GAUGE} IN OUTCOMES : ADVISE

Since the fault can be measured the only problems are those of

selecting an appropriate instrument and ensuring that the overall

illumination is sufficient to read it.@

WHEN {SURFACE_GRAZING} IN OUTCOMES :

ADVISE Consider surface grazing. This uses a collimated source of light with an oval beam directed at the surface at a very low angle.

Small bumps are revealed as high contrast shadows or highlights.@

WHEN {POLARISED_LIGHT} IN OUTCOMES :

ADVISE Consider using polarised light at a near zero angle to the object.

Nonspecular defects in a specular surface will appear as bright spots in a dark surround. @

WHEN {SPOTLIGHTING} IN OUTCOMES :

ADVISE Consider spotlighting at a low angle.

Bumps or dents in a surface will appear as points of light.

Scratches in a surface will appear as lines of light.

Caution: Spotlighting may produce glare.@

WHEN {BRIGHTNESS_PATTERNS} IN OUTCOMES :

ADVISE Consider using a brightness pattern. This is a high contrast image, such as stripes or a grid, projected onto or through the surface. If the object is transparent, consider putting a black surface underneath it to reflect the pattern back through it.

Broad waves, hills and valleys in a specular or transparent surface will appear as distortions in the pattern.

Caution: The inspector may develop peculiar after images when he looks away from the pattern, especially if narrow, high contrast lines are used.@

WHEN {STROBOSCOPIC_LIGHTING} IN OUTCOMES :

ADVISE Consider stroboscopic light synchronised to light the object only at those times when a defect might occur.

Caution: Vertigo and hypnotic effects may occur with stroboscopic light. @

WHEN {MOVING_IMAGES} IN OUTCOMES :

ADVISE Consider projecting a moving light pattern in synchrony with the moving surface to provide a fixation point for the eye. @

WHEN {MOIRE_PATTERNS} IN OUTCOMES :

ADVISE To help find bumps, nicks, or dents on a light coloured surface, consider using a moire pattern. That is, two superimposed moving patterns of lines. The lines produce an interference pattern.

To use a moire pattern for this application, a collimated beam is passed through parallel lines a short distance away from the light coloured surface. The inspector observes the pattern produced by the original lines and their shadow on the object.

Defects appear as irregularities in the pattern. @

WHEN {DARK_FIELD_ILLUMINATION} IN OUTCOMES :

ADVISE Consider dark-field illumination. This is light reflected off a surface or projected through a transparent product and focused near to, but not on, the eye.

Scratches will cause the light to diffract to the side and strike the eye. @

WHEN {CONVERGENT_LIGHT} IN OUTCOMES :

ADVISE To find sheen in a surface or transparent object which is specular, consider convergent light, that is, light reflected from or projected through the object and focused on the eye.

Changes in sheen will appear as light or dark areas.

Caution: Surface depth illusions may occur. @

WHEN {CROSSED_POLARISATION} IN OUTCOMES :

ADVISE Consider using a crossed polariser. That is, two sheets of linear polariser at 90 degrees to each other, one on either side of the object.

Internal stress in a transparent object will appear as a change in pattern or colour.@

WHEN {DIFFUSE_REFLECTION} IN OUTCOMES :

ADVISE Consider diffuse reflection. That is, reflecting a white large area diffuse source of light off the surface of the object.

Changes in thickness in an object made of flat transparent layers will appear as a rainbow.@

WHEN {EDGE_LIGHTING} IN OUTCOMES :

ADVISE Consider edge lighting. That is, directing an intense light through the edge of the transparent object.

Internal imperfections will appear bright against a dark background. Scratches will appear as bright lines.@

WHEN {BLACK_LIGHT} IN OUTCOMES :

ADVISE Consider using ultra-violet light to find fluorescence. Caution: Ultra-violet light can produce retinal burns.@

WHEN {TRANSILLUMINATION} IN OUTCOMES :

ADVISE Consider transillumination. That is, directing light from a large diffuse source through the object. The object is between the light source and the eye.

Changes in opacity will appear as light or dark streaks or spots.@

WHEN {SHADOW_GRAPHING} IN OUTCOMES :

ADVISE Consider shadow graphing. That is, projecting a small source of light through the transparent object.

Changes in refraction will appear as light or dark spots in the projected image of the object.@

MODEL EXPERTS_OUTCOMES

DESCRIPTION This model determines the outcomes actually identified by the expert.@

EQUATION OUTCOMES <-

{HALOGEN,FLUORESCENT,DAYLIGHT,INCANDESCENT,TRANSILLUMINATION,
SILHOUETTE,DIFFUSE_REFLECTION,SPOTLIGHTING,BLACK_LIGHT,
MEASURING_GAUGE,STROBOSCOPIC_LIGHTING};

RULE GLARE_WARNING

DESCRIPTION This rule informs the user of the best object/background brightness ratio to avoid glare and maximise contrast.@

SUPPLEMENTARY This information comes from pilot expert rules 30, 35, 38, 53, 55 and 56.@

WHEN TRUE : ADVISE

A note about contrast and glare.

In order to maximise the contrast between the centre of interest and

the background whilst ensuring that glare does not become a problem

the centre of interest should ideally be three times brighter than the

background or vice versa.

Diffuse light sources or diffusing screens can be used to reduce glare

and unwanted surface reflections.@

DESCRIPTION The ratio of centre of interest to background brightness should

ideally be three to one or one to three (R53, R55, R56).

Diffuse light sources or diffusing screens can be used to reduce

glare and unwanted reflections (R30, R35, R38).@

COMMENT MEASURING AND LIGHTING ARE COMPLEMENTARY OPTIONS @

EXCLUSIVE MEASURE_OUTCOME,LIGHTING_OUTCOME@

COMMENT EVEN AND UNEVEN SURFACE ARE COMPLEMENTARY OPTIONS @

EXCLUSIVE SURFACE_UNEVEN,SURFACE_EVEN@

COMMENT DAYLIGHT IS A COLOUR RENDERING LIGHT SOURCE @

EXCLUSIVE COLOUR_RENDERING,ELIMINATE_DAYLIGHT@

COMMENT IF THE OBJECT AND FAULT HAVE THE SAME CHARACTERISTICS THEN

EACH OBJECT CHARACTERISTIC MUST BE INHERITED BY THE FAULT @

CONCURRENT SAME_CHARACTERISTICS,SAME_SHINY,SAME_LIGHT,SAME_TRANSPARENT,

SAME_FLUORESCENT@

COMMENT EITHER ALL OR NONE OF THE MODELS DEALING WITH SIZE ARE NEEDED @

CONCURRENT CALCULATE_VISUAL_ANGLE,MINUTE_VISANG,VERY_FINE_VISANG,FINE_VISANG,

MEDIUM_FINE_VISANG,MEDIUM_VISANG,LARGE_VISANG,IMPOSSIBLE_VISANG,

SMALL_OBJECT,MEDIUM_OBJECT,LARGE_OBJECT,VERY_LARGE_OBJECT@