A Methodology For Knowledge Acquisition In The Development Of Expert Systems

Thesis submitted in accordance with the requirements of the University of Liverpool for the degree of Doctor of Philosophy by Robert Thomas Plant.

September 1987
To my Mother and Father.
Acknowledgements

I am deeply indebted to my supervisors Prof. Mike Shave and Stuart Moralee for their guidance and constant encouragement during my research.

I would also like to thank Prof. John Thomes & Anne Marchington of Bristol University, Nigel Gardner of CTISS and John Barber of Unilever Research for their assistance, patience and especially their knowledge in the soil erosion case study.

Thanks go to Dr. Paul Dunne for enduring my troff questions also Kenny C. and Dave S. for helping me overcome the frequent late night system crashes.

I am grateful to the Science and Engineering Research Council of Great Britain for their financial support.
Abstract

This thesis presents a methodology for the development of the knowledge based component of an expert system. The methodology takes as its starting point in the development an initial specification that gives both a domain definition and a problem description, this is in the form of natural language. The initial specification is then taken and used in the selection of a knowledge elicitation technique which the knowledge engineer employs to extract knowledge from the domain expert. The methodology works under the constraint that this elicitation technique produces a textual representation of the expert's knowledge. This can then act as the basis of a series of representation refinement and analysis processes. These processes have two aims, the first is to produce a formal specification of the elicited knowledge which can be used as the unambiguous definition of the domain through the life of the knowledge base. The second aim is for the analysis to indicate which classical representation would be suitable to represent the elicited knowledge. Once these two goals have been fulfilled the elicited knowledge held in the formal specification can be represented in the form advocated by the representation specification. This together with a suggested control architecture is known as the concrete specification from which a knowledge base can be implemented.

The thesis does not address the wider issues of expert system construction such as the production of the domain definition or problem description. Also aspects such as the man-machine interface needs of expert systems are not considered.

The thesis demonstrates the applicability of the approach by giving details of two case studies.
CONTENTS

Chapter 1. Introduction.
  1.1 Aims and perspective. 1
  1.2 Structure of thesis. 3

Chapter 2. An overview of expert systems.
  2.1 History of expert systems. 6
  2.2 Development methodologies for expert systems.
     2.2.1 Introduction.11
     2.2.2 Existing methodologies for expert systems construction. 12
  2.3 Considerations in developing a new design methodology for expert systems. 14

Chapter 3. The knowledge elicitation phase.
  3.1 Overview of the design methodology.
     3.1.1 Introduction. 17
     3.1.2 Development outline. 17
     3.1.3 Conclusions. 20
  3.2 Initial specification.
     3.2.1 Introduction. 22
     3.2.2 Comments on the initial specification stage. 24
  3.3 Knowledge elicitation.
     3.3.1 Aim, role and perspective. 24
     3.3.2 Knowledge elicitation techniques. 25
     3.3.3 Verbal transfer of knowledge. 26
     3.3.4 Knowledge engineer investigates literature information. 35
     3.3.5 Expert uses intermediary program. 36
     3.3.6 Machine learning. 37
     3.3.7 Other elicitation techniques. 38
     3.3.8 A summary of knowledge elicitation techniques. 39
  3.4 The elicited representation.
     3.4.1 Introduction. 41
3.4.2 Aims, role and perspective.  
3.4.3 Elicited techniques for the methodology.  
3.4.4 Example elicited representations.  
3.5 Comments and conclusions.  

Chapter 4. The knowledge acquisition phase.  
4.1. Primary analysis.  
4.1.1 Aim, role and perspective.  
4.1.2 Primary representations.  
4.1.3 The selection process.  
4.1.4 Flow diagrams.  
4.1.5 Contour diagrams.  
4.1.6 Decision tables.  
4.1.7 The trace matching process.  
4.1.8 Primary analysis in practice.  
4.2. The Transformational process.  
4.2.1 Introduction.  
4.2.2 Transformation technique one.  
4.2.3 Conversational coherence.  
4.2.4 Comment on conversational coherence.  
4.2.5 Transformation technique two.  
4.2.6 Transformation technique three.  
4.3. Adequacy of a representation.  
4.4. Comments and conclusions.  

Chapter 5. Transformation of the primary representation.  
5.1. Introduction and overview.  
5.2. The Domain specification.  
5.2.1 Aim, role and perspective.  
5.2.2 A review of specification languages.
6.11. Towards the representation specification.
   6.11.1. Profile analysis. 145
   6.11.2. Profile matching. 147
6.12. The representation specification. 149
   6.13.1. The secondary representation. 149
   6.13.2. The control architecture. 153
   6.14.1 Introduction. 154
   6.14.2 Concrete specification. 154
   6.14.3 Implementation. 154
6.15. The implementation. 156
6.16. Comments on the development. 157
6.17. Further investigation into primary analysis.
   6.17.1. Introduction. 158
   6.17.2. Profile selection. 158
   6.17.3. The representation specification. 159
   6.17.4. The concrete specification. 159
   6.17.5. Analysis of a rule based representation. 167
   6.17.6. Analysis of the investigation. 169
6.18. Comments and conclusions. 171

Chapters
7. A large scale case study in expert system development.
   7.1 Introduction. 174
   7.2 Background to case study. 174
   7.3 The initial specification. 175
   7.4 The knowledge elicitation phase. 176
   7.5 The primary analysis phase. 183
   7.6 The secondary analysis stage. 197
   7.7 Conclusions. 205
CHAPTER 1.

INTRODUCTION.

1.1 Aims and perspective.

The aim of this thesis is to provide the first steps towards a rigorous development methodology for expert systems. In doing so we will adopt the following three principles: Firstly, that every step in the development from the definition of the problem domain to the implementation should be capable of justification. Secondly, that this development path has a point at which the knowledge of the domain, elicited from the expert, is represented in an implementation-independent form. Thirdly, that a suitable representation is chosen for the domain and that this choice is capable of justification. The intention is to produce a specification written in a form which has a high degree of rigour from which a system can be implemented.

The creation of expert systems needs a formal or rigorous development philosophy in order to ensure that the system matches the specification and intention of the developers. This is missing in the majority of existing methodologies, many of which have evolved over the past twenty years. During this initial period several knowledge based systems have been produced which are capable of expert levels of performance [Yu, 1979a] or are actively used for giving advice in industrial settings [Kraft, 1984]. However, both the early design methodologies and many of the resultant systems can be criticised in several ways.

The first and most significant criticism is that many existing methodologies do not have a development functionality - i.e. if the process of creating an expert system is performed more than once from a fixed problem definition and set of information, then the development paths would most likely not be identical nor would the resultant systems. This is due to a lack of guidelines, structure and formality in these methodologies.

Secondly, an evolutionary approach to designing systems has sometimes been adopted. In this approach the knowledge engineer examines how an earlier successful system was developed, taking any underlying principles gained from this post-implementation analysis and turning them into the guidelines of a methodology. A flaw in this approach is that many of
the systems upon which the analyses are based were not originally developed in a conscious and planned manner, but utilised the most direct and convenient methods to produce a working system, with little regard for later consequences.

The third criticism of expert systems is the significant amount of time, resources and effort needed in order to build an acceptable system - acceptable in the sense that it works on a non-trivial domain, on a non-trivial set of problems within that domain and can produce an acceptably high standard of results with acceptable resource utilisation.

A fourth criticism relates to the lack of a theoretical basis upon which expert systems can be developed, a fact acknowledged by Bramer:

"Although the expert systems 'industry' has increased enormously in the last five years, as has the field's commercial recognition, it is important to realise that the theoretical basis of the subject has advanced very little. In all the euphoria, difficult theoretical and methodological issues have been forgotten or ignored but not solved" [Bramer. 1986].

Thus, there are many areas within the expert systems field that can be improved. The status of expert systems can at present be likened to that of commercial data processing in the early 60's, when applications were produced but with little theoretical understanding to explain the production processes.

The expert systems area is in need of a change similar to that which commercial data processing underwent with the advent of the more structured design methodologies such as JSD [Jackson. 1975] which turned programming from a secret art into a software science and programmers into engineering professionals [Hoare. 1982].

In this thesis we present a methodology that focuses upon one aspect of the KBS design process, that of taking a domain definition in conjunction with a problem description and by following a series of guidelines produce a specification from which the problem solving component of an expert system can be constructed.

The methodology takes as its starting point a domain definition and a problem description - the initial specification. These can be in any form or style that the knowledge engineer feels is suitable - generally natural language. We have not attempted in this work to investigate or show how the initial specification is derived. The methodology then works from this definition to produce a specification of a knowledge base that once implemented could be used to
solve the problems defined in the initial specification. The methodology however, does not investigate or detail other aspects needed for the development of complete expert systems, many of which utilise the knowledge base. For example the development of a man-machine interface to the knowledge base is not examined.

Having obtained the initial specification the next step is to use this in order to obtain the raw knowledge from which the knowledge base will be ultimately composed. The raw knowledge is obtained from a domain expert by the knowledge engineer using one of a series of techniques and this is known as knowledge elicitation. Several elicitation techniques are available and some are described in chapter three. The methodology described in this thesis does however place a constraint on the type of knowledge elicitation that the knowledge engineer can use, this is that the elicitation technique must result in non diagrammatic textual output. This has been done in order to achieve a solid well defined form to which analytical techniques can be applied. The textual representation then undergoes a refinement process where the organisational structure of the knowledge is highlighted. This new representational form then undergoes two processes. The first re-represents the knowledge into a formal notation, the second selects a suitable classical representation in which to represent that knowledge (i.e., frames, rules etc.) The thesis describes three representations that it uses to illustrate the selection process. These are standardised representations typical of their type, well defined yet not too specific, they are not intended to be the only representations available and the knowledge engineer is encouraged to develop other more sophisticated representations as appropriate. The formal domain specification and the representation specification are then brought together so that the knowledge is held in the form advocated by the representation specification. This plus information on what is a suitable control architecture for the knowledge base then acts as a specification from which a knowledge base could be implemented. However, the accompanying processes that turn the knowledge base into a full system are not specified i.e., interfaces, user profile etc., as this is beyond the scope of this thesis.

1.2 Structure of thesis.

In chapter two we will review the existing design methodologies, discussing their
strengths, weaknesses and applicability to the rigorous development of an expert system. The literature documenting these approaches is however weak and contains relatively little of the detail necessary in order to apply these successfully to a real problem domain, tending instead to either discuss in general or abstract terms the development of a hypothetical system, or concentrate upon one aspect of the development process rather than detailing the whole approach from problem definition through to implementation. A further problem is that many of methodologies given in the literature are based upon the structured, yet unrigorous commercial data processing methodologies which are generally unsuitable for expert systems. We suggest the need for a rigorous approach in order to control the complexity of a system and prevent a breakdown in understanding when changes in the knowledge base are necessary.

An overview of our methodology is presented in Chapter three along with a discussion of the elicitation techniques available to the knowledge engineer.

In Chapter four we consider how the output of an elicitation session can be transformed into a (primary) representation that highlights its organisational structure. We also discuss a series of measures collectively known as ‘adequacy’ that can be used to gauge the strengths and weaknesses of a methodology or representation.

In Chapter five we discuss how this (primary) representation can be analysed: firstly, in order to produce a formal specification of the domain knowledge and secondly to select a representational form (e.g., frames, rules, semantic network etc.) that would be suitable for the domain under investigation. When the representational form has been selected, its definition is also given as a specification. Chapter five then details how the formally specified knowledge is transformed into the constructs defined in the representation specification. The selection of a control architecture is also discussed.

In Chapter six we show through the use of a small case study how our development methodology can be applied in practice from initial specification to implementation.

Chapter seven then presents a large case study on the domain of soil erosion where we interviewed domain experts, analysed several thousand lines of transcript and developed a series of specifications from which the first stages of a system could be implemented. The aim
of the chapter is to show how the methodology works under the constraints of a real world domain.

Finally, Chapter eight provides some concluding remarks about the project and suggests some improvements and possibilities for future work.
CHAPTER 2.

AN OVERVIEW OF EXPERT SYSTEMS.

2.1 History of expert systems.

The first period of expert systems research was dominated by a belief that a few laws of reasoning coupled with powerful computers would produce expert and even superhuman performance, an approach known as General Purpose Problem Solving (GPPS). The first attempt to formalise this through the construction of a computer system was made by Newell, Shaw and Simon who started work on their General Problem Solver in 1957 [Newell, 1961]. The GPS was the first problem solving program to separate its problem solving methods from the knowledge specific to the problem domain. This step of separating the problem solving part of the system which gave no information about the kind of data worked on, and the task specific knowledge was a very significant one in the history of expert systems as we shall subsequently see. GPS was important for two points: Firstly, it demonstrated the possibility of separating knowledge from the inference mechanism and secondly it showed that a system containing only a few general rules will be too weak to solve any significant problems in real world domains, let alone problems over a universal set of domains. An interesting discussion and criticism of the GPS and the philosophy associated with it is given by Drayfus [Drayfus, 1965]

By the mid 1970's several expert system projects were beginning to show some worthwhile commercial results and the field itself was beginning to have more definition with research being focused in several areas considered crucial to a better understanding of expert or 'knowledge based' systems. For example the GPS work showed that research into inference mechanisms and knowledge representations was vital, while other early systems showed that systems that acted as 'black boxes' were distrusted and disliked by users and experts, highlighting the need for systems to explain and justify their reasoning to users.

Expert systems research, aimed to expand on the limited knowledge available on these topics, had by the late 1970's produced many working and useful systems, which could be classified as first generation systems. These systems were intended dually as research vehicles and as systems that could perform at an acceptable level of performance over a given, if
limited domain. In the following section we discuss some of the outstanding first generation systems.

The DENDRAL project is concerned with developing a system to analyse the spectrum produced in mass spectrographic, nuclear magnetic resonance and with other data infer what are the constituent atoms in an unknown molecule.

The objective of the system is to illustrate how a knowledge based (or 'heuristic' as the DENDRAL workers describe it) approach compares to that of an algorithmic approach to determining the constituent molecules. This comparison is possible because the original DENDRAL 'algorithm' was produced by Joshua Lederberg and implemented using an exhaustive search approach to deduce the desired solution [Lederberg, 1964]. The research which started in 1968 has shown two results: Firstly, that the knowledge based approach is far better than the algorithmic one in finding a solution and in so doing avoiding the use of an enormous search space as needed in the non heuristic algorithmic approach. Secondly the results that HeuristicDENDRAL produces are equal to or better than the performance of experts in analysing some aspects of spectrographic data [Smith, 1972] [Smith 1975], hence demonstrating that an expert level of performance is possible from a computer system. HeuristicDENDRAL produces these results even though it has less knowledge in total than an expert, by performing a systematic search of the space of possible molecular structures (like the algorithmic DENDRAL) in conjunction with its knowledge and heuristic information about the domain to limit the search space (unlike the algorithmic DENDRAL - but like the domain expert). Lindsay provides a thorough treatment of DENDRAL [Lindsay, 1980] whilst Buchanan et. al., provide [Buchanan et al. 1969] [Buchanan et al. 1970] a discussion of the system and project.

The MYCIN expert system is probably the most widely discussed and well known expert system built so far [Shortliffe, 1976] and was designed to provide advice on diagnosis and therapy for infectious diseases.

The system pioneered the use of production systems as an approach to problem solving by tackling several difficult problem areas that expert system developers - "knowledge engineers", face when building some systems. These problems can be split into three areas.
Firstly, there are the representational difficulties: i) Dealing with probabilistic evidence for use in detailing an action - known as inexact values - within a system. ii) Representing large volumes of complex knowledge in rule form. Secondly, there are the problems of inference: i) Reasoning effectively over a large rule base. ii) Reasoning with weighted evidence. Thirdly, human/machine communication problems: i) The problem of acquiring and using new knowledge within an existing knowledge base. ii) Dealing with the many social and psychological problems of the human-computer interaction of a system intended for a real world problem domain.

The results of MYCIN’s solutions to these problems have been very extensively documented in the literature and we shall therefore present only a brief synopsis of each of them here. The MYCIN team approached inexact reasoning by introducing the concept of ‘certainty factors’ where a condition and an action have values associated with them on a scale of -1.0 to 1.0, where 1.0 corresponds to complete confidence that a given action always follows from a condition and -1.0 that it never does. These confidence factors were used in preference to the more conventional statistical approaches e.g. Bayes theorem as:

"experience with clinicians has shown that clinicians do not use the information comparable to implemented standard statistical models" [Shortliffe. 1976] and that

"The concept of Certainty Factors did appear to fit the clinicians reasoning patterns - Their judgement of how they weighed factors, strong or weak in decision making" [Shortliffe. 1976].

This representation although not finding total approval within the Artificial Intelligence (A.I) community did offer a new approach and stimulated much research in the area of inexact reasoning.

The problem of representing large volumes of knowledge produced a representation that had strictly defined rules and highlighted the fact that research into consistency, completeness and accuracy of knowledge within a representation is necessary.

The inference mechanism that was produced to manipulate the representation used a simple backward chaining approach resulting in an exhaustive depth first search of an AND/OR tree. However, the system did introduce several new techniques such as: i) Increased utilisation of interactive dialogues between the system and user. For example, this technique was called
upon when the system felt a given subgoal was valuable in the solution process but where
without further external information the subgoal could not be pursued further. ii) In order to
increase the searching efficiency before the entire list of rules for a subgoal is retrieved, the
program attempts to find a sequence of rules that would establish the goal with certainty (C.F = 1.0). This is known as a 'unity path'. This also allows common sense deductions to be made
with minimal effort. iii) The inference mechanism performs partial evaluation of rule premises.
Since many attributes occur in several rules the value of one clause may have already been
established whilst the others in the rule are unknown clauses being expanded and the rule is
not further evaluated.

The problems associated with acquiring new knowledge were investigated through the
TEIRESIAS system [Davis. 1982] an interactive system for knowledge elicitation. The system
is also capable of evaluating MYCIN’s performance and assigning credit or blame to individual
rules. The TEIRESIAS project was valuable in that it was one of the first knowledge elicitation
systems and that it heightened the need to investigate the problems associated with explanation
facilities, the use of meta knowledge, human computer interaction and domain independent
elicitation mechanisms.

Even though the deductive performance of MYCIN compared favourably with experts in
diagnosing infectious diseases and therapy selection for patients with bacteremia and meningitis
[Yu. 1979a] [Yu. 1979b], the real value of the MYCIN project was as a research vehicle that:
i) Highlighted some of the problem areas in knowledge based systems ii) Showed some new
and novel approaches to solving several problems and iii) Heightened the awareness of computer scientists and others of the expert systems area by tackling a problem from a real world
domain.

The two innovative systems described above along with others such as: INTERNIST
[Pope. 1977], PROSPECTOR [Duda. 1978], CRYsalis [Englemore. 1979], GUIDON
[Clancy. 1979] caused a significant increase in the interest of the commercial world in expert
systems. However it was not until the early 1980's that real commercial utilisation of expert
systems occurred, this being assisted by two events.
The first event was the success of a system known as XCON * produced in a collaborative effort between Digital Equipment Corporation and Carnegie Mellon University. The aim of the system is to configure VAX 11/780 computers - specifying the cables, drives and components that are needed for a given machine, determining where they will be located in the machine, specifying restrictions on the machine such as the maximum distance from disk drives to processor etc. The construction of this system was the first large scale commercial system involving large scale elicitation and resulted in over 2000 rules. The XCON project was commenced in 1978 and by 1980 DEC was using it to configure all VAX orders, by 1984 it had analysed nearly 20000 orders [Kraft, 1984] and was running at 95 to 98 percent accuracy. Kraft also states:

"XCON has become an indispensable and effective business tool" [Kraft. 1984] thus indicating to the industrial and commercial world that expert systems were viable and may if used in conjunction with conventional information systems allow new classes of problems to be tackled.

It should be noted however that the project raised serious discussions of the socio-economic impact of such systems.

The second significant event in expanding the commercial acceptance of expert systems was a result of the GPS research work of the early 1960's - the concept of separating the knowledge base from the inference mechanism. This was done by the EMYCIN system [van-Melle. 1979] - Empty MYCIN, where the inference engine was detached from the MYCIN knowledge base allowing other domains to be plugged into the inference mechanism, as long as they were represented in the rule language. This approach has become known as using an expert system "shell". The idea of an easier way to enter the expert system field than that of writing systems from scratch was very appealing to industry and has lead to the development and use of a wide variety of shells.

A further discussion of the historical development of KBS is given in [Weiss. 1983] whilst [Feigenbaum. 1963] describes the early work, gives some of the classic papers and an

*) XCON is also known as R1. John McDermott from Carnegie-Mellon University, who developed the original program named the system R1, reportedly saying "Three years ago I wanted to be a knowledge engineer, and today I am one".
extensive bibliography of the early literature. Buchanan [Buchanan, 1986] gives an extensive
discussion and details of working systems and the research literature.

2.2 Development Methodologies for expert systems.

2.2.1 Introduction.

As we discussed in the previous section the interest in using expert systems has risen
rapidly since the late 1970's, fuelled by the success of systems such as XCON and the availa­
bility of shells.

It was thought that shells would provide an easy way to produce expert systems which
would perform at a high level of competence once the relevant knowledge base had been built
up and plugged in. However, as experience with them grew it became clear that this was not
going to be true for the majority of cases. The reason for this apparent lack of success was that
the representation formalisms used by the shell suffered from the dilemma that in order to
increase the variety of domains to which the shell could be applied they had to be made more
general, this however weakened the power of inference that could be drawn from them. The
alternative to this is to make the representation powerful and consequently more specialised,
but this can cause domains to be unnaturally represented and may prevent systems from
achieving their full potential. The shell based approach has consequently undergone change and
resulted in two schools of thought. One philosophy towards shells is to use them only as a
means of becoming initially familiar with the expert systems world. This has resulted in may
inexpensive packages being produced and used by industry to test out some small systems. The
second school of thought is where the shell approach has been taken to the extreme and turned
into a development environment e.g., IntelliCorp's, KEE system. Both of these schools have
their advantages and disadvantages.

Small shells provide an introduction to the expert systems approach to problem solving
without large commitment of resources, however this can led to a distorted view of the field.
Many shells have poor representational capabilities, with the majority of shells still being based
upon EMYCIN or a PROSPECTOR based architecture that utilises rules. This had lead to the
unfortunate term "Rule based systems" often being used generically for expert systems. The
limitations that simple shells impose on the user can either leave the user disillusioned with expert systems if they do not realise that the limitations are enforced purely through the shell or leave the user feeling frustrated with the limited size and scope of system that the shell allows them to construct, especially if the representation is not truly suited to the domain.

The large environments such as KEE are highly sophisticated packages that allow for vast scope and flexibility in approaching a problem. However they also suffer from the problem of how a knowledge engineer undertakes the task of constructing an expert system with them. This problem, as we stated in chapter one, is similar to that faced by the commercial data processing community in the late 1960’s and early 1970’s when new time share systems were introduced enabling much larger systems to be built than before. As a consequence, new problems appeared: ensuring that the system would do what was required, that the complexity of the system would be maintainable etc. It follows therefore that what was required by knowledge engineers was a design methodology for the construction of expert systems that would allow them the same power of design that methods such as JSD [Jackson. 1975] or Yourdine & Constantine [Constantine. 1974] gave to the developers of commercial data processing systems.

2.2.2. Existing methodologies for expert systems construction.

As we discussed in the last section the introduction of shells led to a dramatic rise in the number of users of expert systems, partly because shells overcome some the difficulties encountered in developing custom built systems for problems in real domains. Potential system designers find difficulty in building such knowledge based systems, as their structures are unlike any found in conventional software systems. Consequently, the design methodologies such as JSD, used to build conventional algorithmic commercial or scientific software are not applicable to the design of expert systems. Thus a different approach within the design methodology is necessary.

The infeasibility of using conventional design methods causes the potential developers to turn to the a.i literature in search of a methodology for developing knowledge based systems. However, the literature is very weak in this respect and as we shall see falls short of giving an
adequate development methodology for use by the knowledge engineer.

Buchanan's Methodology.

A well documented methodology for constructing expert systems is that proposed by Buchanan and his colleagues [Buchanan, 1983] and is characterised by the following five stage model:

1. Identify problem characteristics.
2. Find concepts to represent knowledge.
3. Design structure to organise knowledge.
4. Formulate rules to embody knowledge.
5. Validate rules that organise knowledge.

- requirements.
- concepts.
- structure.
- rules.
- refinements.
- redesings.
- reformulations.

Buchanan's approach to developing expert systems examines far wider aspects of the development process than this thesis addresses but at a very much higher level. In relation to the research presented here Buchanan fails to give enough attention to: i) How the knowledge is elicited, ii) What techniques are necessary to analyse the elicited knowledge, iii) How that knowledge is to be represented in a suitable form, iv) How the knowledge will be prevented from change in semantic meaning over the stages and v) The methodology fails to differentiate between knowledge elicitation and knowledge acquisition, describing the whole process of developing the system as knowledge acquisition. However, he does examine the role of the user in developing the system requirements and considers factors such as user front ends, explanations and user needs (in a descriptive, general, high level style), these are areas we do
not consider.

Other methodologies.

The literature describing the approaches to constructing KBS is weak and apart from Buchanan's methodology only a few others are available. We will now briefly discuss some of these and their relationship to the work presented here.

Grover presents an interesting 'Pragmatic knowledge acquisition methodology' [Grover, 1983] which suggests a three stage model that aims at producing a 'knowledge acquisition document series'. This is a useful concept and emphasises the fact that documentation is essential for understanding a knowledge base as well as for its future maintainence. Grover describes at a high level an approach to producing the problem description and domain description, an area this thesis does not address. However Grover does not discuss many important points such as elicitation, representational problems, semantic consistency constraints or the problems of formality.

Alexander proposes a methodology known as Ontological Analysis [Alexander, 1986] that is intended to provide analysis at what Newell terms the 'knowledge level' [Newell, 1982]. Alexander's paper considers the problems associated with producing a formal specification of the knowledge through static, dynamic and epistemic analysis of the domain knowledge. He concurs with the research presented here for the need of a formal approach to specifying the knowledge base, however he does not expand on any other aspect of the KBS construction process.

2.3 Considerations in developing a new design methodology for expert systems.

The previous section surveyed some of the more influential design methodologies for constructing KBS that are documented in the literature. They all make a contribution to the field but individually none of them offers a truly adequate methodology for the construction of an expert system. One of the common difficulties is that they fail to differentiate between the knowledge elicitation phase and the knowledge acquisition phase in the life cycle, often combining them into an iterative "acquire-test-edit" process. This 'prototyping' approach can cause problems in the consistency, completeness and emphasis within the knowledge base. It is better
consider.

Other methodologies.

The literature describing the approaches to constructing KBS is weak and apart from Buchanan's methodology only a few others are available. We will now briefly discuss some of these and there relationship to the work presented here.

Grover presents an interesting 'Pragmatic knowledge acquisition methodology' [Grover, 1983] which suggests a three stage model that aims at producing a 'knowledge acquisition document series'. This is a useful concept and emphasises the fact that documentation is essential for understanding a knowledge base as well as for its future maintenance. Grover describes at a high level an approach to producing the problem description and domain description, an area this thesis does not address. However Grover does not discuss many important points such as elicitation, representational problems, semantic consistency constraints or the problems of formality.

Alexander proposes a methodology known as Ontological Analysis [Alexander, 1986] that is intended to provide analysis at what Newell terms the 'knowledge level' [Newell, 1982]. Alexander's paper considers the problems associated with producing a formal specification of the knowledge through static, dynamic and epistemic analysis of the domain knowledge. He concurs with the research presented here for the need of a formal approach to specifying the knowledge base, however he does not expand on any other aspect of the KBS construction process.

2.3 Considerations in developing a new design methodology for expert systems.

The previous section surveyed some of the more influential design methodologies for constructing KBS that are documented in the literature. They all make a contribution to the field but individually none of them offers a truly adequate methodology for the construction of an expert system. One of the common difficulties is that they fail to differentiate between the knowledge elicitation phase and the knowledge acquisition phase in the life cycle, often combining them into an iterative "acquire-test-edit" process. This 'prototyping' approach can cause problems in the consistency, completeness and emphasis within the knowledge base. It is better
to separate the elicitation of the knowledge from the analysis of that knowledge.

Further, as different elicitation techniques require different types of analysis, a methodology should state which techniques can be utilised within it.

A second area in which the existing design methodologies are weak is that of suggesting and selecting a suitable knowledge representation. A sound selection process is vital if the system is going to be capable of reaching its potential. We therefore suggest that an approach that first analyses the representational needs of the domain prior to selecting a representation to fulfil these needs would be advantageous.

The selection of a suitable representation could be complemented by reasoning about the domain knowledge in a form other than in the selected representation language. For example, the formal approach taken by Alexander could be adopted. It would be possible for the formally represented domain knowledge to be transformed into the representation language producing a representation that is not only syntactically correct but has a high degree of semantic consistency with the original elicited knowledge.

We therefore propose that a design methodology should have some or all of the following features: A domain definition, in which the domain is surveyed, analysed and considered for development. A knowledge elicitation phase, in which the knowledge is extracted from the domain expert through one or more techniques. A knowledge acquisition phase in which the elicited knowledge can be analysed. This analysis can take several parts: i) The selection of a suitable representation language by analysing the representational needs of that domain and ii) A formal analysis of the knowledge in order to ensure both semantic and syntactic accuracy. The formal representation can then be transformed into the selected representation language. Based upon this representation and other properties of the domain, a suitable control architecture can be implemented.

We believe however, that a methodology should not have isolated stages that are described only in general terms. This is a deficiency common to many of the earlier methodologies. The stages should be as detailed as possible providing guidelines and examples of practical experience when necessary and any transformations of knowledge in or between
stages should be capable of justification in order to preserve the semantic integrity of the knowledge.

It is the aim of the following chapters to present a methodology which reflects this design philosophy.
CHAPTER 3.

THE KNOWLEDGE ELICITATION PHASE.

3.1. Overview of the design methodology.

3.1.1 Introduction.

The aim of this section is to provide an overview of our development methodology. This will assist the reader to relate the individual stages presented in the following sections and chapters to the methodology as a whole, as well as the development philosophy that lies behind it.

3.1.2 Development outline.

A diagram that accompanies the following descriptions is given in figure 2 (page 21) and illustrates the inter-relationships between the stages.

Initial specification.

The development commences with the preparation of an 'initial specification'. The construction of this document allows the knowledge engineer to familiarise himself with the domain and start to build up a working relationship with the domain expert.

The specification is intended to provide guidelines on the domain boundaries, suggest areas of importance, glossary of terms etc. This can then act as a base line document for the remainder of the development.

The knowledge elicitation phase.

The initial specification is utilised in the second stage - that of knowledge elicitation - where it provides the knowledge engineer with the background and direction needed to commence the elicitation process. It is used in conjunction with further information gained from the domain expert and other sources to select the most suitable approach to elicitation. There are many elicitation techniques open to the knowledge engineer, for instance, interviewing, documentation or asking the expert to describe critical problems encountered whilst performing
his expert task, could be used. Thus the knowledge engineer selects the most appropriate approach and elicits the knowledge from the expert using the guidelines given in the initial specification.

The elicited representation.

The outcome of the elicitation phase is an elicited representation which, unless a purely visual elicitation technique has been used - such as video taping, will provide a representation which is textual in nature. This then acts as the 'raw knowledge base' upon which analysis can take place.

The knowledge acquisition phase.

Primary analysis.

The elicited representation that results from the knowledge elicitation is, as stated above, usually in the form of natural language text. This results in the representation containing many of the problems associated with natural language, for instance: ambiguity, incompleteness, inconsistency etc. It is therefore necessary for the representation to undergo a refinement process before the knowledge can be analysed and used to specify our knowledge base.

In order to select an appropriate 'primary representation' in which to represent the elicited knowledge, several steps have to be performed. These will now be briefly described.

The primary representation.

The first step in selecting a primary representation is to examine the elicited representation for the presence of certain characteristics that we feel underlie the vast majority of elicited text. For example: modularity of knowledge, linkage of knowledge, level of explanation within the elicited representation, etc. The levels to which these characteristics are present within the elicited representation are then plotted on a graph, this is known as a 'primary trace'.

The next step in selecting a primary representation is to examine which of the primary representations has the capacity to best fulfil the representational needs of the elicited representation. This is done by creating 'bandwidth diagrams' for each of the primary representations.
A bandwidth shows the upper and lower bounds feasible for each characteristic when the syntactic and semantic requirements of that representation are fulfilled. The primary representation that best accommodates and matches the trace of the elicited representation is selected. The domain knowledge is transformed from the first representation to the other.

Having created a primary representation, we are now in a situation to analyse and examine the knowledge more closely. Two analyses are performed: In the first, the primary representation is analysed and subjected to a re-representation process that has as its target language a formal specification language in which the elicited knowledge can be specified.

The second analysis of the primary representation is in order to determine which of the "classical" representations; such as frames, semantic networks, rules etc., would be suitable to use as a representation formalism.

The results of these two analyses are captured in the 'domain specification' and 'representation specification' respectively.

The domain specification.

The increased organisational structure of the primary representation over the elicited representation allows for a higher level of analysis to be undertaken upon it. However, the primary representation still has several drawbacks. It is still far too ambiguous and may contain inconsistencies and incompleteness that can not be spotted due to the structures used. Thus, it is the aim of the domain specification to help reduce these problems. The domain specification will have a mathematical basis and although our philosophy is to allow the knowledge engineer to have the freedom to choose a suitable language, we have suggested and adopted the "Z" notation throughout this work. This is a formal specification language developed at the Programming Research Group, Oxford [Sufrin, 1985] [Morgan, 1984].

The representation specification.

The process of producing a representation specification is similar to the problem discussed earlier - that of producing a primary representation - in that the knowledge engineer must consider the characteristics of the primary representation and judge which of the classical
representations has the best structures to support those characteristics. The characteristics are not the same as those for the selection of the primary representation but are based on different knowledge types present in the domain. Briefly, the process has three stages: Firstly, a ‘profile’ is constructed that shows the amount of knowledge a given domain has for a certain set of knowledge types.

Secondly, the profile is compared to profiles that show the amount of knowledge each of the classical representations is capable of supporting for the same set of knowledge types - where classical representations are frames, rules, semantic networks etc. Finally the most suitable representation is chosen and the syntax/semantics of this are denoted as the representation specification.

The concrete specification & the implementation.

The final stage is the creation of a concrete specification, which can act as a specification for an implementation. The concrete specification is a composite object composed of a ‘secondary representation’ and a ‘control architecture’. The secondary representation is a form in which the domain specification is represented in the language defined by the representation specification. The control architecture gives a description of the control mechanisms that should be used to manage and manipulate the secondary representation.

3.1.3 Conclusion.

The above section has given a brief overview of the stages involved in our development methodology. These stages will be described in more detail in the following sections and chapters.
formal language

initial specification

knowledge elicitation phase

elicited representation

Primary analysis

primary representation

Profile analysis

classical representation methodology

representation profiles

knowledge profile

profile matching

representation specification

representation specification

domain specification

formal language

concrete representation

secondary representation

control architecture

implementation

fig. 2.
3.2. Initial specification.

In this section we shall examine the role that can be played by an informal software requirements document. Such a document provides the knowledge engineer with an outline of the domain, the boundaries of the domain and the outline of any proposed systems for that domain.

3.2.1 Introduction.

An expert system can be viewed as a type of computer system and as such the development parameters that software engineers place on conventional software systems can also be applied to the expert system life cycle. The first stage of this life cycle is often referred to by software engineers as the software requirements document. In this methodology we refer to it as the 'initial specification'.

The aims of the initial specification are: To help familiarise the knowledge engineer with the domain whilst building up a working relationship with the domain expert, establish the area of interest for a possible system within the domain and where possible establish boundaries on the solution space of the problem.

We have found that the literature is weak in the area of initial software requirement documents for knowledge based systems. This may stem from the fact that most large and published systems have been built by the instigator of the system, rather than through a 'third party'. We will now suggest a series of steps that can be followed in order to arrive at an outline for an initial specification. These steps are useful in that they give the developer a chronological sequence of events that with adaptation could be applied to a variety of situations and domains. However, the knowledge engineer is encouraged to develop other steps and guidelines where necessary.

Step One. (Area outline)

The first step in the production of an initial specification is the creation of an "area outline". Here the instigator of the project drafts a report that outlines: i) The proposed project area, ii) The reasoning behind the need for the system, iii) References which introduce litera-
ture on the projected area.

Step Two. (Knowledge engineer surveys the area)

The knowledge engineer takes the report produced in step one and by following up the literature references, surveys the area. The survey is aimed to produce two results. Firstly, the knowledge engineer can gain further insight to the problem area and secondly, it provides initial views towards feasibility. The first of these results is a necessary pre-requisite of the second. This stage is primarily a familiarisation one, where a preliminary feasibility report is produced.

Step Three. (Prepare an initial specification)

The knowledge engineer, taking the feasibility report produced in step two, now tries by discussion with both the initiator of the project and if possible the domain expert, to focus the areas of the domain to be investigated.

The first area of discussion is a clarification process where the knowledge engineer tries to ensure that his fundamental knowledge of the domain is balanced and accurate.

The second area of discussion is more wide ranging, with an attempt to increase the knowledge engineer’s understanding of the more critical and sensitive areas within the domain. The problem of scale of domain can also be considered as it may be felt that the initial problem should be changed or scaled down.

Thus, the aims of this stage in the development of an initial specification are: Firstly, to clarify the knowledge engineer’s understanding of the domain, and having done this the knowledge engineer can appreciate the project put forward by the domain expert. Secondly, the domain expert gains an insight as to the applicability of his problem to solution by knowledge based techniques.

At the end of this stage the knowledge engineer will be in a position to produce a report detailing the domain definition and problem description. This is known as the initial specification.
Step Four. (Clarification)

The initial specification produced in step three then needs to be refined. This is an iterative process of meetings - discussions and amendments between the knowledge engineer, the project instigator and the domain expert. The aim is to produce a document which defines as clearly as possible the area under investigation and especially the boundaries to that area and define what problem types within the area are applicable.

3.2.2 Comments on the initial specification stage.

In the creation of the initial specification the domain expert has not explicitly been subject to a rigorous knowledge elicitation process. The interaction between the knowledge engineer and domain expert has been "low key" and "informal" with the emphasis being more on clarification than on elicitation. The only stages where it may have been necessary to tape record the discussion, were the initial stages of step three, where the knowledge engineer tries to clarify his understanding of the domain.

Having created an initial specification this can then act as a base line document that the knowledge engineer or domain expert can refer back to during any part of the development. It will also be useful in the post development stages - for instance in maintenance as an introductory document.

3.3. Knowledge elicitation.

3.3.1 Aims, roles and perspective.

The second major stage to be encountered in the development methodology is the knowledge elicitation phase. The aim is for the knowledge engineer to utilise the information gained in the creation of the initial specification, in order to elicit as much knowledge as possible from the domain expert using the most suitable techniques.

The term "knowledge elicitation" is used intentionally here as opposed to "knowledge acquisition". The literature at present gives a confused view of these two terms, for instance:

"Knowledge acquisition is defined as learning new symbolic information coupled with the ability to apply that information in an effective manner" [Michalski. 1983]
"Knowledge acquisition involves relating something new to what we already know in a psychologically complex way" [Barr. 1981]

"The process of eliciting, designing and implementing can overlap during development, so knowledge acquisition pervades all stages" [Hart. 1985]

"Knowledge acquisition is the transfer and transformation of problem solving expertise from some source to a program" [Hayes-Roth. 1983]

It can be seen that the terms tend to be used interchangeably to mean the whole process of constructing a knowledge based system, commencing with extracting the knowledge from the domain expert and extending to the implementation of this knowledge. In our methodology however, knowledge elicitation differs from knowledge acquisition in that the emphasis of the knowledge elicitation is upon the extraction, gathering and articulation of knowledge and information from the domain expert on a particular area of interest, whereas the emphasis in knowledge acquisition is upon the analysis of the information provided by the domain expert gathered during the elicitation phase. The two are not however discrete in their purposes, if for example say an interview takes place as a means of elicitation, the elicitation will involve a certain amount of analysis on the part of the knowledge engineer in order for the interview to be discursive, continuous and have a dialogue rather than being a set of disjoint questions and answers.

3.3.2. Knowledge Elicitation Techniques.

There are several different approaches that the knowledge engineer can take to the problem of knowledge elicitation. Some of the major elicitation techniques currently available are:

1. Verbal transfer of knowledge.
2. Knowledge engineer investigates literature information.
3. Expert uses intelligent intermediary program.
5. Knowledge engineer observes problem solving process.

Each of these techniques has its own subset of specialised techniques. We will now examine some of these approaches to the elicitation of domain knowledge.
3.3.3. Verbal transfer of knowledge.

The largest category of elicitation techniques falls under the general heading of 'verbal approaches to the transfer of knowledge from domain expert to knowledge engineer'. The area of verbal transfer can be further broken down into: i) Interview techniques & ii) Reporting techniques. These two areas will now be examined.

1. Interview techniques.

The generic term "interviewing" is used to cover all types of knowledge elicitation where the knowledge engineer verbally communicates over a given period of time with the domain expert in order to extract knowledge from that expert. Within this generic term there are however, many different techniques and styles of interviewing, some of which elicit specific types of knowledge within a specialised range of domain parameters.

We will first describe some of the techniques associated with traditional interviewing, that is, interviews which are based on the structured, focused or unstructured questioning of the domain expert by the knowledge engineer. This will be followed by an examination of the more formalised elicitation techniques that draw from the world of psychological interviewing.

Interviewing - Structured, focused & unstructured.

The technique known as interviewing refers to the process where the knowledge engineer elicits knowledge from a domain expert through a series of questions and points of discussion.

It is the placement of emphasis upon these questions, their order and relationship with each other, that decides whether the interview is structured, focused or unstructured, however the divisions between the three are not disjoint.

A significant factor in the structure content of an interview is the amount of prompting and guidance that the knowledge engineer puts into it. In an unstructured interview the knowledge engineer, after giving a few seed questions, allows the domain expert to develop the discussion, directing it into areas he feels are important. The knowledge engineer's job is mainly to ensure that the domain expert does not digress too far from the area of interest. In a structured interview the knowledge engineer takes much more of a leading role. The
knowledge engineer will try to regulate the depth of the knowledge the interview is generating in a more controlled way, introducing new information into the discussion when deemed necessary. This is something an unstructured interview consciously tries to avoid doing, instead working solely off information generated by the domain expert. A structured interview changes into a focused interview when the level of knowledge it is attempting to capture moves from the general to the specific. Focused interviews are designed to cover areas of the domain in greater depth, the interview may concentrate on one point or several.

Much of the literature from psychology advocates the use of structured interviews [Bainbridge, 1979] [Cuny, 1977], as the introduction of new points and ideas by the knowledge engineer acts as memory cues to the domain expert. As McDougall points out "recognition is much more accurate and complete than recall" [McDougall, 1904]. The use of memory prompts and leading questions by the knowledge engineer does however have its problems, in that the information flow generated by the domain expert may become unnatural, the priorities changed and the emphasis misplaced. This may be due to the domain expert having to talk solely about the areas that the knowledge engineer is knowledgeable or interested in. The knowledge engineer may on some occasions find structured interview techniques useful, for example, when the domain expert is not naturally forthcoming with information or is overly hesitant. The knowledge engineer can then if necessary revert back to an unstructured style of interview when a higher degree of response has been obtained from the domain expert. This balanced approach may be achieved through a mixture of leading and naive questions which probe the knowledge engineer to different depths.

Techniques & guidelines for interviewing domain experts.

The procedure for interviewing domain experts in order to elicit their knowledge can have no hard and fast rules. We feel however, that due to the weakness of the literature in relation to interviewing for knowledge based systems that it is useful to give some guidelines we devised during our case studies.

In conducting an interview we feel that it should be carried out in surroundings natural to the domain expert and in an environment where interruptions are kept to a minimum. The
length of an interview will need to be judged separately for each session. The aim is to keep the interviews short and focused and not allow them to go stale, digress or turn into general discussions. It is important that a situation of 'knowledge overload' for the knowledge engineer be prevented from occurring. This is where too many pieces of knowledge are being discussed and the interviewer lose his perspective, at which time the interview should be stopped for the knowledge engineer to reassess his position. The state of knowledge overload may be prevented from occurring in the early stages by asking the domain expert to give a seminar or lecture designed for an audience unfamiliar with the area or concepts. This should help the knowledge engineer obtain an overview of the domain. A seminar would also serve to familiarise him with the terminology used by the domain expert, and this should be used from then on for consistency. The terms can also be checked and clarified. The pressure of note taking or memorisation is relieved if the interviews are taped for later transcription.

A good interview technique for knowledge elicitation is a skill that has to be learnt by practice but the following guidelines give some of the underlying principals. The initial interviews will be general in nature, their aim being to obtain high level conceptual knowledge whilst building up a working relationship with the domain expert. Having performed the initial interviews the knowledge engineer can develop and select the most appropriate approach with which to proceed. For instance, if a structured interview is selected, then this technique will need applying with the utmost care so as the questions do not unduly influence the weight or emphasis of the experts responses. If the interview is to be structured then the knowledge engineer may find it useful to have a list of points to cover. These points can then be covered in any order and the digressions tolerated. The more unstructured an interview is the more important it is to be aware of what you want from it and ensure that the domain expert does not digress. In either case, the knowledge engineer must not project a false image of having too high a degree of knowledge about the domain or else the expert may feel pressured into giving false behaviour patterns in attempting to respond rationally and knowledgeably. The domain expert will talk much more freely if he thinks that the listener is interested and informed about the area, thus understanding what the expert is saying. The problem is to ask specific questions that cover the domain completely without knowing very much about the
It may be that an interview is not progressing as well as could be expected due to hesitancy on the part of the domain expert. In this case there are several techniques that we can employ to help increase the output of the elicited knowledge. 1) The knowledge engineer can, when appropriate, make use of ‘memory cues’, to prompt the expert. 2) The principal of using counter examples to cross check the information and prompt more expansive replies can be used and 3) the knowledge engineer can ask the domain expert to give all options at every disjunction, these can then be examined individually followed by their inter-relations.

In some instances it may be useful for there to be more than one knowledge engineer to be present at the interview. For example, when a large system is being developed it may call for multiple knowledge engineers and/or domain experts, in which case there may be instances where more than one knowledge engineer may need to be present at an interview. This can be a valuable exercise, if say two knowledge engineers are present then one can be deemed the ‘principal knowledge engineer’ who takes the leading role in the interview and who in a structured interview decides upon the directions it takes. The other, ‘secondary knowledge engineer’, can then act as a moderator, in that he can follow the discussion without being under the pressures that the principal is under. This enables him to identify where possible inconsistencies, ambiguities and omissions occur on the part of the domain expert. The secondary knowledge engineer can also see whether the interview is digressing and that all the points listed as part of a series of options to be examined have actually been covered. The secondary knowledge engineer, when he raises and discusses these points with the domain expert, provides the principal knowledge engineer with time to think. If there are two domain experts then they can have similar roles to those just described for two knowledge engineers, with one being the principal expert and chief respondent to the questions and the other monitoring, summarising and adding to the principal experts answers.

To conclude, we can say that the knowledge engineer should try and build up a good working relationship with the expert and promote clear communication between them. This is important as it is factors such as these that encourage an easy flow of ideas and information between the knowledge engineer and the domain expert in elicitation. It is very difficult to
elicit knowledge from someone who does not wish to give it and this state may occur if for instance the domain expert is suspicious of the knowledge engineer's motives or worried about discussing his work with a stranger. The ideal is for the domain expert to regard the knowledge engineer as a trusted colleague. The knowledge engineer, in order to be in control of the elicitation session should have a clear strategy and not lose his perspective whilst eliciting the information. A useful analogy is for the knowledge engineering process to be viewed as analogous to mining for a valuable ore. Strips can be removed first at the highest levels of strata revealing where the valuable seams are. These can then be mined in a vertical manner. Finally the interconnecting seams can be examined to fully cover the domain.

2. Reporting techniques.

The second major verbal knowledge elicitation technique is that of "self report". This title covers a large range of techniques and is based upon the domain expert describing how the expert task is performed either on-line, while the task is being performed or off-line or as a retrospective recall of the task. The knowledge engineer may or may not add to the report in the way of questions. An examination of some of these techniques will now be given.

On-line reports.

An on-line report consists of the domain expert explaining to the knowledge engineer what he is doing whilst actually performing his expert role. This report is tape recorded for later transcription. An advantage of the concurrent vocalisation approach is that the elicitation occurs in a natural situation with real cases to be examined and solved. The approach also places fewer demands upon the expert's time than some other methods. The method is however problematic in that it does not cover every situation that can occur, also domain experts cannot vocalise as fast as they reason. Some research by Berry has show that concurrent vocalisation "has no effect on task performance or question answering" [Berry. 1984], in fact there is evidence [Ericson. 1980] that it may in fact even improve task performance because a somewhat more explicit and orderly reasoning strategy may be adopted. However, several problems have been identified and associated with this approach, in that the domain expert may not report what is obvious to him and there may be a lack of control knowledge with the domain
expert not stating why one action was chosen over another.

Off-line reports.

The difference between an on-line and an off-line report is that in off-line reports the domain expert talks about his expert task whilst not actually doing it. It is sometimes called retrospective analysis if the expert is discussing a task previously performed. The aim of off-line reporting is to overcome some of the problems associated with on-line reporting, for instance it is not possible to generate on-line reports if the task itself involves verbal communication. The problem of the vocalisation of reasoning whilst performing tasks in real time can be overcome off-line as the domain expert is no longer under this temporal pressure.

The off-line approach to reporting not being constrained by real time, allows for variations of the straight verbal description of a task to be employed and several simulation techniques have been suggested in the literature [Bainbridge, 1979]. The simplest is where the domain expert is given a scenario and talks through what he would do, why he would do these things and what factors are being considered during the process. This basic approach is called 'static simulation' as the situation does not change whilst the domain expert analyses it, however for some domains it is useful to employ a more changeable scenario and this is modelled by 'dynamic simulation'. In dynamic simulation the domain expert has to contend with changing parameters within the domain. The domain is not modelled in real time and this allows the expert time to reason and report in a reasonable way on more complex problems.

In using retrospective analysis reporting, a problem may be encountered in that the domain expert may forget parts of his reasoning strategy. This knowledge will tend to be at a fine grain size and is of the most important types needed by the knowledge engineer. If the domain expert is using a simulated problem devised by the knowledge engineer then this model must be clearly defined as the domain expert may have a different mental model to that of the knowledge engineer.

The use of off-line reporting does not tend to place the same pressure on the domain expert as that for on-line reporting, which when used to analyse a complex domain may cause information overload to occur and detract from the expert's performance.
Hybrid techniques.

In order to overcome some of the problems associated with reporting, a hybrid technique can be used that incorporates both interviewing and reporting techniques, this is known as 'concurrent probing'. Here, the knowledge engineer, who in reporting tends to have a less than active role, interrupts the domain expert whilst he is performing his reporting task to ask questions. This technique can only be employed in situations where the on-line task is not time critical.

A situation that the knowledge engineer has to be careful to avoid however is that of task overload. This is particularly caused by unanticipated questions or questions that are not related to the task, where the task involves a highly complex dynamic situation.

Experiences with reporting techniques.

Experience from our work indicates that the main problem with reporting techniques tends to be incompleteness. This stems from the fact that the domain expert is generally reporting only what has happened in the tasks examined in the report, not what could happen in the wider context of the domain as a whole. The knowledge from reports therefore tends to be implicit as only the result of applying the knowledge, not the use of the knowledge are given. The problem of knowledge grain size is also experienced as expert's tend to assume that certain items of information are obvious and miss them out when in fact they are necessary for the knowledge engineer to understand the experts reasoning strategy.

A second problem that can be encountered is that of inconsistency. One possible way to overcome this is to repeat the elicitation and check the results. However, as the expert may regard this as a waste of his time an approach such as the following may be useful: when eliciting knowledge in a later session on an area, different but close to the one in need of confirmation, the knowledge engineer can move the questioning to overlap the area already covered and by using a different approach to the questions, tackle and check the area again.

3. Formalised interviews.

In order to extract specialised knowledge from the domain expert a set of focused elicitation
tion techniques have been developed and can be used to plug the gaps where the more encom­
compassing but consequently more generalised approaches are weak. Many of these techniques ori­
ginate in the field of experimental psychology.

**Repertory grid.**

One of the most useful techniques for domains which have less than approximately a
dozen factors influencing decisions made within it, is that of the repertory grid. Invented and
theorised by Kelly the approach involves systematic coverage of the interrelationships of the
objects and factors from which the domain is composed [Kelly, 1955]. The method involves
collecting a set of objects about the domain which are then presented to the domain expert in
threes. The domain expert is then asked to say in what way two of the three are alike and dif­
ferent from the third. The approach has the advantage of providing a complete examination of
all the objects in the domain and their relationship to each other. This helps to make it clear
what the key objects and features of the domain are. The trade off for this information is the
high number of comparisons that have to be performed. This in reality limits the applicability
of the approach.

**Critical Incident.**

The techniques used in the critical incident approach to structured interviews is based
upon the early work of J.C. Flanagan in 1954 [Flanagan, 1954]. The aim of this technique is
to obtain the most accurate recall possible, and the approach used to do this is for the
knowledge engineer to ask the domain expert to select especially memorable events and then to
describe why they were memorable, what the influencing factors or conditions were and how
that event relates to other experiences. The advantage here is that the expert finds it easy to
remember critical and outstanding cases and consequently finds them easy to describe and dis­
cuss. The problem with this approach is that due to the specificity of its nature, it can not be
used as a general purpose elicitation technique and has to be reserved for focused elicitation.
A secondary problem with the approach is that the expert may digress to generalities as more
and more influencing factors are brought into the reasoning behind, and influences on the criti­
cality.
Inference structure.

A technique described in the literature by Bennett [Bennett. 1981] [Welbank. 1983], that is applicable to domains which are diagnostic in nature or that have high amounts of cause and effect is that of examining the link between a set of problems and their associated symptoms. The approach involves firstly obtaining a list of all the possible problems that could occur within the domain and then a list of all the possible symptoms that could be looked for, over the whole domain. The knowledge engineer then asks the domain expert to specify which symptoms point to which problem. The advantage of this technique is that the inter-relationships become well covered, however these relationships may tend to be direct and give only shallow knowledge if the approach is taken too literally. A refinement that can be used in association with this technique is rather than just listing symptoms and faults, place intermediate steps in as well. A chain of these links could be of the form: Initial symptoms - type of problem - possible sub problems - interconections - solutions. The aim here is to establish what are the criteria at each stage that identify the set of symptoms at the next stage.

Goal decomposition.

A technique advocated by Grover is that of 'reclassification' [Grover. 1983]. The approach here is for the knowledge engineer to present the domain expert with a goal situation that needs to be reached. The domain expert is then asked to break down the goal into a set of facts that would, if available, provide evidence for that goal. This reclassification strategy is recursively applied to each of the subgoals until the base condition is met. This is where the facts can be directly observed by the expert. The strategy being similar to that of backward chaining. A problem with this system is that it is only suitable for highly factual domains.

Distinguishing evidence.

A useful approach identified by Welbank [Welbank. 1983] for eliciting knowledge in diagnostic domains is that of examining what distinguishes one possible answer from another in a given situation. For example, in the domain of starting a motor car the expert would be asked what the criteria were he used, to correctly diagnose that the reason for the car not starting was a faulty condenser and not a flat battery.
Summary of verbal techniques.

Having described a range of techniques that the knowledge engineer can use in obtaining knowledge from the domain expert in a verbal style, it is possible to see that different techniques are best applied to different situations.

One way to utilise the different techniques in eliciting knowledge is to start off by interviewing the domain expert. This will give a large amount of knowledge quickly at the higher levels of abstraction. It will also help to build up a working relationship between the knowledge engineer and the domain expert. Having analysed this information, more specific information can now be elicited. This may involve the use of a self report approach. For specific types of information and knowledge the specialised techniques such as repertory grids can then be used. Finally, focused interviews can then be used to extract very specialised knowledge.

3.3.4 Knowledge engineer investigates literature information.

Expert systems can be considered as a medium for perpetuating the knowledge of a domain expert for the use of future generations. This is a similar role to that of man's most widely used form of stored knowledge - the printed word. Text in the form of books, papers, journals etc., is a natural supply of information and knowledge, which if utilised carefully can be a most valuable source from which to elicit knowledge for the construction of knowledge based systems.

There are varying levels of text ranging from learned papers, journals and books through instruction manuals to guides and introductory text books. Thus, theoretically a high proportion of the information required to create a knowledge base can be found without referral to a human domain expert. This does not mean that knowledge elicitation from literary sources by the knowledge engineer is the panacea to get around the elicitation bottleneck but rather a useful additional source of information that the knowledge engineer can draw upon. The balance between theory and the practice is that the books may have a higher proportion of theoretical knowledge in them than practical knowledge whilst the human will be expert at applying the theoretical knowledge in real practical situations providing a rich source of heuristic, meta and

-35-
other forms of applied knowledge.

The development of very large systems (where large is a reference to the domain size and complexity) may be one area where this elicitation technique will be profitably employed in the future as it will reduce the amount of 'contact time' placed upon the resources of a human expert. The elicitation with the human expert can be reserved for non explicit forms of knowledge missing from the literature. For example, literary sources do not often explicitly state the control knowledge necessary to utilise the factual, heuristic and other types of knowledge present in the text, knowledge which can be provided by the domain expert.

The domain expert thus has several roles, firstly, to provide knowledge in say a verbal form, secondly to advise on literary sources from which knowledge can be elicited and thirdly comment, arbitrate and add to the textual information.

We therefore suggest that the technique of using literary elicitation techniques should be combined with the other elicitation techniques to elicit levels and amounts of knowledge relevant to a given situation and domain.

3.3.5 Expert uses intermediary program.

There has been some recent research performed in examining the possibilities of developing intelligent systems that allow the domain expert to enter knowledge directly into a knowledge base.

The reasons for such a system stem from the fact that this would help overcome the "knowledge elicitation bottleneck" [Feigenbaum, 1979] in that a large knowledge base may be built more quickly when non-experts do not have to be intimately involved in describing each object in the knowledge base. The creation of a knowledge base directly by the domain expert may aid the accuracy and completeness of the knowledge as it may become distorted during elicitation by knowledge engineers, especially as much of the knowledge may be complex and subtle, with the knowledge engineer not fully aware of its purpose and real value.

There are however many problems with the intermediate program concept. Firstly systems of this sort will be limited to elicitation of only specific knowledge types as their representational formalism will have been built to hold knowledge based on a set of
assumptions regarding such factors as: type of domain, the human computer interface and user profile.

A second problem with this type of system is the assumption that the domain expert is of sufficient computer literacy to interact with the system.

The third consideration is that the computer system can not be as flexible as the human knowledge engineer who not only elicits the knowledge but helps the domain expert to formalise his knowledge.

We feel that even with the apparent success of systems like MOLGEN [Stefik. 1981a] [Stefik. 1981b] and UNITS [Stefik. 1980] that this technique is still too limited in applicability and until very sophisticated man machine interfaces are developed along with very flexible representation capabilities these systems will stay in specialised environments.

3.3.6 Machine learning.

The process of knowledge elicitation has been acknowledged by knowledge engineers as the most time intensive process involved in the creation of an expert system. This is due in part to the very nature of the exercise where the knowledge engineer acts as an interface between the domain expert and the computer system, having the role of firstly extracting the knowledge from the domain expert and then reformulating it so that the knowledge can be acquired by the system. Thus the knowledge engineer first has to ‘learn’ the knowledge in order to ‘teach’ it to the machine. It has therefore been suggested [Michie. 1982] that the knowledge engineer’s role should be curtailed. Michie suggests that the system should be able to learn inductively by analogy - examples. The initial knowledge base having been created rapidly by the machine through machine-domain expert interaction the knowledge engineer can then take a more active and predominant role in helping to improve, refine and develop this knowledge base through other techniques such as verbal elicitation with the domain expert. This type of knowledge elicitation is called machine learning.

I will now briefly outline two techniques from those of machine learning [Michalski. 1980]. One approach to learning that humans take is learning by example and this can be carried over into the machine learning world. The system is given a series of example situations
and decisions by the domain expert, these are often called the ‘training set’ [Quinlan. 1979] along with a set of attributes which are parameters that can influence the outcome of the situation. These factors are then used by the system in an inductive process that attempts to find general principles around which decisions affecting the outcome are made. This area has been one of the intense research and a review of many inductive methods is given by Dietterich & Michalski in [Dietterich, 1983].

A second approach, again modelled on one way in which humans learn is that of learning by analogy. This approach is different from that of learning by examples because the emphasis of these systems is upon the transformation and augmentation of old knowledge in order to solve a new problem. Thus in solving a (new) problem the system examines its knowledge base and attempts to find a similar situation - an analogy - that it could transform into a close approximation of the new problem and by similarly transforming the old situation solve or partially solve the new problem. Again this is an area of intense research and a large literature on the approach is being built up, [Carbonell. 1987].

We feel that machine learning will become more important in the future as techniques improve and its applicability becomes wider. An interesting comparison of machine learning techniques by Dietterich & Michalski can be found in [Dietterich, 1983].

3.3.7 Other elicitation techniques.

Having covered four of the main elicitation techniques, it should be stated that these are not the only ones available to the knowledge engineer.

There are several techniques which can be briefly mentioned. Firstly, observation can be used by the knowledge engineer as an aid to any of the other techniques, this may for instance be in the form of videoing the domain expert performing an expert task, the tape can then be used later in retrospective analysis. A second approach is for the domain expert to give written accounts of his expert action or to fill out questionnaires. These methods are however time consuming for the domain expert and multiple choice questions require the answers to be known before hand implying the test is on how much the expert knows rather than any deeper knowledge. Finally Gammat & Young advocate 'concept sorting' as "a helpful technique for
obtaining organisational knowledge" [Gammat. 1985].

3.3.8 Summary of knowledge elicitation techniques.

In this section we have briefly described the major knowledge elicitation techniques available to the knowledge engineer. Each of these has different properties and elicits different areas of the domain knowledge. The way that these should be utilised by the knowledge engineer will vary depending upon the situation and the domain. In one of the more standard problems such as medical diagnosis, one approach could be:

<table>
<thead>
<tr>
<th>Stage in elicitation process (% of k. gained.)</th>
<th>Techniques advocated</th>
<th>Aim of knowledge elicitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - Minimal</td>
<td>Tutorial/lecture</td>
<td>General - High level knowledge.</td>
</tr>
<tr>
<td>2 - Low</td>
<td>Unstructured interview</td>
<td>Conceptual knowledge</td>
</tr>
<tr>
<td>3 - Intermediate</td>
<td>Structured interview</td>
<td>Declarative knowledge</td>
</tr>
<tr>
<td>4 - Significant</td>
<td>formalised</td>
<td>Dependant upon technique.</td>
</tr>
<tr>
<td>5 - High</td>
<td>focused interview</td>
<td>Fine grain size of knowledge.</td>
</tr>
</tbody>
</table>

The elicitation process commences (1 - Minimal) where the knowledge engineer has only a cursory knowledge of the domain. The quickest and most efficient way to increase the amount of knowledge elicited on the domain is for the domain expert to give a tutorial/lecture like interview where as like a formal lecture the student only asks questions at formally prescribed times i.e., at the end. This lecture-interview will give large amounts of high level, general knowledge with which the knowledge engineer can plan the second phase of the elicitation (2 - Low) which could be an unstructured interview. This would give the domain expert a way of presenting further conceptual high level knowledge to the knowledge engineer. In an interview of this kind the knowledge engineer would be able to discuss with the expert areas of interest in order to gain enough knowledge so that at the third stage (3 - Intermediate) a more structured interview can take place. The third phase will elicit declarative knowledge of many types and will be where the main emphasis of the elicitation process would be. After the knowledge engineer has performed his structured interviews he can then evaluate the information and knowledge gained thus far and where possible isolate areas where the elicited knowledge is weak. He can then attempt to fill the gaps in the knowledge by using the formalised techniques described earlier (4 - Significant). Which technique should be used depends
upon the type of information that needs to be elicited. Having elicited a significant amount of
knowledge the final elicitation (5 - High) can be preformed by the use of focused interviews.
These interviews can provide the fine grain sized knowledge essential to the building of a
knowledge base, for example control knowledge, fine grained heuristics, subtle differentials in
diagnoses and treatment may all be elicited here enabling a knowledge base to be created with
high levels of completeness, consistency and other desirable characteristics.

This is of course only one scenario of many that could be considered each of which util-
ises some subset of the techniques described. In order to help differentiate which technique
would be of use in which situation the following table was constructed. It results from our
experiences in constructing case studies and our view of the collective literature surveyed.

<table>
<thead>
<tr>
<th>KNOWLEDGE ELICITATION TECHNIQUE</th>
<th>TYPE OF KNOWLEDGE</th>
<th>USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTERVIEW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unstructured</td>
<td>general - high level</td>
<td>surface level knowledge, large grain size</td>
</tr>
<tr>
<td>structured</td>
<td>conceptual to factual</td>
<td>general purpose</td>
</tr>
<tr>
<td>focused</td>
<td>factual - low grain size</td>
<td>specific - localised knowledge and information</td>
</tr>
<tr>
<td>REPORTING</td>
<td></td>
<td></td>
</tr>
<tr>
<td>on-line</td>
<td>sequential process, knowledge control, factual, heuristic declarative</td>
<td>control and application of declarative knowledge, non demanding real-time situation.</td>
</tr>
<tr>
<td>off-line</td>
<td>As for on-line plus added explanation and elicitation - procedural</td>
<td>As for on-line plus focused procedural knowledge. Used on smaller areas of a domains than on-line reports. Demanding real-time situations.</td>
</tr>
<tr>
<td>Hybrid</td>
<td>As for on-line reports plus variable grain size elicitation and added justification and elicitation</td>
<td>non demanding real time situations.</td>
</tr>
</tbody>
</table>
3.4. The elicited representation.

3.4.1 Introduction.

The aim of this section is to describe the role of the 'elicited representation'; what the criteria are for allowing a form to be an elicited representation, the problems associated with an elicited representation and how they act as the cornerstone upon which the rest of the development is based.

3.4.2 Aims, role and perspective.

An underlying principle of the development method is that every step within it should be capable of justification. It is important therefore, to have a firm foundation upon which to start the development and this is the role of the elicited representation.

The elicited representation is the direct result of knowledge elicitation and acts as the basis of the knowledge acquisition process. This latter process takes the information provided by the elicitation stage in order to deduce what the best development path for the knowledge
should be. In order to reduce the necessity for a new individual set of analytical techniques to
be developed each time an elicitation representation is produced, the decision was made to re­
strict the types of elicited representation that could act as the basis for the knowledge acquisi­
tion stage. Inevitably this restricts the number of knowledge elicitation techniques available to
the knowledge engineer.

The major restriction placed on the elicited representation is that it must be textually
based. The reasoning behind this is that text is a naturally occurring resultant of many
knowledge elicitation techniques, whilst also being a suitable form around which to develop
analytical techniques.

There are also secondary restrictions placed upon the elicited representation. These are:
that the representation should be a purely textual one, not containing diagrams, tables or other
types of figure. This step is taken as the development of techniques to analyse the meaning of
these forms and their relationship with other elicited knowledge is beyond the scope of our
work as presented here.

It should be noted that the elicited representation contains 'elicited knowledge' which has
a different definition to 'The domain knowledge'. The domain knowledge is intended to be the
'universal set' of all knowledge about the domain held in different raw forms e.g., books,
domain experts, data banks, where as elicited knowledge is just the subset of this which is held
initially in a textual form known as the elicited representation. The value of this elicited
representation can not be judged until the analysis of the elicited representation has been per­
formed and the knowledge is transformed into a more rigorous form.

3.4.3 Elicitation techniques for the methodology.

Having stated that the elicited representation must be textual in form this then raises the
question of which knowledge elicitation techniques produce textual output? Examination of the
elicitation techniques discussed in chapter 3.3 reveals that all of the following are capable of
producing a textual representation if the knowledge engineer, domain expert interaction is
taped and transcribed:

Interview
  - structured
  - unstructured
3.4.4 Example elicited representations.

The following are three examples of the commonest forms of representation resulting from the knowledge elicitation process.

The first example is of an interview on Statutory Sick Pay [Greene. 1987] between a knowledge engineer and an accountant acting as a domain expert.

Key: C = Knowledge engineer, BT = Domain expert.

C An employer rings into the office and asks if an employee is eligible for SSP, what questions do you automatically ask yourself to determine if the employee is eligible for SSP?

BT First of all you must say to them are they in receipt of a salary above the lower limit which at the moment is 38.00 per week. Obviously if anyone is below that wage for a start you discount them immediately.

C And that is averaged out over the 8 proceeding weeks.

BT You might get someone who comes along and because they are off sick and they hear about the SSP thing they go along to the Doctor and get a SSP note from him well if you employ an employee at the wage of say 35.00 per week you don't have to go any further there is no liability at all. I know shopkeepers who deliberately keep themselves below the limit 35 - 36 -37 - 37.50 which is below that limit, so if there is anyone below that limit of 38.00 then there is no SSP to pay.

C That's because they don't pay any National Insurance contributions paid.

BT Yes that's correct because you are only going to pay NI on a figure of 38.00 or above then you may on the off chance for example ask if in the previous 8 weeks has the employee had a bonus, I mean if someone has had a bonus paid to them let us say for example 150.00 bonus on top of their normal salary of 35.00 with the bonus on top of that, for that particular week therefore they have got 185 in the week now the previous week surrounding that if it was within the 8 weeks previous to going off ill well of course because they have had that big payment in that particular week then they have gone off sick, it only needs that one weeks payment their average pay could then shoot over that 38.00 do you follow what I mean?

C Yep
BT So there are 2 points to say,......

The second example is a representation from an on-line report [Fox. 1985] where the domain expert is performing concurrent vocalisation without any knowledge engineer interaction - 'concurrent vocalisation':

This patient is a 50 year old adult who in common with the last patient was said to have chronic myeloid leukemia diagnosed in 1978 and has now gone into blastic transformation and is said to be lymphoid by morphological and cytochemical criteria. ... Looking at bone marrow which has 70% blasts. Um I also know, though I can’t give a numerical value to it, that when this value of 70% is obtained from a smear of blood then......

The third example is a pure textual form, taken from a literary source [Sully. 1983] and has the following form:

"Engine cranks very slowly or not at all - turn on the headlights; if the lights are dim, most likely the battery or connecting wires are at fault. Check the battery using procedures described in chapter 7. Check the wiring for breaks, shorts and dirty contacts. If the battery and connecting wires check good, turn the headlights on and try to crank the engine. If the lights dim drastically the starter is probably shorted to ground. Remove the starter and test it using the procedures described in chapter 7...."

3.5 Comments and conclusions.

The chapter opened with an overview of our development methodology and then proceeded to expand upon the first three phases of that methodology: The initial specification, Knowledge elicitation and the elicited representation.

The aim of discussing the initial specification was to show the importance of clearly defining as early as possible in the development process the domain boundaries and the parameters within which the knowledge engineer has to work. A series of steps were suggested that could be used to work towards achieving these two goals.

The second section of this chapter discussed the area of knowledge elicitation and the major knowledge elicitation techniques available to the knowledge engineer. Three points were drawn from this discussion:

Firstly, that the term 'knowledge elicitation' is distinct from 'knowledge acquisition', the former signifying the extraction of knowledge from an expert source, whilst the latter is the analysis of the extracted knowledge.
The second of our observations indicates that the selection of an elicitation technique depends upon the type and level of knowledge to be extracted from the knowledge source. It may also be necessary to utilise more than one elicitation technique for a domain of non-trivial depth. To assist in this selection process a summary of the elicitation techniques was provided plus suggestions on how the knowledge engineer may wish to vary the elicitation process to extract the different knowledge types as the elicitation progresses.

Thirdly, the weaknesses of the literature on elicitation techniques designed for use in the area of knowledge based systems, highlighted the need for more research into the area.

The final section of the chapter discussed the representation that results from the elicitation process - the elicited representation. The decision was reached that in order to develop a system via a rigorous approach it would be necessary to have a firm base upon which to commence the development process. The survey of elicitation techniques in this chapter indicated that the output of several was or could be transformed into, a textual form, thus providing a firm representation upon which the knowledge engineer could apply a variety of analytical techniques.

This chapter has shown that once a detailed domain definition has been produced the knowledge engineer can, by utilising one or more elicitation techniques, extract knowledge of varying types and levels from knowledge sources to produce a textual elicited representation upon which a more detailed analysis can ultimately be performed.
CHAPTER 4.

THE PRIMARY ANALYSIS PHASE.

Introduction.

The knowledge engineer, having elicited raw knowledge and information from the domain expert must then consider ways in which this knowledge can be refined. The design methodology as a whole has several refinement stages which ultimately, through progressive re-representations, enable a concrete specification of the elicited knowledge to be written. These processes form the knowledge acquisition phase and will be considered in this and the next chapter.

The first of these representation refinement stages is the primary analysis stage. The aim of this stage is to analyse the elicited representation in order to identify and then re-represent the knowledge into a form with increased organisational structure known as the primary representation.

4.1. Primary analysis.

4.1.1 Aim, role and perspective.

In this section we examine the 'selection process' of the primary analysis phase. This consists of three parts. Firstly, the characteristics of the elicited representation are examined. These are then compared against the representational ability of the various primary representations. Finally, the primary representation that has the ability to best represent the characteristics present in the elicited representation is selected. The aim is to ensure that the closest match between the elicited knowledge and the form that represents it is found, thus giving maximum adequacy of representation.

This process and what is meant by 'adequacy of representation' will now be described in detail.
4.1.2 Primary representations.

As previously stated the aim of this phase is to take the elicited representation and represent it in a form with an increased organisational structure. This structured form or 'primary representation', will then enable us to reason about the domain in more analytical ways.

A primary representation should possess several characteristics, in particular it should have a very simple syntax, semantics and be weakly typed. This should enable a balanced compromise to be made between rigour and flexibility. Rigour is used in the sense that the structures utilised in a primary representation should encourage unambiguity and highlight any lack of consistency or completeness, whilst the representation should be flexible enough to mould the primary representation around a variety of elicited representations.

There are several consideration's that must be taken into account when developing or selecting a primary representation. A premier consideration is to ensure that the characteristics of the representation are sufficient to prevent information loss or change of semantic meaning from the elicited representation. One crucial factor is the representations amenability to the transformation of large quantities of textual information. This is important from a practical point of view as the texts obtained from transcriptions are extensive and will often run into many thousands of words (The transcription given in chapter 7 is approximately 3000 words long for one hour of interview). The prevention of information loss or change in semantic meaning is assisted by giving the representation little or no capacity for making inferences. This is because i) the representation will not be required to act as a basis for inference ii) inclusion of these facilities would only add to the complexity of the language and form and iii) the significance of each knowledge item is as yet undetermined and a large change in representational form would be premature at this stage.

Having compiled this set of characteristics describing the form a primary representation should take, three representations which adhere to them were considered: The flow diagram, the contour diagram and the decision table. These three primary representations will now be briefly described.

The flow diagram is based upon the flowchart idea but, instead of detailing the flow of
control for a program we try to establish the knowledge flow contained within the elicited representation. For example see figure 8.

The contour diagram is generally used for elicited representations which have a finer granularity of textual information than that used by the flow diagram. This is because the finer the granularity, the more care has to be taken in order to ensure that semantic integrity, correctness and other forms of adequacy are maintained, so preventing information loss or error in the re-representation and transformation of knowledge. The preventative measures necessary to maintain these qualities resulted in the techniques for a structured decomposition of text and the contour diagram form of primary representation. An example contour diagram is given in fig 10.

Decision tables are also concerned with the knowledge flow within the elicited representation, however unlike flow diagrams which can represent several data types, decision tables focus purely on organising and documenting complex decision procedures that may occur in an elicited representation. The decision table can be utilised in different situations in preference to primary representations such as the flow diagram which requires the elicited representation to have certain levels of sequencing and linkage of information in order to be of use. These characteristics are not however needed to the same extent for the creation of a decision table and thus it offers a more flexible form in which to represent unstructured information.

These are not the only possible primary representations and the methodology can be extended by the knowledge engineer to encompass other representations when necessary.

The three primary representations outlined will be examined in greater detail later, but the question is raised at this point as to how the knowledge engineer can tell which is the most suitable primary representation to use for any given domain, situation and elicited representation. This is the area that will be examined next.

4.1.3 The selection process.

Overview.

This section aims to describe the process of selecting the most suitable primary represen-
The elicited representation produced in the knowledge elicitation phase is analysed for the presence of certain characteristics. The levels of these characteristics are then plotted on a graph to form a "primary trace". The capacity of the primary representations for handling these characteristics is shown by use of a "bandwidth diagram". The primary trace is then compared against the bandwidth diagrams and the representation for the best match is then selected.

The last three of these stages can now be considered in greater detail.

Trace creation.

Introduction.

The aim of this stage in the selection process is to determine what the constituent characteristics are that compose an elicited representation and to what extent each of these characteristics is present in the representation under consideration.

The results of examining the elicited representation are to be given on a graph which is to be known as a primary trace.
It has the following form:

![Diagram](chart.png)

We will now describe the graph in more detail.

**Characteristics scale - Horizontal axis.**

One set of characteristics that to some extent can be found in all elicited representations is the following:

1. Noise
3. Linkage of knowledge.
4. Operational types.
5. Sequencing.
7. Explanation.

These characteristics are sequenced in a specific order, commencing with what could be termed low level characteristics and progressing to higher level ones. This spread of characteristics helps to produce not only a balance within the resultant trace but also allows them to be applicable to a wide range of elicited representations.
Each of these characteristics will be examined in greater detail later.

Vertical axis.

In drawing the vertical axis of the graph a major question that has to be taken into account is whether the y-axis is to be discrete or not. After careful consideration we took the decision to make the vertical axis both non numeric and non discrete.

A numerical scale is not acceptable for the purpose of assigning values to primary characteristics, as it is not possible to obtain meaningful numeric values from elicited representations regarding the presence of these characteristics. For instance, the amount of the "linkage" characteristic present in an elicited representation can not be considered in terms of percentages. Thus because of the difficulty of obtaining a precisely quantified scale a subjective non-numeric interval scale was chosen.

However we believe that it is possible to recognise certain "levels" when considering the presence of the primary characteristics. In the crudest sense, we can seek to distinguish the two extremes 'minimal' and 'high' together with a moderator 'intermediate'. Moreover, in view of the common experience of five point scales, we believe that two further subdivisions can be added to clarify the distribution. Thus the vertical axis has the form:

```
High
 Significant
 Intermediate
 Low
 Minimal
```

The non numeric ranges are also non discrete and are considered to overlap slightly. This allows the knowledge engineer flexibility in assigning values to characteristics, especially those in borderline situations.

Finally, due to the nature of the scale we feel that it is inappropriate to assign the value
zero to any factor as this is a specific point on a numerical scale.

The primary characteristics.

Introduction.

As stated earlier, there are seven primary characteristics which may to some extent be
found in all elicited representations.

The assignment of values for a characteristic is left up to the discretion of the knowledge
engineer who will become more skilful at assignment as more situations and problems are
examined. Thus, the guidelines given here are the fundamental ones enabling the knowledge
engineer to understand the philosophy of the approach. These guidelines have been gained
through the development of several real case studies, the details of two of which can be found
in chapters 6 and 7.

We will now be look at the guidelines for drawing traces and the assignment of values to
different characteristics.

Noise.

The 'noise' characteristic can be categorised into two levels - a 'syntactic noise factor'
and a 'semantic noise factor'.

The lowest level of noise is syntactic noise. This can be defined as "words or segments
of text within the elicited representation that are undefinable or unrecognisable". For example,
expletives such as "Umm" or "Ah" can be considered as noise as can repetitions within sen­tences. Sentences that are not finished, sentences that change direction and are meaningless are
also candidates for being described as noise.

The semantic noise factor is information in the text of the elicited representation that is
not pertinent to the problem task at hand. This type of noise is far more difficult to detect and
assistance may be needed from a domain expert or knowledge engineer who is very familiar
with the domain to assess this noise.

A block* that has no noise in it at all will be labelled as 'minimal' as will blocks that

*) If a section of transcript is examined for the occurrence of a primary characteristic this section is known as a 'block'
have only a few instances of syntactic and/or semantic noise. As the noise grows in regularity and frequency then so does the value that can be assigned to it. When the stage is reached such that for syntactic noise the repetitions of phrases and uncompleted sentences is becoming detrimental to a clear meaning of the block then the value assigned to noise may be significant or high. This is also the case for semantic noise as the focus of the text moves away from the problem area.

Modularity.

A module of knowledge can be described as a collection of statements or information with the following characteristics: Firstly, that the statements are all related to a single subject which is a subset of the domain. Secondly, that the statements are physically in close proximity to each other in the transcript or block under consideration.

A block that is composed of different themes and ideas, without any of them combining to form a module can be said to be minimal in modularity. Again as the themes become areas of specialist attention within the block then the value assigned to them rises. The picture is complicated somewhat by the size a module may take. For instance a module could be the entire size of a block or the block could be composed of several modules.

The modularity can be measured as the amount of information in a block that is associated with a module to the amount of information not contained in any module. For example, if there is one module which takes up all the information contained in the block this is high. If there is only one module and it takes up only half of the information contained in the block then an intermediate value can be given. Similarly, the value can increase/decrease as the value changes. If there are say ‘n’ modules and together they take up all the information in a block, again a high value can be assigned, this value decreasing as the amount of modularised information decreases.

- this term will be used throughout the following descriptions.
Linkage.

The linkage parameter can be broadly defined as the relationship modules have with one another and is a measure of their independence.

Linkage is a measure of the inter-relationship of the knowledge within modules. If all modules have a common theme but address different aspects of that theme then the modularity is high. As the number of modules that address that theme decreases then so does the linkage. However, if there is more than one theme in a block with there own associated modules then the cumulative result is taken.

Operational types.

The operational type characteristic reflects the different types of operations that are performed within the elicited representation. For example, condition, action, diagnostic and test are all operational types. The number of operational types can be used as an indicator of the complexity of the domain knowledge as the more operation types there are the more complex the elicited representation is likely to be.

The value assigned to an operational type is obtained by examining the ratio of the operational types to the volume of information in the elicited representation. For example, if there are two operational types in a large volume of information then the value is low, but if the volume is low with two operational types then the trace can be assigned the value significant or high. Whilst a large number of operational types in a large volume of information generates a trace value that is significant or high.

Sequencing.

The sequencing of information and knowledge within an elicited representation can, like the previous characteristic, be regarded as a complexity measure. Sequencing refers to the relationship that individual items of knowledge have with each other within the block under consideration and can be considered in terms of the temporal ordering of that information. If the sequencing is critical then it is important that a primary representation able to capture this aspect of the domain knowledge is used. It is important that not only the sequencing of infor-
formation within a linear block is taken into consideration but also the sequencing of information within and between the modules that compose a block.

The assignment of values starts with minimal sequencing when there is little or no sequencing. The trace value rises to intermediate when every item in the elicited representation has a sequencing dependency - i.e., the pieces of information follow on from each other. When the information also involves forward, backward and cross references the trace value can rise from intermediate into significant and high.

**Justification and explanation.**

Justification is the process where by the decision to perform an action or operation for a situation is shown to be correct, whereas explanation is defined as "explaining, especially with the view to mutual understanding" [Oxford Concise Dictionary].

Both of these characteristics may be associated with a whole block, a module or an individual statement and as such the value assigned to them will be the ratio of statements in the whole block that have a justification/explanation associated with them. An intermediate trace value assigned to values tending towards fifty percent with higher and lower values being assigned around this point.

In looking at explanation we are considering the way that the domain expert qualifies statements and links information together. We use this to advantage when considering the semantics of the elicited knowledge and it assists us in the representation refinement processes that are fundamental to our development methodology.

**The primary trace.**

Having examined the elicited representation for the seven characteristics in the manner just described it is possible to draw a primary trace to show the presence of each characteristic within the elicited representation.

For example, figure 6 shows a primary trace of an elicited representation which has low noise and justification, intermediate levels of modularity, linkage, sequencing and explanation with a significant amount of operational types.
Trace Matching.

Introduction.

Having created a primary trace for the elicited representation the next step is to determine which primary representation would be the most suitable for the re-representation of the elicited knowledge. In order to do this the representational ability of the primary representation has to be known, as only then can the most suitable match be made.

Bandwidth diagrams.

In a similar manner to that used for primary traces where values are assigned to signify the presence of a characteristic in an elicited representation, the ability of a primary form to represent these characteristics is given by a "bandwidth diagram":

The following diagram:
is an example of a bandwidth diagram for an imaginary primary representation. It shows that the representation language can adequately represent information providing that the noise, modularity and linkage level of the primary trace is between the minimal and significant levels, the operational types and sequencing are between the high and the low levels, whilst the justification and explanation characteristics are between low and significant levels. Outside these boundaries the use of this particular representation would be less than optimal.

We will now consider the three primary representations mentioned earlier and discuss their bandwidth diagrams.

4.1.4 Flow diagrams.

Let us consider the following diagram:
CONDITIONS
1: starter turning
2: loud grinding noises
3: gear teeth broken
4: pinion drive assembly ok
5: engine engages
6: starter disengages
7: manual gearbox
8: Headlights go very dim
9: battery and wires ok
10: faulty starter
11: crank starts normally
12: starter makes noises

TESTS
1: check gear teeth
2: pinion drive assembly
3: turn on headlights
4: check battery
5: check wires
6: check starter
7: test crank

ACTIONS
1: remove starter
2: replace pinion drive assembly
3: replace starter
4: the pinion can be fed by rocking the car in high gear
5: remove starter
6: replace starter
7: replace starter and wires
8: short the solenoid terminals together
9: check the solenoid wiring up to the ignition switch and seat belt relay
10: check starter
11: replace starter

DIAGNOSIS
1: Could mean the teeth on the pinion flywheel are not meshing properly or it may mean that the overrun clutch is broken
2: Usually a sticking solenoid or occasionally the pinion can jam on the flywheel
3: Usually a defective pinion or solenoid shifting fork, it may also be that the teeth on the pinion, flywheel ring or both are worn down too far to engage properly
4: Could be battery or connecting wires
5: Starter is probably shorted to ground
6: Could be solenoid, starter or wiring
7: Could be wiring
8: Faulty starter

fig. 8.
This shows a flow diagram for a domain where the elicited representation deals with a non starting motor car. From the diagram several things become apparent. One of which is the power that diagrammatic representations have over textual ones, an aspect that although not greatly explored in the past is becoming recognised (in work such as the KEATS system [Motta, 1986]) as potentially very important e.g knowledge representations that employ visual characteristics become not only useful to the knowledge engineer but also to the domain experts as they are easily understood by laymen. The visual form also enables problems in the representation such as incompleteness and inconsistency to be highlighted.

The way in which a flow diagram is created will be described later, the next thing we will consider is the representational abilities of this form.

The bandwidth diagram for a flow diagram is as follows:

![Bandwidth Diagram](image)

We will now outline the reason for the shape of this bandwidth:

Noise.

The ability to represent noise in a flow diagram is limited, this is due to the inability of assigning an operational type to noise whilst flow diagrams are strongly typed with the typing based upon these operations.
Modularity.

The way in which a flow diagram is created is such that the development is easier when the elicited representation contains modules. These modules are used to create small flow diagrams which ultimately get combined into the large final diagram. Thus, an elicited representation that has only low modularity does not assist in the creation of a flow diagram as the pre-requisite is that the information be related and 'flow' into one another.

Linkage.

Due to the final form of the flow diagram there is a strong necessity to have significant amounts of linkage between the modules of knowledge.

Operational types.

The flow diagram necessitates that there be several knowledge types. The minimal number are 'conditional' and 'action', where this is rated as low. Above this the flow diagram can cater for as many types as necessary.

Sequencing.

In order to construct a flow diagram an intermediate amount of sequencing must be present or else the knowledge would be garbled, the sequence being temporally based.

If the guidelines for constructing the flow diagram were changed to include an instruction that allowed the temporal order of information to be changed then the bandwidth could be widened.

Justification.

Any level of justification can be covered within a flow diagram. Justifications are desirable but not essential components of a flow diagram as a box can always be added to the flow diagram as and when the information becomes available.

Explanation.

As for justifications, explanations are also desirable but not essential components of a flow diagram.
4.1.5 Contour diagram.

The following figure is an example of a contour diagram:

<table>
<thead>
<tr>
<th>RULE: IF language = high THEN assembler = not applicable</th>
</tr>
</thead>
<tbody>
<tr>
<td>RULE: IF problem = business THEN language = COBOL</td>
</tr>
<tr>
<td>RULE: IF problem = scientific THEN language = FORTRAN</td>
</tr>
<tr>
<td>RULE: IF problem = a.i THEN language = LISP</td>
</tr>
<tr>
<td>HEURISTIC: IF programmer = novice</td>
</tr>
<tr>
<td>AND environment = sophisticated</td>
</tr>
<tr>
<td>THEN success = limited</td>
</tr>
<tr>
<td>RULE: IF language = low level THEN assembler = applicable</td>
</tr>
<tr>
<td>RULE: IF machine = IBM PC THEN language = 8088</td>
</tr>
<tr>
<td>RULE: IF machine = ATARI THEN language = 68000</td>
</tr>
</tbody>
</table>

fig. 10.

It shows the final state to which the representation, by following the guidelines, develops. The aim is that through structured decomposition of text, the inter-relationships of the information contained within that text can be expressed. The nested levels of information within the contour diagram attempt to show and reproduce the nested way that information is related and structured within the elicited representation.

This form, as for flow diagrams, utilises the diagrammatic component of the representation to help the knowledge engineer's task of analysing the elicited representation.

The guidelines for contour diagram construction are given later. We will now examine the ability of this form to represent the primary characteristics and justify the bandwidth diagram given in figure 10.
Noise.

Contour diagrams can contain any amount of noise as this can be filtered out at a later stage.

Modularity.

If the text is non-modular then this will be reflected in a linear set of contours. However, when modularity occurs these are reflected by a more complex pattern of contours which show the inter-relationships within that module as well as the depth of the information. Thus, non-modular text can be represented, whilst modular text is represented in an advantageous manner.

Linkage.

The contouring effect can be used to show the linkage between modules of knowledge that are at the same level. The knowledge engineer must take care not to link unrelated sections in an ambiguous way. This problem can be partially overcome by partitioning the diagrams however this does not eradicate the problem and hence the upper bound is only 'intermediate'.
Operational types.

A contour diagram is flexible and can if necessary represent a significant number of types. These will include such types as decision, factual, rule, queries, directions etc. however, it is best to keep the number within a limit so as to make the rerepresentation process rigorous and feasible.

Sequencing.

Any degree of sequencing is tolerated on the diagram. The sequencing is represented by contours, these being one of the basic features of this type of representation.

Justification and explanation.

The representation can cope with varying levels of justification and explanation. This is due to the availability of suitable operational types along with the contouring which can relate them to the object being justified or explained.

4.1.6 Decision tables.

Let us consider the following example:

<table>
<thead>
<tr>
<th>Lights dim</th>
<th>T</th>
<th>T</th>
<th>F</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engine won't turn over</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>Battery problem</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Starter problem</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No problem</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

fig. 12.

This shows a small decision table. The form is highly diagnostic and is aimed at representing problems composed of complex decision procedures.

The bandwidth diagram for decision tables is as follows:
The reasons for the shape of the bandwidth will now be considered.

Noise.

Decision tables cannot represent noise.

Modularity.

The modularity of the knowledge cannot be represented here unless the modules are composed of conditionals in which case the table can be partitioned into areas, each one representing a module, this however would be a variant representation and demand an alternative bandwidth diagram.

Linkage.

Decision tables do not have any facility for representing the linkage between modules.

Operational types.

The decision table is able to represent actions and conditions.

Sequencing.

The sequencing of conditional statements can be represented.

Justification and explanation.

There is no facility to represent justifications or explanations within decision tables.
4.1.7 The trace matching process.

Having created a primary trace of the characteristics present within the elicited representation concerned, the knowledge engineer can select the most suitable primary representation by superimposing the trace on top of each bandwidth. As long as the trace from the elicited representation fits inside the bandwidth of the primary representation then the elicited knowledge could be represented in that primary representation.

For example, the following trace:

```
High
Significant
Intermediate
Low
Minimal
```

Fig. 14.

can be represented in both primary representation A *

```
High
Significant
Intermediate
Low
Minimal
```

Fig. 15.

and primary representation B:

*) These bandwidths have been constructed for illustration purposes only.
It can be seen that the bandwidth of primary representation B is much wider than that of A. This indicates that it is much more accommodating to matching traces. This however, is an indication of the fact that the wider the bandwidth the more general purpose the primary representation.

The following guidelines can be followed in selection of a primary representation.

1. Match the trace with the bandwidths.

2. Remove those diagrams where the trace does not fit. Exceptionally, if the trace is enclosed in a well fitting bandwidth apart from say one characteristic, then the possibility of building a variant of the standard primary representation could be considered.

3. Of those representations that the trace fits into, the one with the smallest bandwidth area should be selected.

4. If no bandwidths are suitable then the trace should be re-examined and if it is still found to be correct the possibility of creating a new, hybrid or variant primary representation should be considered.

4.1.8 Primary Analysis in Practice.

The primary analysis process described above: trace creation and trace matching, is based on the assumption that the elicited representation is either small enough so that the trace drawn is valid for the whole of it or that the elicited representation is uniform in character throughout its length. In developing real case studies these assumptions were found to be valid for some
elicited representations, noticeably some textbooks and some verbal reports which were short in length whilst also being uniform and standard in character. However, longer texts, reports and especially interview based verbal elicitation tend to be less uniform in context, being variable in characteristics over that length. To overcome this the following remedy is suggested: the text of the elicited representation should be partitioned into lengths that can be individually reasoned about and accurate traces drawn.

The length of text to be used as the basis for the trace should be left up to the knowledge engineer as this is domain dependent. The knowledge engineer will have to take factors such as the following into account when selecting the block size - volume of text in an average adjacency pair, total number of adjacency pairs, quality of information in the adjacency pairs etc. These factors are important as the knowledge engineer is trying to obtain blocks that hold an amount of information that can be reasoned about and have primary traces drawn for them. For example, it is of very little benefit to set the block size low if each adjacency pair individually contains very little information as this may not provide enough information to draw a useful trace from. In this situation it would be better to increase the size of the block - increasing the amount of information on which to draw in creating the trace. However, if the total representation is composed of many hundred adjacency pairs then the knowledge engineer may find difficulty in going through the entire text and splitting it into blocks. In this case we suggest two choices: one is to use a single page of A4 as the standard textual block. The second is to use a binary decomposition of the full text length, where for a large text it would first be divided into two - one for each half of the block which is partitioned down the middle. Then if more accuracy were required - i.e., it was felt that the blocks were still too large to have a trace characteristic of the text within them - the two blocks could then be partitioned themselves to make four blocks. This process could be theoretically repeated until say a trace was drawn for each adjacency pair in an interview transcript. (An adjacency pair is one knowledge engineer - domain expert interaction.)

The second approach discussed above was used in a case study where very long interviews were transcribed. Four hours of interview were broken down into four one hour sessions. One of these had for example 42 adjacency pairs and this was divided by two giving integer
values of 21 and 21 as block sizes. The process was then repeated giving four blocks of 10, 11, 10 and 11 adjacency pairs respectively. At this point the process was stopped as it was felt that these were workable block sizes with stable characteristics.

Diagrammatically this can be viewed as:

```
+-----+-----+
| 42  |
+-----+-----+
| 21  | 21  |
| 10  | 11  |
| 10  | 11  |
```

fig. 17.

Traces can then be drawn for each of these blocks. Each of these can then be matched with the bandwidth diagrams in order to produce the most appropriate primary representations.

Having more than one type of primary representation is not a problem as each block is treated individually, before the whole text is transformed ultimately into a unified domain specification. Blocks are also considered individually when the formulation of the representation specification is performed. The issue of integrating different primary traces in addressed in chapter six.

4.2. The transformation process.

4.2.1 Introduction.

Having created the primary trace that characterises the elicited representation and then having matched this with the bandwidth diagrams in order to select the most suitable primary representation, the next step is to transform the elicited representation into the selected primary representational form.
As shown by the diagram above, the transformation process is dependent upon two factors; firstly, the form of the primary representation selected during the matching process and secondly, the format of the elicited representation. Thus, there is a need for several different transformation processes, the most important of which will now be described.

4.2.2 Transformation technique one.

Elicited representation: Knowledge engineer - domain expert interview.

Primary representation: Contour diagram.

Introduction.

This section attempts to give guidelines on how to transform an elicited representation in the form of an interview into a contour diagram. We will present guidelines and their theoretical basis, these are supported where necessary by small examples.

A full case study that uses these guidelines to create a primary representation is given in chapter 7.

Analysing and re-representing interviews.
A: Partitioning the transcript.

The transcript is partitioned into a series of knowledge engineer - domain expert, question and answer interactions, each of which we will refer to as an 'adjacency pair'. The pairs are numbered and then marked off from each other.

The numbering of the pairs helps in referencing information and the dialogue can be seen as a series of points where a single question and an adjoining answer combine to make a conceptual point.

Having numbered the adjacency pairs, observations can be made regarding the referential flow within a domain through examining its information points. For example:

i) Points can only reference other points which temporally proceed them.

ii) Items usually refer to items at a similar or higher level before referencing a lower level item. For example, someone might say:

DE1: "... this method needs an arbitration procedure"
KE1: "... as in conflict resolution"
DE2: "... yes"

Here the knowledge engineer tries to clarify a point made by the domain expert in DE1 by asking if the point is similar to the concept of conflict resolution (which we will assume was discussed as an earlier concept) to which the domain expert gives an affirmative response DE2.

The knowledge engineer is more inclined to do this than to ask about an individual conflict resolution strategy which is at a lower level - an information point.

B: Analysing the transcript.

Having broken the elicited representation down into adjacency pairs these can then be analysed using the following fourteen guidelines in order to produce a contour diagram:

1. Decide upon a set of connectives to use in the decomposition process.
2. Compile a list of keywords to use in the decomposition process.
3. Decide upon a criteria for filtering noise from the text.
4. Remove noise from the text.
5. Examine the knowledge engineers text for conversational coherence and alignment (characteristics recognised in communication theory research to help describe verbal interaction - this will be discussed in section 4.2.3).

6. If necessary break the text into a series of smaller segments approximately a paragraph in size.

7. Decompose the paragraphs into sentences.

8. Take each line in turn and check it for referential characteristics of the form:


   and where necessary add text that helps to identify the reference.

9. Based upon the position of the keywords and connectives, partition the sentences up into subsentences - removing the connective in the process. Repeating the partitioning until the sentences are "singularities". (contains no connective or keyword) or have only connectives which help form a list of conditions or items. Whilst this is being performed a careful note of the nested structure of the decomposition must be made.

10. After the whole text has been decomposed, the numbering and indentation can be used to create a contour diagram.

11. Examine each section of the contour diagram and try and assign it a type:

   1. Noise.
   2. Sequencing information.
   3. Facts.
   4. Rules.
   5. Queries.
   7. Explanations.
   9. Qualification of other types.
Any sections not coherent or incapable of having a new type invented for them, should be classified as noise.

12. Remove noise.

13. Each section of the text in the contour diagram is then replaced by a more concise definition.

14. Take the distilled information and draw out as much detail as possible, listing all of the definitions for each type together.

Applying the guidelines.

i. The second and third guidelines tell the knowledge engineer to collect a set of connectives and keywords that can be used in order to help decompose the adjacency pairs. The connectives are of the form 'and', 'or', 'so', 'thus' etc., whilst the keywords are domain specific. For example, in a medical domain the keywords may be of the form: 'diagnosed', 'treatment', 'symptom'. The aim of these connectives and keywords is to have a basis of suitable points around which long sections of text can be broken down into smaller ones.

For example, the following text (taken from [Fox. 1985]):

1. The patient is a 50 year old adult who was said to have CML in 1978 and has now gone into blastic transformation and is said to be lymphoid by morphological and cytochemical criteria.

can be divided by the connective AND

1.1 The patient is a 50 year old adult who was said to have CML diagnosed in 1978
1.2 And is said to have gone into blastic transformation and is said to be lymphoid by morphological and cytochemical criteria.

thus simplifying the amount of text to be analysed in one go.

ii. The granularity of the text, information or knowledge are all factors that need to be taken into account when deciding upon what constitutes noise, as it may be different things in different domains.

iii. Conversational coherence will be covered in greater detail in section 4.2.3.
iv. If an adjacency pair contains long verbal texts from the knowledge engineer, the domain expert or both then these may need to be broken down into more manageable sizes to simplify analysis. In practice a paragraph (10 - 15 lines) is found to be a manageable size.

v. The paragraphs can first be decomposed into individual sentences and these worked upon, the whole process being one of structured decomposition.

vi. The decomposition of the sentences begins with a check for referential characteristics. The aim of this is to make the sentence complete and self contained, substituting into it the item or piece of knowledge referred to by the referential.

For example, in the following sentence:

"... it is also because of this factor that the ratio of steel to copper is four to one."

The referential characteristic is "this" which if it refers to, say, "low malleability" from the previous sentence we get the new sentence:

"... it is also because of the low malleability factor that the ratio of steel to copper is four to one."

vii. After the paragraphs have been contoured each of the sentences can then be typed. The types may be facts, rules, heuristics or any other type that classifies the sentence concerned. However, some sections will be without meaning or incapable of classification. These are then regarded as noise, which is subsequently removed.

viii. The typed sections are then made more rigorous and given concise definitions.

For example, the section:

In adult normal TdT positive cells are not more than 5%

becomes typed:

RULE: In adult normal TdT positive cells are not more than 5%
Then made more rigorous:

```
RULE: Adult = normal => TdT positive cells < or = 5%
```

ix. Finally, the typed definitions are collected together, this enables consultation and other adequacy checks to be performed.

4.2.3 Conversational coherence.

In order to obtain a clearer insight into the dialogue between the knowledge engineer and the domain expert it is necessary to draw from both ethnomethodology and from grounded theory.

"Ethnomethodology" is an area of linguistics where communication theorists use transcripts to examine conversational microdetails, whilst "grounded theory" recommends procedures through which categories emerge from the text during the process of descriptive analysis. The origin of both these research areas is that of conversational coherence. We can define conversational coherence by saying that: coherence refers to the fact that utterances produced by competent speakers in conversation are usually seen to be connected to each other in orderly and meaningful ways. Thus, research in conversational coherence seeks to discover ways of describing the level of coherence a conversation has.

Alignment in conversational coherence.

Conversational coherence makes the pre-supposition that the participants in a conversation talk in orderly, patterned, non random ways enabling them to define and make sense of their conversational situation. Researchers in communication theory [Jacobs. 1983] have found that one of the most important factors for the participants in being able to make sense of their situation, is the perception of their social identities within that conversation. For example, in a conversation where only one party, the knowledge engineer say, asks questions to which the other party, the domain expert, responds it is possible to assume that the relationship between the knowledge engineer and the domain expert is an asymmetrical one (i.e., one of the communicators is in a more dominant position than the other). The knowledge engineer asks and
the domain expert answers, and the domain expert responds to queries directly and concisely as a matter of course, as if this were the prescribed behaviour in this situation. The knowledge engineer and the domain expert are conforming to conventionalised role behaviour.

A technique developed for the examination of conversational coherence by the symbolic interactionists Stokes and Hewitt is that of "alignment" [Stokes, 1976] which refers to the alignment actions as verbal strategies that communicators can use to repair misunderstandings or descriptions in conversation. For example, when a knowledge engineer says: "what do you mean by that?" this can be thought of as aligning talk.

In analysing transcripts it is therefore useful to consider the role that alignment plays between the two participants.

Seven alignment strategies that may occur in knowledge engineer - domain expert interactions will now be outlined.

1. Accounts.

Accounts are statements offered to explain unanticipated or questionable behaviour. An account consists of a justification or an explanation for conduct deemed untoward within or outside the context of an interview.

2. Formulations.

Formulations are coded utterances that summarise previous utterances or offer interpretations - gists - of a conversation in progress.


This consists of items or utterances referring explicitly to the verbal properties of another message. Metatalk can focus on the speakers talk (e.g., "like I say") on the receivers talk (e.g., "you mean") or on the interview process itself (e.g., "could I ask you a few questions?").

Metatalk is one of the most frequently occurring alignment actions and as such further refinements of this strategy have been made.
3.1. Clarifying.

These are attempts to refine the meaning of a previously expressed message or to express the need for making that meaning clearer; rephrasing for the sake of clarity. For example, "I mean", "I assume you're talking about"; "You're talking about".

3.2. Remediating.

Attempts to express misunderstanding or mishearing or to provide apology, correction, or disclaimer for behaviour that might otherwise be seen as untoward. For example, "I misunderstood you on that", "It's going to sound funny", "excuse me".

3.3. Directing.

Messages that masquerade as asking permission to perform certain speech behaviours while functioning in fact to perform these behaviours. For example; "can I ask a question", "will allow me to make one point if I can", "let me interrupt you for just a moment".

3.4. Requesting.

Messages that make actual requests of the receiver or of his or her talk. For example, "What else can I respond to?", "May be you can describe that for me".

3.5. Agendizing.

Messages that may also function as requests but requests that specifically announce an agenda for the interview stage or process. For example: "If you have anything that you would like to ask me or wrap up with".

3.6. Side particles.

Messages referring to or qualifying prior or subsequent messages - often an idiosyncratic speech habit. For example, "I said", "you mentioned", "you know", "Know what I mean Harry"[F.Bruno Esq,...]
4. Side Sequences.

These are conversational sequences that constitute a meta communicative break in the conversation, after which the ongoing conversation resumed. These sequences are used to ensure consensus on a point. For example, Q: "do you follow that", R: "yes", Q: "sure", R: "yes that's fine".

5. Metacommunicative digressions.

These are conversational sequences similar to side sequences in that they are metacommunicative breaks in the conversation but they are more elaborated and less ritualised than side sequences. Generally metacommunicative digressions consist of extended sequences of metatalk where conversation turns to peripheral information.

6. Digressions.

These are conversational sequences where the conversation totally digresses from the topic under discussion.

7. Qualifiers.

Qualifiers are coded words and phrases that indicate tentativeness, uncertainty and nonassertiveness such that the issue is evaded or the opinion expressed is diluted. For example, words of the form: "kind of", "somewhat", "sort of".

Thus, when considering an elicited representation the descriptions given above of the argument strategies and meta talk will benefit the knowledge engineer's understanding of the composition of the text and help in its breakdown.

4.2.4 Comment on conversational coherence.

We find that the use of conversational coherence is a useful approach to take on categorising the type of interaction that an adjacency pair contains. This is important as firstly, the theoretical framework with which to reason about knowledge engineer/domain expert interaction is at present in its infancy and if we can draw upon a more established field of research this will allow us to consider our field in light of this work and adopt/amend any
theory we feel would assist our understanding and secondly, the categorisation of an interaction will allow, when more studies have been performed, an analysis of which techniques are most applicable in which situations and help us strengthen the guidelines used in the methodology.

4.2.5 Transformation Technique Two.

Elicited representation:

i) Concurrent vocalisation from domain expert or

ii) Text from a literary source.

Primary representation: Flow diagrams.

Introduction.

This section attempts to give guidelines on how to transform an elicited representation of text derived from concurrent vocalisation or a literary source into a flow diagram primary representation. A full case study that uses these guidelines to create a primary representation is given in chapter 6.

Analysing and re-representing text.

The following guidelines are a suggested way to produce flow diagrams from text. They are not intended to be rigid and unchangeable as they may need to be amended or added to depending upon the individual domain or situation.

1. Decide on any criteria that may be needed for the removal of noise.

2. Remove any noise.

3. Take a small segment of text and break this down into the components of the flow diagrams and give them types.

4. Construct a separate flow diagram for each of the textual points.

5. Unify the separate flow diagrams into one large diagram.

6. Examine the structures for repetitions and redundancies.
7. Examine the structures for deficiencies i.e., undefined paths from decision boxes, incom­
pleted actions upon diagnosis of problems.

8. Recheck elicited representations to improve the deficiencies.

Applying the guidelines.

1. As in the development of a contour diagram noise must be removed but what exactly
constitutes noise is still influenced by the individual situation and domain.

2. As texts may be large the knowledge engineer will when necessary find it useful to
break the text down into smaller more manageable segments. Techniques have
already been suggested for decomposing interviews and these can be adapted for literary
or report texts.

3. Once the text is broken down into a series of small or manageable segments these can
then be represented as a series of flow diagrams each detailing small areas of the
domain. The individual diagrams can then be combined into one or more larger
diagrams. Ensuring that each of the small flow diagrams is correct aids the adequacy of
the primary representation as a whole (adequacy is a collective term that describes a
representations ability to achieve levels of correctness, consistency, completeness etc.,
and is discussed in section 4.3).

4. A syntax for a suggested flow diagrammatic language is given in appendix B.3

4.2.6 Transformation technique three.

Elicited representation: Text from a literary source.

Primary representation: Decision table.

Introduction.

This section attempts to give guidelines on how to transform an elicited representa­
tion of text from a literary source into a decision table. The guidelines are intended
only to act as an indication that such a transformation can be carried out.
1. Decide on any criteria that may be needed for the removal of noise.

2. Remove any noise.

3. Take the text and define its constituent parts by their operational types.

4. Take the conditions and associated actions and place them into a decision table structure.

5. Rationalise the structure. This can be done through utilising "don't care" condition, "don't perform" actions and "else" structures where appropriate.

6. Consider the use of an extended entry format for the conditions and actions. This involves using an abridged natural language for the entries and has the benefit of heightening readability.

Other techniques.

This section has suggested three techniques that can be used to transform elicited representations into primary representations. These are not only the possible techniques that can be used and the methodology can be extended by the knowledge engineer who will have to be flexible in his approach, using existing transformation strategies, adapting techniques and inventing new ones to meet a given situation.

4.3. Adequacy of a representation.

"Adequacy" can be defined as "sufficient, proportionate to needs; satisfactory" [Oxford English Dictionary]. We use the term throughout the description of our development methodology as a generic description covering the following seven measures of a representation; which together compose its adequacy.

1. Completeness.

2. Consistency.

3. Versatility.

4. Accuracy.

5. Correctness.

7. Unambiguity.

The completeness of a representation is the degree to which the representation captures the original information. For example, if the domain consists of three facts then if the primary representation contains these facts with their original semantic meaning then that representation can be considered complete.

Consistency is important if an item is held more than once in a representation as the item must have the same semantics and representational form at each point.

In an environment where the representation is to cover many domains the importance of flexibility can not be stressed too strongly. A representation should be versatile, such that it can be adapted to suit a variety of domains. However a representation should not be 'stretched' to fit a domain it is not suitable for, nor should it be necessary to force a domain to fit a representation.

The accuracy of a representation is a measure of semantic correctness and is looked upon as an indication of whether the spirit of the original representation is being maintained. ‘Correctness of a representation’ is a measure of the representations internal syntactic and semantic correctness, as defined by the definition of the representation language.

The representation must be sensitive enough to cope with varying grain sizes of information and accommodate changes to the knowledge base at all levels. If the representation is insensitive then the 'balance' of the knowledge base will not accurately reflect the domain experts original intentions and meanings.

An important measure of adequacy is the clarity and unambiguity of a representation. Ideally the representation should be totally unambiguous; as in formal languages, the syntax and semantics of which are well defined, as opposed to the potential total ambiguity offered by natural language texts.

4.4 Comments and conclusions.

This chapter has discussed the processes involved in taking the elicited representation
and producing a primary representation of the elicited knowledge.

The chapter has three main themes running through it. Firstly, that a suitable primary representation be chosen and that this choice be capable of justification. Secondly, that the transformation of the knowledge from the elicited to the primary representation should prevent information loss or change in semantic meaning and thirdly, that the process be capable of application to real world problems.

The chapter showed that in order to select a suitable primary representation and justify the selection it is necessary to examine the characteristics that underly the elicited knowledge. When these have been isolated, representations can be produced that are amenable to these characteristics and provide suitable forms through which to represent the elicited knowledge. Progress in achieving this was made with the recognition of the role such factors as modularity, linkage, sequencing of knowledge etc., play in the organisation and definition of elicited knowledge. These factors were then incorporated into a representation selection process where traces were drawn showing their presence in the elicited knowledge. The traces were then matched against bandwidth diagram showing the representational capacity of the contending primary forms. The most suitable match determined the primary representation language.

The utilisation of primary representations showed the value that a more visual form of representation can have over a plain textual one whilst the increased organisational structures associated with the primary representations allowed some of the deficiencies and inconsistencies in the knowledge to be highlighted. However, further research into primary representations and their underlying characteristics is needed.

The chapter gives details to the knowledge engineer on how to utilise the theory in a real world problem and provides guidelines for creating traces, bandwidths, the selection of a primary representation and guidelines for transforming elicited representations into primary ones. These were based upon the experience of case studies but are not meant to be definitive and the knowledge engineer is encouraged to adapt them and create new ones where necessary.

Finally, the chapter discussed how the adequacy of a representation can be measured. This definition was useful to us in creating representations as the literature is weak in this area.
and it is important to have parameters against which the level of representational acceptability can be assessed.
CHAPTER 5.

TRANSFORMATION OF THE PRIMARY REPRESENTATION.

5.1. Introduction and overview.

The creation of a primary representation has significant benefits for the development process. These stem from the increased organisational structure of the primary representation over the elicited representation and from the justification procedures used to maintain the semantic integrity of the knowledge during the re-representation process.

Having created a primary representation that has semantically equivalent knowledge to the elicited representation but within a more structured form we are now in a position to analyse and examine the knowledge more closely.
Two processes are now performed on the primary representation: 1) A formal specification of the domain knowledge - the domain specification. 2) An analysis of the primary representation to determine a suitable "classical" representation to use as a representation language - the representation specification, e.g. frame based, semantic network, rule based, etc. The relationship of these two processes with the stages in the methodology is shown in figure 19.

The domain specification requires a formal notation and a discussion of the available notations is presented later in this chapter. We have used the "Z" specification language for domain specification, a decision justified in section 5.2. A specification in "Z" consists of formal text and natural language text. The formal text provides a precise specification, while the natural language is used to introduce and explain the formal parts. Specifications are developed via small pieces of mathematics that are built up using a formalism known as the schema language. This gives specifications added structure and leads to specifications that are more readable than those written purely in mathematics. More details of both "Z" and the schema notation are given in section 5.2.4.

The representation specification is a specification of the form identified as most suitable in which to represent the elicited knowledge, and is a description of its syntax/semantics. The representation specification is created by examining the knowledge types present in the primary representation. The amount of conceptual, factual, control, procedural and heuristic knowledge is plotted on a graph known as the 'knowledge profile'. This is then compared against profiles for some classical representations such as frames, semantic networks and rules - 'representation profiles'. These profiles show the capacity of the representations for handling the different knowledge types. The classical representation with the closest profile to that of the primary representation is then chosen to act as the basis of the representation specification.

After the domain and representation specifications have been created, the penultimate stage is to combine them into one form, in which the elicited knowledge from the domain specification is represented in the form advocated in the representation specification. The result is known as the secondary representation.
Having defined the secondary representation, the final step is to determine which control architecture would be applicable to the problem. The secondary representation and control architecture form the concrete specification which can act as a specification for an implementation. This is discussed in chapter 5.4.

In this chapter we will examine each of these processes in more detail and discuss their roles in the development methodology.

5.2. The domain specification.

5.2.1 Aim, role and perspective.

The primary representation provides us with a more rigorous form than the natural language text of the elicited representation with which to reason about the domain knowledge. However, the primary representation is still not rigorous enough to allow a detailed analysis of the elicited knowledge to be undertaken. This is because the primary representation may contain ambiguities, inconsistencies or incompleteness that would detract from the analysis but can not be spotted due to the limitations of the structures used.

The utilisation of a domain specification presents a new way in which to consider knowledge bases. The literature indicates that with the exception of the work of Levesque (Levesque, 1984), there has been little research done on the specification of knowledge bases within the a.i. community. We feel that three aspects of a formal approach are particularly advantageous to the development of a knowledge based system. Firstly, the domain specification is the most formalised state that the elicited knowledge reaches during the development cycle. Secondly, the domain specification can act as a medium for communication between the knowledge engineer and the domain expert as well as the knowledge engineer and the implementer. This is especially true of the specification language we have chosen for our domain specifications - "Z" - as "Z" consists of both formal text and natural language. The formal specification can be given to the implementer and the natural language text that informally describes the mathematics, can be given to the domain expert, thus the domain specification performs a dual role in communication. Thirdly, the specification can act as a basis for the future maintenance of the knowledge base.
The use of a formal language to specify the knowledge base enables the adequacy of that knowledge to be greater than it would be under any other circumstances. The aim is to enable an unambiguous specification to be built that is complete and consistent. It is complete in the sense that every item of information present in the elicited and primary representations is also in the domain specification and consistent in that if an item is referenced more than once then the reference mechanism and data types are identical.

The area of ambiguity between the knowledge engineer and domain expert in their interpretation of what the knowledge base states or is meant to state at any one stage in the development can be diminished by the introduction of a domain specification. This can then act as the authoritative unambiguous statement around which discussion can proceed. Similarly, the knowledge engineer and programmer can refer back to the domain specification if the implementer feels there to be any lack of clarity in the concrete specification.

The maintenance of the system also benefits from the use of a specification. For example, if a knowledge base were to be extended by, say, taking another elicited representation and developing it through the methodology there would be the opportunity to add the new domain specification to the existing one and to check the adequacy of the resultant specification prior to upgrading the concrete specification and ultimately the implementation. This would prove to be a major advantage as the current generation of expert systems have knowledge bases that are often so complex and undocumented that updating or changing them can prove to be a difficult task. For example, Clancy describes MYCIN [Shortliffe. 1976] in the following way:

"Production rules are a popular representation for encoding heuristic knowledge in programs for scientific and medical problem solving. However, experience with one of these programs, EMYCIN, indicates that the representation has serious limitations: people other than the original rule authors find it difficult to modify the rule set, and the rules are unsuitable for use in other settings such as application to teaching. These problems are rooted in fundamental limitations in MYCIN's original rule representation: The view that expert knowledge can be encoded as a uniform, weakly structured set of if/then associations is found to be wanting". [Clancy. 1983]

Similarly McDermott tells of DEC having set up a team of twelve computer scientists, specifically to maintain and develop the R1 system. [McDermott. 1982].
5.2.2 A review of specification languages.

The domain specification has the following relationship in the design methodology:

![Diagram showing the relationship between primary representation, formal language, domain specification, secondary representation, control architecture, and implementation.]

Knowledge is taken from the primary representation and formalised into a domain specification. The aim of the diagram above is to show that the language used within the domain specification does not have to be fixed. The syntax and semantics of the specification language can be "plugged in" at the knowledge engineer's discretion and utilised to write the domain specification. The language may be changed to meet a particular situation or because of the knowledge engineers' familiarity with a particular, yet suitable language.

There are currently three types of specification languages available. There are the logic programming languages including Prolog [Clocksin, 1981], ML [Gordon, 1979b] and PRO [Hehner, 1984]. There are the implementable functional specification languages such as LISPKit [Henderson, 1983], Miranda [Turner, 1985a] and KRC [Turner, 1982]. Thirdly, there are the formal languages which include VDM [Jones, 1980], Z [Sufrin, 1984] and CSP [Hoare, 1983].
Logic specifications.

The use of logic as a means of specification is well known to computer scientists and is documented in such seminal papers as [McCarthy 1962], [Floyd 1967], [Hoare 1969] and [Dijkstra 1976]. These papers gave a basis to the movement away from "trial and error" programming to the development of programs that can be proven to have the desired capabilities.

Thus, from the use of formal logic in specifying and proving programs correct it is but a short step to the use of a logic programming language as a specification language. This is advocated by Leibrandt and Schnapp who make the following statement about the language Prolog:

"Prolog, although originally conceived for artificial intelligence applications, can be used as a formal specification language for software systems because its underlying semantics provide for a very high degree of abstraction." [Leibrandt. 1983]

Other logic languages such as PRO can also be used as mediums for specification.

A keen proponent of Prolog and logic programming as a vehicle for programming and specification is Kowalski whose paper entitled "The relation between logic programming and logic specification" is an interesting discussion on the subject [Kowalski. 1985]

Prolog and many other logic programming languages can be thought of as implementable specifications. An implementable specification is midway along a line which has ‘abstract specification languages’ at one end and ‘general purpose implementation’ languages at the other.

Abstract specification languages  <-------------|------------------>  General purpose implementation languages

The use of implementable specification languages within the a.i. community tends to be by developers of systems who wish to implement rapidly some of the main features and functions of that system; this is more widely known as 'prototyping' [Floyd. 1983]. Using implementable specifications in which to prototype a system has both advantages and disadvantages. It allows the developer to test the feasibility of a proposed system in a minimum of time using a low level of resources. This is because the systems are developed in a 'run-debug-edit' or 'construct and test' [Newell. 1972] environment, which allows less
than fully developed ideas to be expanded and implemented. It can be argued that this is a
creative environment in which to develop a system but it is important that the developer
must not consider this as a formal or rigorous development style. The developer must also
be aware of, and resist, the temptation to perform what Kowalski describes as 'the trial
and error approach' to software development [Kowalski. 1983], the exact state we are try­
ing to avoid.

Functional specifications.

A further class of implementable specifications are those based on functional program­
mimg languages.

The first major list programming language was developed in 1958 by McCarthy and was
called Lisp [McCarthy. 1965]. Lisp can also claim another first for being the original func­
tional or applicative programming language.

The use of functional programming languages is very appealing to the computer scientist
as they have several advantageous properties that are lacking in conventional imperative
languages.

A functional program, or as it is sometimes termed 'script', is a series of recursive
equations that are based on the mathematical idea of a function f, where for a given input x
to that function, the output f x is always the same. The equations also have the property of
'referential transparency' [Russell. 1910], [Quine. 1960] where there is no concept of a 'vari­
able' as used in the assignment statements of imperative languages. A further property of
the functions is that if two or more expressions derive the same values for the same input
then they are equivalent and possess the property of extensionality.

A language that has these properties is MIRANDA which is described by its origina­
tor David Turner as:

"such a language may be regarded as lying somewhere on the boundary
between programming languages and specification languages. Com­
pared with a conventional programming language it is often rather concise,
and shares with ordinary mathematics the property which programming
languages do not normally possess, of being static i.e., equations do not
change their truth values over time". [Turner. 1985b]
The conciseness Turner refers to can be seen in his executable specification of Hoare's quicksort algorithm [Hoare. 1962]

\[
\text{sort } [ ] = [ ]
\]

\[
\text{sort } (a:x) = \text{sort } [b <- x \mid b <= a] ++ \text{sort } [b <- x \mid b > a]
\]

This should be compared to Wirth's implementation of it in Pascal which takes over thirty lines [Wirth. 1976]

Functional specifications have several further properties, each of which stems from the mathematical background of the languages. For example, consider the following function:

\[
\text{reverse } (x ++ y) = \text{reverse } y ++ \text{reverse } x
\]

This states that when we have two lists \( x \) and \( y \) then if list \( x \) is reversed and appended to the reverse of list \( y \) then this will give the same result as appending \( y \) to \( x \) and reversing the combined list.

We can inductively prove this to be correct.

\[
\text{reverse } (x ++ y) = \text{reverse } y ++ \text{reverse } x
\]

Definitions:

\[
\text{reverse } [ ] = [ ] \quad \text{... rev 1.}
\]

\[
\text{reverse } (a:x) = \text{reverse } x ++ [a] \quad \text{... rev 2.}
\]

\[
[ ] ++ y = y \quad \text{... def 1.}
\]

\[
(a:x) ++ y = a:(x ++ y) \quad \text{... def 2.}
\]

\[
y ++ [ ] = y \quad \text{... def 3.}
\]

Base case let \( x = [ ] \)

\[
\text{reverse } (x ++ y) = \text{reverse } y ++ \text{reverse } x
\]

\[
\text{reverse } ([ ] ++ y) = \text{reverse } y ++ \text{reverse } [ ] \quad \text{... rev 1.}
\]

\[
\text{reverse } ([ ] ++ y) = \text{reverse } y ++ [ ] \quad \text{... def 1.}
\]

\[
\text{reverse } y = \text{reverse } y ++ [ ] \quad \text{... def 3.}
\]

\[
\text{reverse } y = \text{reverse } y
\]

Inductive Hypothesis

\[
\text{reverse } (x ++ y) ++ a = \text{reverse } y ++ \text{reverse } x ++ a
\]

-91-
Inductive step

\[
\text{reverse} \ ((a:x) ++ y) = \text{reverse} \ (a : (x ++ y))
\]

\[= \text{reverse} \ (x ++ y) ++ a \quad \ldots \text{rev 2.}\]

\[= \text{reverse} \ y ++ \text{reverse} \ x ++ [a] \quad \ldots \text{I.H.}\]

\[= \text{reverse} \ y ++ \text{reverse} \ (a:x) \quad \ldots \text{rev 2.}\]

Q.E.D

A second important property of functional specifications is that through the use of program transformation techniques the specifications can be made optimal regarding efficiency. For example, suppose we had the following specification: "x and y are two lists each of distinct numbers arranged in strictly increasing order. The function 'marry' produces a sorted list of these numbers on one list but not the other". This can be defined by:

\[\text{marry} \ x \ y = \text{sort} \ ((x -- y) ++ (y -- x))\]

which is quadratic in running time. This can now be transformed into a more efficient implementable specification:

\[\text{marry} \ x \ y = \text{sort} \ ((x -- y) ++ (y -- x))\]

\[\text{marry} \ x \ [\ ] = x\]

\[\text{marry} \ [\ ] \ y = y\]

\[\text{marry} \ (a:x) \ (b:y) = \text{sort} \ (((a:x) -- (b:y)) ++ ((b:y) -- (a:x)))\]

There are three cases.

Case 1. \(a = b\)

\[\text{marry} \ (a:x)(b:y) = \text{sort} \ ((x -- y) ++ (y -- x))\]

\[= \text{marry} \ x \ y\]

Case 2. \(a < b\)

\[\text{marry} \ (a:x)(b:y) = \text{sort} \ (((a:x) -- (b:y)) ++ ((b:y) -- (a:x)))\]

\[= \text{sort} \ (a : (x -- (b:y)) ++ ((b:y) -- x))\]

\[= a : \text{sort} \ (x -- (b:y)) ++ ((b:y) -- x)\]

\[= a : \text{marry} \ x \ (b:y)\]

Case 3. \(a > b\)

\[\text{marry} \ (a:x)(b:y) = \text{sort} \ (((a:x) -- (b:y)) ++ ((b:y) -- (a:x)))\]

\[= \text{sort} \ (b : ((a:x) -- y) ++ (y -- (a:x)))\]

\[= b : \text{sort} \ ((a:x) -- y) ++ (y -- (a:x)))\]
Therefore the definition of marry without sort is:

\[
\text{marry } x \ [ \ ] = x \\
\text{marry } [ \ ] y = [ \ ] \\
\text{marry } (a;x) (b;y) = \text{marry } x y, a = b \\
= a : (\text{marry } x (b;y)), a < b \\
= b : (\text{marry } (a;x) y), a > b
\]

Which is of linear order time.

Thus it can be seen that functional applicative languages are powerful yet flexible forms, through which domains can be specified. A discussion of functional languages as executable specifications is given by Turner in [Turner. 1984] and [Turner. 1985] whilst Darlington discusses program transformation techniques [Darlington. 1982].

**Formal specifications.**

The third type of specification is that of the formal approach and is characterised by such languages as VDM and "Z". These languages attempt to give a mathematical framework around which specifications can be developed.

A formal specification is a declaration of what the system is required to do and not an algorithm of how to do that task. Thus, formal specifications are not themselves executable, but there is currently much research underway to develop techniques that will enable specifications, through a series of formal (hence provable) 'data development' steps to be turned into more concrete and ultimately executable forms.

The underlying mathematical principles for many of the formal languages are the same, being based upon abstract data typing, set theory and predicate logic. However, the 'visual notation' used is as important to the knowledge engineer as the mathematics behind the language, for it is the 'readability' of the language that becomes more important as the complexity of the specification increases. For example, a specification in VDM [Jones. 1980] of the operation to square a natural number has the form:

\[
\text{square: Int } \rightarrow \text{ Int} \\
\text{pre-square: True} \\
\text{post-square } (x,r) = r = x \times x
\]
while in "Z" the same operation is specified as follows:

\[
\text{square: } \mathbb{N} \rightarrow \mathbb{N} \\
\forall n: \mathbb{N}, \text{ square } n = n \times n
\]

The "Z" specification can also be written in a horizontal form:

\[
\text{square: } \mathbb{N} \rightarrow \mathbb{N} \mid \forall n: \mathbb{N}, \text{ square } n = n \times n
\]

Which is more space efficient. However, it is the strength of the visual notation due to the vertical form and simplicity of its constructs that differentiates "Z" from other specification languages.

Further examples of "Z"'s visual strengths and the ability to abstract the specifications will be given later.

In the past, criticism has been aimed at formal notations for being only applicable to small "toy" examples. However, in recent years much research has been carried out to alleviate these criticisms and the VDM, OBJ and "Z" notations amongst others, have been used in documented industrial scale projects [Hayes, 1985], [Morgan, 1984], [Cottam, 1984]. One of the largest of these projects is the formal specification of IBM's CICS system in "Z" at IBM Winchester [Hayes, 1985b].

5.2.3 Selection of a specification language.

In examining logical languages as a form to use in our domain specifications, we felt that the available notations were not flexible enough to cover the variety of situations and domains that would need to be specified. This is because the syntax and semantics of the logic languages are primarily intended for implementing, rather than specifying, algorithms. A specification is a description of what a system is meant to do in terms humans can understand, whereas an algorithm is a description of that system specification intended for computer, rather than human interpretation.

A secondary reason for not using an executable form of specification is that it removes any temptation a knowledge engineer may have towards "trial and error" programming -
prototyping.

A third influencing factor in our decision not to use a logic programming language for the domain specifications was the practical experience of Leibrants who used Prolog to specify a personal data base system of which he states:

"specifying and modelling the deeper levels of the system got increasingly more difficult to keep the specification crisp and consistent and avoid an ever growing collection of ad-hoc procedures useable at one and only one place" [Leibrant. 1983].

Similar arguments can be applied to executable specifications. They too are constrictive in the type of specifications that are natural and fit the functional notation. Turner states that:

"A functional language when considered as a specification language suffers from the restrictions inherent in being recursive: only computable functions can be denoted, so there are some useful and interesting specifications that can not be expressed within it" [Turner. 1985b]

The implication of Turner's statement is that the functional notation is more suited to specifying those domains which have an algorithmic content, rather than the often non-mathematical applied domains which must be specified for knowledge based systems.

Having examined the specification languages available, we felt that the best way to utilise the various approaches would be to allow the knowledge engineer to select the notation he felt best suited the domain. This would allow for a flexible approach to creating a domain specification. This freedom is indicated by the formal language input in figure 20. However, in our experience the most suitable approach for general usage is one based upon the formal notations, with "Z" in particular being highly suitable.

The nomination of "Z" as the most suitable notation was made for several reasons. Firstly, as the notation is to be used by knowledge engineers who may not be intimately familiar with formal specification languages, "Z" offers a highly visual option in its vertical "box" form. It also helps reduce some of the heavy syntactic requirements found in other formal languages. This does not however reduce its formality as it is a very well documented and highly defined notation with both a formal syntax [Sufrin. 1984] and denotational semantics [Spivey. 1984] constructed.

A second reason is a consequence of the 'schema' notation [Morgan. 1984b] which
gives "Z" the ability to abstract away heavy mathematical specifications, allowing the person specifying the system to concentrate on only those parts of the specification currently relevant (we will give an example of this in the following section).

Thirdly, the mixture of natural language text and mathematics allows the specification to be accessible to both the domain expert and the knowledge engineer/implmentor, alleviating the need for any special training on the part of the domain expert to read and understand the specification. When the domain expert can read the specification this can encourage high degrees of consistency, completeness, correctness and other measures of adequacy.

The future acceptability of "Z" is heightened as research is currently underway to develop software tools that will help the developers of "Z" specifications.

5.2.4 An introduction to "Z".

The "Z" notation was originally proposed by J.R. Abrial in 1979 but has since been developed by researchers at The Programming Research Group, Oxford University.

"Z" is composed of two complementary formal languages - a mathematical language and a schema language. The mathematical language is based on set theory with strong typing in the schema language giving a structured framework to aid in the development and presentation of large scale specifications. It is this latter feature which makes "Z" different from other formal languages based on set theory, as languages such as VDM do not have this structuring feature.

The basic syntax of "Z" includes the following definitions:

```
syntax:: definition:: := axiomatic
                      syntactic
                      generic

axiomatic:: := _______________
             |_____________________
             | signature
             | _______________
             | predicate
             | _______________

signature:: := decl | decl;signature
```

-96-
The basic syntax allows us to define such things as:

1. \[ \lambda x: N \rightarrow N \]
   \[ \forall x: N, \text{quad } x = 2x^2 + x + 4 \]

2. \[ \lambda \text{angle_of_slope:PN} \]
   \[ 0 < \text{angle_of_slope} < 45 \]

3. \[ \lambda \text{primes:PN} \]
   \[ \text{primes} = \{ n: N \mid \text{divisors } n = \{ 1, n \} \} \]

Example 1) can be interpreted as 'the symbol quad denotes a function which maps
numbers to numbers. Every number maps under `quad` to the sum of twice the square of the number, plus the number itself plus four'. The second example (2) can be considered as 'the symbol `angle_of_slope` denotes a constant of the type number that is between zero and forty five degrees'. The third example (3) can be read as 'the symbol `primes` denotes a constant that is the set of all numbers divisible only by themselves and one'.

The three examples given above show how the notation can be used to define simple operations. However, the primitive operators between types used in these examples are not powerful enough to enable more complex specifications to be constructed. In order to overcome this the generic types can be constructed from primitive types, the specification of these generic types being done in "Z" itself.

A summary of common operators is given in Appendix A.2.

The second half of "Z" is the schema language where a schema is composed of a signature and a predicate, either of which may be empty.

A small example will now be given, describing a transport system: We first define four wheel vehicles:

```
4W_Vehicles
| cars, trucks: P Vehicles
| cars ∈ trucks
| trucks ∈ cars
```

Then two wheel vehicles are defined:

```
2W_Vehicles
| scooter,
| push_bike,
| motor_bike: P vehicles
| push_bike ∈ motor_bike
| scooter ∈ motor_bike
| motor_bike ∩ push_bike = {} 
```
These can then be combined by what is known as schema inclusion into a super
schema:

```
| 4W_vehicles
| 2W_vehicles
```

Parts of this can be expanded if needed, for example we can expand the
2W_vehicle schema:

```
| 4W_vehicles,
| scooter,
| push_bike,
| motor_bike: P vehicles
|
| push_bike \in motor_bike
| scooter \subset motor_bike
| motor_bike \cap push_bike = {}
```

or the 4W vehicle schema:

```
| 2W_Vehicles
| cars, trucks: P Vehicles
|
| cars \in trucks
| trucks \in cars
```

or both

```
| cars, trucks: P Vehicles
| scooter,
| push_bike,
| motor_bike: P vehicles
|
| cars \in trucks
| trucks \in cars
| push_bike \in motor_bike
| scooter \subset motor_bike
| motor_bike \cap push_bike = {}
```
Thus it can be seen that the schema language is a useful structuring device which when combined with further operators and the mathematical language make "Z" a powerful yet still versatile language with which to specify systems.

5.3. The representation specification.

5.3.1 Aim, role and perspective.

The representations used in expert systems have been of the form: frames, rules, semantic networks, procedures etc., but more than any other, the rule representation formalism has been utilised in knowledge base construction. This has been the case to such an extent that expert systems have also become known as 'rule based systems', whilst many expert system shells have adopted a rule based representation and control architecture. As a consequence of this emphasis on production systems, knowledge engineers are tempted to think purely in terms of rules - for example asking the domain expert "how can I represent this information in a rule?".

A second factor that has influenced the knowledge engineer to consider primarily or only a rule based approach to knowledge representation is the strong emphasis that has been placed by some sections of the a.i. community on logic programming as the most suitable form in which to represent information. The representation primarily takes the rule based form - a form which can be relatively easily expressed in a logic language. For example Kowalski states:

"There is only one language for representing information - whether declarative or procedural - and that is first order predicate logic. There is only one intelligent way to process information - that is by applying inductive inference methods", [Kowalski, 1980]:

We do not feel however, that the rule based approach should be used regardless of suitability for the domain knowledge nor that it is the only representation available to the knowledge engineer.

The philosophy of the development methodology described here follows more from Sloman who states:

-100-
"Logic is not all embracing, it may be best for the largest range of uses without being best for everything" [Sloman. 1984]:

The philosophy we have adopted is to utilise different formalisms in different situations and for different purposes. Sloman tells of the consequences of adopting this philosophy:

"if it is true that different sorts of representations and inference procedures should be used for different sorts of tasks, then the implications for expert system design are profound" [Sloman. 1984]

5.3.2 Selecting a representation specification.

Overview.

This section aims to describe the process of selecting the most suitable classical representation around which to develop the representation specification.

The following diagram can be used to aid explanation:

![Diagram](image-url)

**fig. 21.**

The primary representation, produced in the primary representation phase, is firstly analysed for the presence of fundamental knowledge types, the levels of which are plotted on a graph. This is known as a "knowledge profile". The profile is then compared to the "representation profiles" of the classical representations. These representation profiles show the capacity of standard classical forms for representing the same knowledge types. The profiles with the best match are then selected and the corresponding classical model becomes the
representation specification. The model is a syntax/semantics for the form.

These stages can now be considered in greater detail.

5.3.3 Creating a knowledge profile.

Introduction.

The aim of this section is to examine the process of creating a domain profile from the primary representation. We will first of all consider what are the fundamental knowledge types upon which the profiles are based, and this will be followed by a detailed examination of these knowledge types.

The principal knowledge types.

In our initial survey of knowledge, 57 different types were found in the literature. We then performed a series of 'experimental' case studies and found that domains were principally composed of five knowledge types: factual; control; heuristic; procedural; and conceptual with other knowledge types composing a minority of the total.

This then raises the question - what are the attributes of a principal knowledge type and how is it possible to assess how much of a particular type is present within the domain.

5.3.4 A theory of knowledge types.

Overview.

The principal knowledge types can be considered as follows: The fundamental knowledge type - factual knowledge, is classified as such because it can only be decomposed into data items which alone can not form a knowledge type. It is termed fundamental because it is factual knowledge that acts as the basis for the higher order knowledge types - 'compound' and 'complex'.

A compound knowledge type uses the fundamental knowledge type in conjunction with operators in order to construct its compound types such as 'control' and 'heuristic'.

Finally there are the 'complex' knowledge types - which includes 'procedural' and 'conceptual' knowledge. These are so called because they are composed of other principal...
knowledge types including compound types.

**FACTUAL KNOWLEDGE.**

1. Statements.

The mathematical concept of a statement can be defined as:

"The fundamental property of a statement is that it is either true or false but not both" [Lipschutz. 1964]

Thus, it follows that facts can be defined in similar terms:

"Facts are premises which can be shown as either true or false"

We can say therefore that the following expressions are statements as the first expression is false and the second is true:

i. Paris is in England. ii. 2 + 2 = 4

By contrast the following expressions are not statements since neither can be assigned the logical value true or false:

iii. Where are you going? iv. Put the book on the shelf.

Thus factual knowledge can be defined in terms of statements.

2. Negation.

During the course of manipulating knowledge it is useful to have the ability to negate facts and so the negation of statements can be included in the description of factual knowledge.

Given any statement ‘p’ another statement called the negation of ‘p’ can be formed by writing "it is false that....." before ‘p’ or by inserting the word "not". Symbolically:

\[ \neg p \]

denotes the negation of p.

Theorem: "If p is true, then \( \neg p \) is false; if p is false, then \( \neg p \) is true."
3. Compound statements.

It is possible to have factual knowledge which is composed of several statements. These statements are joined by logical connectives and are known as compound statements. For example:

i) London is in England ∧ Houses of Parliament are in London
ii) London is in England ∧ Paris is not in England

are both compound statements.

Compound statements have two forms; the conjunctive and disjunctive, neither of which allows conditionals to be defined.

i) Conjunction: Any two statements can be combined by the word "and" to form a compound statement called the conjunction of the original two statements.

Symbolically: \( p \land q \) denotes the conjunction of the statement \( p \) and \( q \).

Theorem: "If \( p \) is true and \( q \) is true then \( p \land q \) is true; otherwise \( p \land q \) is false"

ii) Disjunction: Any two statements can be combined by the word "or" to form a new statement which is called the disjunction of the original two statements.

Symbolically: \( p \lor q \) denotes the disjunction of the statements \( p \) and \( q \).

Theorem: "If \( p \) is true or \( q \) is true or both \( p \) and \( q \) are true, then \( p \lor q \) is true; otherwise \( p \lor q \) is false".

Note: The logical \( \lor \) refers to the "\( p \) or \( q \) or both" usage of \( \lor \) as opposed to its "exclusive disjunctive form".
COMPOUND KNOWLEDGE TYPES.

Control knowledge.

Many statements are of the form "If p then q" and are generally known as "rules", which can be more formally denoted by:

\[ p \Rightarrow q \]

Where 'p' is the conditional and 'q' the resultant. An example of this type of statement is:

colour of paper = red \( \Rightarrow \) solution = acid

Thus a rule can be thought of as a relation between facts.

However, control knowledge is more complex than being purely a collection of rules and merits further examination.

A correctness obligation for conditional statements.

A conditional statement of the form \( p \Rightarrow q \) can have the logical definition:

"The conditional statement \( p \Rightarrow q \) is true except in the case that \( p \) is true and \( q \) is false"

This can be represented by the truth table:

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>( p \Rightarrow q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

and illustrated by the following examples:

1. If Paris is in France then \( 2+2=4 \)
2. If Paris is in France then \( 2+2=5 \)
3. If Paris is in England then \( 2+2=4 \)
4. If Paris is in England then \( 2+2=5 \)

This shows that logically only example 2 is defined as false, even though two of the other conditional statements contain either a false condition, false resultant or both. We therefore need to state if it is acceptable to allow false information to be present within a conditional statement. This is performed by defining a 'correctness obligation' for conditional statements. First,
we need to define the term ‘correctness obligation’.

"The correctness obligation of a statement is the acceptability of a given set of truth values for that statement."

We can now define our correctness obligation for conditional statements to be: "the value 'true' should not be capable of generation from conditional statements that contain the value 'false' for either the conditional, the resultant or both the conditional and resultant".

When applied to our logical definition this gives:

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p \Rightarrow q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Argumental knowledge.

The definition of the conditional given above in its restricted form is the basis of a knowledge type known as 'argumental' knowledge.

An argument [Lipschutz, 1964] is an assertion that a given set of propositions $p_1, p_2, \ldots, p_n$ called premises, yields another proposition $Q$, called the conclusion. Such an argument is denoted by:

$\text{Pl} \rightarrow \text{P2} \ldots \text{Pn |--- Q}$

The truth value of an argument is determined as follows:

" $\text{Pl}, \text{P2}, \ldots, \text{Pn |--- Q}$ is true if $Q$ is true whenever all of the premises $\text{Pl}, \text{P2}, \ldots, \text{Pn}$ are true otherwise the argument is false."

Thus, if control knowledge is considered to be based upon the conditional statement then it can also be termed as argumental knowledge.

Argumental knowledge can be considered in several ways, based upon the interpretation of "truth". We therefore propose that argumental knowledge be divided into "total" and "partial" categories and it is these that will be examined next.

*) Argumental knowledge has no intended connection with the term 'argument' as used in connection with parameter passing in programming languages.
Total argumental knowledge.

Total argumental knowledge is argumental knowledge where not only is Q true whenever all the premises are true, but also that Q can be proved (shown) correct from the premises. Total argumental knowledge is therefore the type classification of rules.

The following are examples of total argumental knowledge:

i) Litmus paper = Red $\Rightarrow$ Solution = Acid  
   ii) Litmus paper = Blue $\Rightarrow$ Solution = Alkaline

The left hand side and right hand side are both facts and the right hand side can be shown to follow rigorously from the left hand side.

Partial argumental knowledge.

Partial argumental knowledge is knowledge which is sufficiently correct to be argumental, in that all the premises can be shown as true and the consequence statement can also be shown true but where neither set of statements can rigorously show the other to be true. Thus, partial argumental knowledge is the type classification of heuristic knowledge and is technically a subclass of control knowledge.

The following are examples of partial argumental knowledge:

i) Sky = Red & Time = Evening $\Rightarrow$ Weather forecast = Good  
   ii) Sky = Red & Time = Morning $\Rightarrow$ Weather forecast = Bad

where the right hand side and left hand side are both facts but where the right hand side can not be rigorously shown to follow from the left hand side. We can consider these as heuristics.

COMPLEX KNOWLEDGE TYPES.

Procedural knowledge.

Procedural knowledge is the first of our complex knowledge types and is a collection of knowledge whose focus is the achievement of a goal. This procedure is often a sequence of operations specific to one area of the domain that can be utilised when needed. The sequence of operations within a procedure may involve factual and/or control knowledge in order to perform a sequence of operations to take that procedure from its initial state to its
goal state. It is possible that a domain may contain many areas of procedural knowledge.

Conceptual knowledge.

Conceptual knowledge can be defined as knowledge about 'mental conceptions or concepts' [Oxford English Dictionary] where a concept is an 'idea of a class of objects or general notion' [O.E.D]. For example 'the concept of evolution' or when an invention is discussed it is often in terms such as: 'a new concept in motor design'.

In elicited text, domain experts often talk in these terms. They also vary the generality when describing an idea i.e., the concept of 'man's evolution from the primates' is a concept but at a lower level of generality than 'the concept of evolution'. Thus, the knowledge engineer can consider that conceptual knowledge is both a measure of the level of abstraction the knowledge has and the meta knowledge associated with these levels of abstraction.

In order to reason at, and about these different levels of abstraction we need to have knowledge about the knowledge which defines these levels - this is termed 'meta knowledge' which Barr [Barr, 1981] describes as follows:

"We often know about the extent and origin of our knowledge of a particular subject, about the reliability of certain information or relative importance of specific facts about the world"[Barr. 1981]

Aiello also acknowledges the important tie that exists between meta knowledge and the notion of abstracting knowledge:

"The integration of knowledge and meta knowledge seems to be a very powerful abstraction mechanism that allows us to cope with a variety of representational problems, thus providing an alternative to the development of special purpose languages" [Aiello, 1984]

It can be seen that conceptual knowledge is truly a complex type involving abstract, meta and other knowledge types. We therefore feel that it is an important characteristic of the domain knowledge to be examined.

5.3.5 The profile structure.

The knowledge profiles are drawn on graphs with the following form:
This has the same vertical scale as the trace and bandwidth graphs used in primary analysis (Chapter 4.1). The X-axis above shows the full breakdown of the principal knowledge types, but in practice the axis is simplified to read "factual, control, heuristic, procedural, conceptual".

5.3.6 Profile matching - standard profiles.

Introduction.

Having obtained a knowledge profile for the elicited knowledge, the next step is to match this with the representation profiles in order to select an appropriate form to use as the representation specification.

Representation profiles.

The profile of a representation shows the ability of that form to represent certain types of knowledge - in this case the principal knowledge types.

The representations upon which the profiles are based are a set of fixed standardised classical forms. The aim of this is to reach an acceptable compromise - typical of their style of representation yet not too specific or specialised. This enables them to be adapted,
should a profile match be less than exact, allowing a better representation specification to be produced.

Standard representation profiles.

In this section we will briefly review each of the representations we chose as our standards.

Standard rule representation.

The work used as the basis for our rule representation was that given in The Handbook of Artificial Intelligence Vol. I (pp190-199) [Barr. 1981], where a production system is defined as follows:

"A production system consists of three parts: (a) a rule base composed of a set of production rules; (b) a special, buffer-like structure, which we shall call the context; and (c) an interpreter, which controls the system's activity".

Of these three parts the representational structure is defined through the first of these, the production rules, as:

"A production rule is a statement cast in the form: 'If this condition holds, then this action is appropriate...' the IF part of the productions, called the condition part or left-hand side, states the conditions that must be present for the production to be applicable, and the THEN part, called the action part or right hand side, is the appropriate action to take".

It is upon this very simple definition of a production rule that we will base our profile.

Standard frame representation.

The standard frame representation refers to a frame based language developed by Plant [Plant. 1985]. The language is, like the production rule representation discussed above, not directly implementable. This allows the desired style of representation to be obtained without having to reach a compromise for the sake of implementability.

A frame can be defined as 'a means of providing a structure in order to allow new information and data to be interpreted in terms of that structure and existing information'.

The frame based language used as our standard has the capacity to define many different knowledge types and then combine these in an inheritance structure.
For example, we can define 'facts':

<table>
<thead>
<tr>
<th>beginner [fact]</th>
</tr>
</thead>
<tbody>
<tr>
<td>desc: Are you new to this operating system</td>
</tr>
<tr>
<td>expl: This is to determine the level of help necessary</td>
</tr>
<tr>
<td>query: Are you a beginner with UNIX?</td>
</tr>
</tbody>
</table>

or numbers

<table>
<thead>
<tr>
<th>student year [number]</th>
</tr>
</thead>
<tbody>
<tr>
<td>desc: year of study</td>
</tr>
<tr>
<td>expl: Tests to see which computers undergraduates can use</td>
</tr>
<tr>
<td>range: 1..4</td>
</tr>
<tr>
<td>query: What year are you in?</td>
</tr>
</tbody>
</table>

These knowledge types form the building blocks for the more powerful representational structures. For instance the frame:

getting started  

<table>
<thead>
<tr>
<th>Desc: This section enables people to get started with UNIX</th>
</tr>
</thead>
<tbody>
<tr>
<td>(not reg_user)? =&gt; ref reg_user_inst</td>
</tr>
<tr>
<td>(not reg_user)? \ (not log_in)? =&gt; ref log_in_inst</td>
</tr>
<tr>
<td>(reg_user)? \ (log_in) \ (cont)? =&gt; ref system_commands</td>
</tr>
</tbody>
</table>

presents a series of predicates that when answered causes the right hand side of the implication to be fired. This invokes another frame which inherits the information and continues the process. For example, if 'not reg_user' is instantiated then the next frame to be fired would be 'reg_user_inst':
and so the process would continue until a termination state was achieved.

The syntax of this language is given in Appendix B.1

Standard semantic network representation.

The literature on semantic networks is extensive, however there does not appear to be any consensus on what the definitive syntax and semantics for a network should be. As Levesque says:

"Typically, a semantic network is described using diagrams of directed labelled graphs with little or no indication as to what exactly the diagrams are intended to represent" [Levesque, 1979]

We have therefore taken what we feel to be a well defined formalisation of semantic networks as our standard representation - contained in the paper 'A procedural semantics for semantic networks' by Levesque and Mylopoulos.

In their approach Levesque and Mylopoulos develop a procedural semantics which involves the notation of system behaviour under certain operations. Their approach is to distribute the interpreter (the mechanism that manipulates the knowledge base in order to answer user queries) with each component of the semantic network in the form of 'programs' (they can be thought of as procedures). These programs are necessary in order to perform the operations defined on the component. Every component consequently acts independently and is responsible for its own behaviour under its defined operations.

The program structures are abstractly typed in terms of concepts and relationships, the programs providing the interpretation of operations on these types.

Levesque and Mylopoulos describe their utilisation of programs and typing by saying that:
"To accommodate clearly this notion of type within our representation, we use the terms 'class' and (binary) 'relation' to represent respectively 'concept type' and 'relationship type', reserving the terms 'node' and 'edge' for our diagrams. Particular concepts are represented by 'instances' of classes, while relationships between concepts are formulated as assertions. Finally, the term 'object' is used to denote any component of the semantic network ...

For example:

![Diagram](image)

fig. 23.

Here, PERSON and NUMBER are classes, AGE is a relation, the unlabelled edges indicate the type of the component, and the edges labelled domain and range can be considered diagrammatic notation". [Levesque, 1979]

Having defined these basic terms, the semantic network notation is then expanded with operations being defined on relations and classes.

Frame profile.

The following frame profile is based on the frame based language described above and given in Plant [Plant, 1985].
The factual knowledge is the primitive unit of the language and is the building block upon which the other types are created. Thus the ability to represent this knowledge type is high.

The language also has several frame structures for control and heuristic knowledge of a similar form to that for facts and consequently the levels are similarly high.

The ability to represent knowledge procedurally is assisted by the availability within the language of high level constructs that enable the knowledge to be abstracted several levels, whilst allowing the knowledge to be kept in a procedural form.

The ability to represent abstract concepts and hierarchical forms is low, a result of several factors. Firstly, the strong typing that is used in our standard representation does not give the ability to represent concepts to the same extent that a more flexible form of typing would allow. Secondly, the typing used on our frame based form does not permit the representation of meta knowledge and consequently this affects the ability to express conceptual knowledge.

Rule profile.

The following rule based profile is based on the production system model given by Barr and Feigenbaum [Feigenbaum. 1981] described above:
High
Significant
Intermediate
Low
Minimal

fact cont heur proc conc

fig. 25.

Here the factual value is minimal because facts can not be represented in this definition of a rule based system.*

Control knowledge in its purest form - that of total argumental knowledge - is high as the major source of total argumental knowledge is rules. These are the major components of production systems.

Heuristic knowledge (a subcategory of control knowledge) in the form of partial argumental knowledge is representable up to the intermediate level. The value is not higher because it is felt that above this limit the knowledge engineer may wish to use production rules in conjunction with uncertainty constraints and this would require a different language and a correspondingly different profile to that used here.

The level of representation for procedural knowledge representation is low as only one of the techniques that can be used to proceduralise the knowledge is available, this being the partitioning of the knowledge base. If a technique such as attaching procedures or frames to rules were added to the language this would change the profile.

Conceptual knowledge is intermediate, in that rules can be modularised and then linked to super antecedent conditions. This technique may be used repeatedly - the further that these super antecedents are from the rule the higher the amount of conceptual

*) When drawing profiles they are drawn directly from the x-axis as the absence of syntactic/semantic constructs in the representation language may make it impossible to represent certain knowledge types and produce a zero reading.
knowledge. The rule form of representation is also amenable to representing meta knowledge and this allows an intermediate amount of conceptual knowledge to be representable.

Several ways in which our structured language and profile could be adapted have been suggested. The language given in Appendix B.2 takes up one of these - that of increasing its ability to represent factual knowledge to a significant level. The high level has not been reached as the language does not have any facility to represent conjunctions and disjunctive forms.

**Semantic network profile.**

The following semantic network profile is based on "The procedural semantics for semantic networks" by Levesque and Myopolous [Levesque. 1979].

![Semantic network profile diagram](image)

*fig. 26.*

The basic building block of semantic network representation is the fact, thus its value is high.

Control knowledge is allowable due to a construct that allows conditionals to be simulated.

Heuristic knowledge is represented in the direct conditional form but in our standard representation no attachment of uncertainty to conditionals is allowed.

The ability of the standard semantic network to represent procedures is significant and its ability to represent concepts is high. These both stem from the representations' ability to partition itself and to represent different levels of abstraction on the same
5.3.7 The Matching Process.

Representation selection.

Having drawn the knowledge profile and examined the representation profiles, the selection of the most suitable representation can now be considered.

The first task to be performed is to overlay the knowledge profile on top of the primary representations.

For example: If the knowledge profile of "domain A" is:

![Graph showing knowledge profile](image1.png)

When overlaid on the classical representations this gives:

Domain A and frames:

![Graph showing overlay of knowledge profile on frames](image2.png)

Domain A and production system:
A selection of options is now open:

1. A match is found.
2. A partial match is found.
3. No match is found.

If a total match is found then the profile matching can terminate as the representation specification has selected itself.

If the partial match is found then one of two situations may have arisen.

1. One representation is a close match.
2. Two or more representations are close matches.

It is the latter of these two cases which has occurred in our example above. A Technique is
therefore necessary to discover which match is closest. We must first define two terms: 'positive deviance' and 'negative deviance'.

This can be illustrated by the use of the domain A and rule representation overlay:

![Diagram of deviance levels]

fig. 31.

This overlay has two perfect matches, those for the heuristic and procedural knowledge types. There are however mismatches for the factual, control and conceptual knowledge types.

The factual mismatch is an example of negative deviance which is where the representation does not have adequate facilities to capture fully the amount of a certain knowledge type that the domain contains.

The control and conceptual mismatches are examples of positive deviance - this being where the representation has the capacity to represent a certain form of knowledge but that knowledge type is not present in the domain to the full extent of the capacity for representing it.

Having defined these two terms we can now give some guidelines for an informal technique to help select the best matching profiles.

1. Overlay the representation profile and domain profile.
2. Count the deviances both positive and negative and place them in a table.
3. Add together all the positive deviations and give them a non numeric value.
4. Add together all the negative deviations and give them a non numeric value.
5. Examine the deviances for each knowledge type to determine the number of matches.

6. Unless all matches contain a 'high' deviance, those containing such a deviance should be removed from further consideration.

7. Select the representation with the lowest number of negative divergencies.

8. If two or more representations have the same number of divergencies then examine which of the divergencies are at the higher levels and eliminate that representation. Repeat this until only one representation is left.

9. If two or more positive representations are equal in all aspects then select the one with the lowest positive divergence factor.

10. If two or more representations are equal in all aspects then if after reconsideration of their profiles they are still equal then examine whether a change from the standard representations would aid the matching process.

11. If the negative and positive divergencies are great then investigation of new or hybrid representations may be appropriate.

We will now use these guidelines to select the most appropriate representation specification for our example.

step 1: Overlays - already performed.

steps 2, 3 & 4: By using the following scale to translate numeric into non-numeric values:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>≥ 5 units</td>
</tr>
<tr>
<td>Significant</td>
<td>4 units</td>
</tr>
<tr>
<td>Intermediary</td>
<td>3 units</td>
</tr>
<tr>
<td>Low</td>
<td>2 units</td>
</tr>
<tr>
<td>Minimal</td>
<td>1 units</td>
</tr>
<tr>
<td>Match</td>
<td>0 units</td>
</tr>
</tbody>
</table>

we achieve the following results:

frame match:

negative deviance = 0 = match
positive deviance = 7 = high

Rule match.
negative deviance  = 2 = low
positive deviance  = 3 = intermediate

Semantic network match.

negative deviance  = 0 = match
positive deviance  = 8 = high

the following table structure:

<table>
<thead>
<tr>
<th></th>
<th>n.d</th>
<th>p.d</th>
</tr>
</thead>
<tbody>
<tr>
<td>frame match:</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

This has no negative divergencies.

Production system match:

<table>
<thead>
<tr>
<th></th>
<th>n.d</th>
<th>p.d</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

This has two negative divergencies.

Semantic network match:

<table>
<thead>
<tr>
<th></th>
<th>n.d</th>
<th>p.d</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

This has no negative divergencies.

Thus the frame representation and the semantic network representations can be selected at this stage.

step 8. The frame representation has the lowest positive deviance and this is therefore selected as the classical representation upon which the representation specification is to be based.

Custom, variant and hybrid representations.

The selection guidelines given above are strongly influenced by deviance in deciding
which representation to select. The emphasis is firstly placed upon negative deviance as this is harder to change than positive deviance. Positive deviance is not a problem unless the deviations are large in which case the representation may be difficult to make complete. Negative deviance on the other hand indicates that the standard representation language will need amending in some way in order for the primary representation to be adequately represented in that form.

The representation specification.

As stated earlier, the representation specification is the syntax/semantics of the most suitable classical representation as selected from the profile analysis stage.

Two example representation specifications are given in Appendix B.1 and Appendix B.2

5.4. The concrete specification.

5.4.1 Aim, role and perspective.

The previous stage in the development methodology involved taking the primary representation, analysing it in order to create two specifications, a domain specification and a primary representation specification. The domain specification is a formalised representation of the elicited knowledge contained within the primary representation whilst the representation specification is a formalised description of a suitable classical form with which to represent that elicited knowledge. The next step therefore is to combine these two specifications to produce the secondary representation in which the domain knowledge is represented in the form advocated by the representation specification.

Once the secondary representation has been created the next step is to select a suitable control architecture to place on top of it. This specifies the inference mechanism to be used in conjunction with the secondary representation.

The concrete specification is therefore a composite object composed of a secondary representation and a control architecture. The aim of the concrete specification is to provide a specification from which the system can be implemented.
The following diagram shows the inter-relationships that together compose the concrete specification.

![Diagram]

5.4.2 Creation of a secondary representation.

The creation of a secondary representation from the domain and representation specifications should not cause any significant problems. This is due to the development process used to reach this point, for the aim at this stage is to have a good match between the knowledge in the domain specification and the representation specification selected to map that knowledge.

The formal descriptions of both specifications enable their "type" information to be used to develop the secondary representation.

An example, taken from the motor car case study given in chapter six, is now given:

Domain specification: the domain specification in "Z" is of the form:

```
car
| switch-wiring, relay-wiring, wires, starter : {ok,not ok}
| lights : {high, dim}
| battery : {ok, bad}
| replace-switch-wiring, replace-relay-wiring, replace-battery
| replace-teeth, replace-pinion : {yes, no}
| wiring : {good, bad}
| noises : {loud}
| solenoid : {shorted, ok}
| crank : {normal, bad}
| teeth : {broken, ok}
```
car-starts

| car |
| starter-turns |
| lights-high |
| battery-ok |
| wires-ok |

wiring-1

| car |
| |
| not switch-wiring replace-switch-wiring = yes |

car-fails-to-start-1

| car |
| |
| car-fail-to-start replace-battery = yes |

Representation specification: The representation specification is in the form of a frame based representation:

<section-name>----------------------

<table>
<thead>
<tr>
<th>&lt;desc-spec&gt;</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>&lt;section-body&gt;</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>&lt;para-type&gt;</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>&lt;para-body&gt;</th>
</tr>
</thead>
</table>

<section-spec>::= <section-name> <desc-spec> <section-body>

<section-name>::= <simple-string>

<section-body>::= <pred> ref <section-spec> | <pred> <text-gen>

-124-
Secondary representation: The domain specification and representation specification together produce the following secondary representation:

RULE: Rule one IF switch-wiring = not ok THEN replace-switch-wiring = yes

RULE: Rule two IF relay-wiring = not ok THEN replace-relay-wiring = yes

RULE: Rule three IF starter-turns = not ok AND
\quad \text{lights} = \text{dim} \quad \text{AND}
\quad \text{battery} = \text{bad} \quad \text{THEN replace battery} = \text{yes}

RULE: Rule four IF starter-turns = not ok AND
\quad \text{lights} = \text{dim} \quad \text{AND}
\quad \text{battery} = \text{ok} \quad \text{AND}
\quad \text{wiring} = \text{bad} \quad \text{THEN replace-wiring} = \text{yes}

It should be noted that this example comprises only part of the full secondary representation.

5.4.3 Control architecture.

Introduction.

As stated earlier the production of a secondary specification brings together the formal domain specification and the representation specification giving a representation that has the potential for information to be deduced and retrieved from it. In order to make inferences possible from the representation, a control architecture has to be applied to it. For example, a set of rules can not in themselves produce inferences, but when a backward chaining control architecture is place over them an inference mechanism becomes possible.

The process of selecting an appropriate control architecture is very poorly documented in the literature, most of which takes the stance that the developer has decided upon a representation and also "knows" that say, a breadth first search is the most appropriate strategy to use for that representation.

In this section we will examine what control architectures are available and discuss how a suitable one can be selected.
Rule based systems architectures.

One of the most extensively used secondary representations is that based on the rule formalism, which can be transformed into a 'production system' [Nilson. 1982] [Post. 1943] when a control architecture is place on top of it.

The process of controlling a production system can be considered as equivalent to controlling search in a graph, a node being the current state of the context list and the fan out from the node being the rules available to fire from that state.

The control architecture which examines the search space of the graph for a solution can be said to have two aspects:

1. The direction of search.
2. Selection of an applicable search procedure.

where the direction of search could be:

1.1. Forward.
1.2. Backwards.
1.3. Bi-directional.

and the search procedure could be one of:

2.1. Breadth first search.
2.2. Depth first search.
2.3. Heuristic.
2.4. Generate and test.
2.5. Hill climbing.
2.6. Production reduction.
2.7. Constraint satisfaction.

Each of these is extensively documented in the literature [Rich. 1983] [Nilson. 1982].

Thus it is the aim of the control architecture stage to determine which of these techniques would be applicable to the given secondary representation in order to achieve a maxi-
nal inference capacity. Theoretically in order to do this the topology of the search space must be examined and analysed. However, as Rich states [Rich. 1983] this is not always possible:

"The complex structure of the knowledge used in most a.i programs makes mathematical analysis of the corresponding programs very hard. But there are a few interesting results in this area. And it is important to keep performance questions in mind as we design programs even if we can not answer them exactly".

The results mentioned by Rich are those of researchers such as [Lewis. 1978], [Reingold. 1977], [Martelli. 1977] and [Gashnigg. 1979], who have examined some well defined cases such as 'the travelling salesman' and other NP complete problems [Lewis. 1978] along with algorithms such as A* [Nilson. 1982], [Gashnigg. 1979], [Aho. 1983].

The knowledge engineer should consider this work in the selection of a control architecture for a problem. However, the optimal solution to any given problem may be very difficult to achieve for a complex domain. It may be necessary for the knowledge engineer to achieve a working solution and then attempt to optimise it for worst, average, best or other case of solution.

Control architectures for semantic networks.

A secondary representation that is based upon a semantic network structure can be used to provide inferences when a control architecture such as 'spreading activation' [Quillian. 1976] is used.

Spreading activation is the underlying strategy for searching many forms of semantic networks and has two sub-categories: uni-directional and intersection search. In these searches:

"A network fragment is constructed, representing a sought-for object or a query, and then matched against the network database to see if such an object exists" [Barr. 1981]

A uni-directional search is where the search starts from a single point and spreads out from that point, while an intersection search is where the search is initiated from more than one point and a multi directional search occurs. This is sometimes called 'island driving' [Paxton. 1976].

The search strategies mentioned are well documented in the literature.
Control architectures for frames.

The control architectures for frame based representations vary with the specific kinds of knowledge the frames will contain and with the sort of reasoning that the system will be called upon to perform.

The control architecture of a frame system has as its first task to identify a frame that can act as its initial state. This is a frame applicable to the current knowledge. Once a frame has been ‘instantiated’ the next task is to fill any empty slots. This results in one of two states: either all of the slots become filled or else they fail to be all instantiated. Both situations leave the control architecture with several options. If the frame has all its slots filled then the next frame has to be chosen, this may be done by developing a conflict resolution strategy or say through reference to a specific link between the frames. If the frame has failed to be instantiated then, when this is the first frame, the reason why the slots were not filled may indicate a better frame with which to start the procedure again. If the frame was not the first frame but one encountered during the problem solving process then the control architecture must try to re-select a more appropriate frame. This may be done for instance by backtracking up through the hierarchical structure and selecting another frame or by utilising the default reasoning capabilities of the frame if they are available.

Comments on control architectures.

The variety of forms that can be used as secondary representations necessitates that an associated variety of control architectures be available. These range from the basic ones, such as some of those given above, to the more complex, such as ‘blackboard architectures’ [Reddy, 1976]. The knowledge engineer will therefore make an initial selection and then refine and tailor that architecture to the given secondary representation.

5.5. Towards implementation of the concrete specification.

The processes involved in developing concrete specifications towards implementation have not been investigated in greater detail as we feel that this is beyond the scope of this work, however the case study given in chapter 6 does take the development process through to implementation.
The role of the concrete specification is to act as a specification that the knowledge engineer could give to an A.I programmer very much in the way that a systems analyst gives Jackson Diagrams [Jackson, 1975] and other forms of instructions to programmers to implement systems in the commercial data processing environment.

The a.i. programmer, as for his commercial counterpart may be allowed some freedom to select the language or environment used to implement the system. For example if a concrete specification based on a production system representation with say a backtracking control architecture were given to programmer, then with a free choice of programming language any of the following might be selected: OPS5 [Forgy, 1981], Lisp [Winston, 1981] [Hughes, 1986], prolog [Clocksin, 1981].

5.6 Comments and conclusions.

In this chapter we have examined and discussed the processes and operations that occur in order to change the primary representation into a concrete specification.

The stages involved in this transformation have highlighted several points that are fundamental to the development of any knowledge based system. The first is the need to separate the specification of the domain knowledge from the selection and specification of the (classical) representation that will ultimately be used in the implementation process. The specification of the knowledge should be in a language that is designed for the purpose, as it is only in a suitable form that the consistency, correctness, etc., of the knowledge can accurately be judged. In our design methodology we chose the "Z" specification language to fill this role, but our philosophy is that the knowledge engineer can use any 'suitable' specification language. A comparative discussion of specification languages was given, as the literature is weak on the use of specification techniques within the expert systems field.

The selection of a suitable representational form is a topic about which workers in the KBS community often have strongly focused opinions. In this chapter and thesis we have adopted the philosophy that 'the representation must be made to fit the knowledge not the knowledge to fit the representation'. We proposed that by considering the knowledge types within the elicited knowledge, a representation that caters for them can be produced, making
the representation and manipulation of that knowledge easier and more natural. We constructed a theory of knowledge types and discussed their inter-relationships and dependencies. The work indicated that there are five knowledge types which constitute a large percentage of the knowledge elicited from an expert source. However, these knowledge types are not discrete, with the fundamental knowledge acting as building blocks for compound and more complex types.

The second point that needs to be considered is that once the individual knowledge and representation specifications have been reconstituted to form a secondary representation then a suitable control architecture needs to be selected. The current literature on the selection process is weak, however we indicate the options available and give some guidelines towards determining a suitable architecture.

Having created a secondary representation and selected an architecture they can then both be given to a programmer to be implemented in the form of a concrete specification. This is the basis of our third point, that the developer of a system should be confident that the system specification accurately reflects the system he intended.
A DETAILED CASE STUDY IN EXPERT SYSTEM DEVELOPMENT.

6.1 Introduction.

In Chapter 2 we discussed the existing approaches to developing expert systems, identifying the weaknesses and deficiencies within these approaches. We then described in detail (chapters three, four and five) the theoretical basis for a development methodology that attempts to overcome the problems associated with the earlier methods. Our methodology was based upon three principles: Firstly, that every step in the development, from initial specification to concrete specification should be capable of justification. Secondly, that this path has a point at which the domain is represented in an implementation independent form - the domain specification. Thirdly, that a suitable representation is chosen for the domain and that this choice is capable of justification - this is the representation specification.

Having described the theory the next stage is to show how it works in practice. In this chapter we will develop a case study showing how our development methodology can be applied to a motor-car domain to produce a diagnostic system. The processes involved at each stage will be examined, highlighting any practical problems that may arise and how an attempt to solve them might be made.

If the reader should need a recap of the whole development methodology they should refer to chapter 3.1.

6.2 Initial specification.

For the purposes of this case study we could take the initial specification to be the following:

"The goal of the project is to prepare a concrete specification that can be used for the implementation of a knowledge based system for the diagnosis of possible faults on a non starting motor car. The system is aimed towards a user who is naive about problems of this type."

However in a more realistic system we would expect a larger more detailed specification.
6.3 The knowledge elicitation phase.

6.3.1 Introduction.

The first major stage to be encountered is that of knowledge elicitation, in which knowledge is extracted from the domain expert. The emphasis at this stage is placed on ensuring the full extent of the knowledge is obtained, with little or no analysis taking place.

6.3.2 Choice of elicitation technique.

In Chapter 3 we discussed some of the knowledge elicitation techniques available to the knowledge engineer. These were then considered with regard to (use in) our methodology and the following were thought applicable:

1. Interview
   - structured
   - unstructured
   - focused

2. Reporting
   - on-line
   - off-line
   - hybrid

3. Literary sources
   - books, papers and journals

4. Formalised techniques
   - repertory grid
   - inference structure

For the purposes of this case study we are taking the knowledge source to be (of the third type), a literary source.

The literary work we used as the basis of our elicitation phase is a section from "Automobile engines - questions and answers" by Sully and Unstead [Sully, 1983]
6.4 The elicited representation.

The result of the knowledge elicitation phase in this case is the following text:

Engine cranks very slowly or not at all - turn on the headlights; if the lights are dim, most likely the battery or connecting wires are at fault. Check the battery using procedures described in chapter 7. Check the wiring for breaks, shorts and dirty contacts. If the battery and connecting wires check good, turn the headlights on and try to crank the engine. If the lights dim drastically the starter is probably shorted to ground. Remove the starter and test it using the procedures described in chapter 7. If the lights remain bright or dim slightly when trying to crank the engine, the trouble may be in the starter, solenoid or wiring. To isolate the trouble short the two large solenoid terminals together (not to ground); if the starter cranks normally, check the solenoid wiring up to the ignition switch and seat belt interlock relay(1976). If the starter still fails to crank properly, remove the starter and test it.

Starter turns but does not engage the engine - this trouble is usually a defective pinion or solenoid shifting fork. It may also be that the teeth on the pinion flywheel ring gear or both are worn down too far to engage properly. Starter engages but will not disengage when ignition switch is released - This trouble is usually caused by a sticking solenoid, but occasionally the pinion can jam on the flywheel. With manual transmissions the pinion can be temporarily freed by rocking the car in high gear. Naturally this is not possible in automatics; the starter must be removed. Loud grinding noises when starter runs - this trouble may mean the teeth grinding on the pinion and/or flywheel are not meshing properly or it may mean the over running clutch is broken. In the first case remove the starter and examine the gear teeth. In the latter case, remove the starter and replace the pinion drive assembly.

6.5 The knowledge acquisition phase.

The next stage in the methodology is that of knowledge acquisition, this being the process of analysing the knowledge extracted during the knowledge elicitation phase.

The aim of this phase is to produce a representation (the primary representation) of the elicited knowledge that is rigorous enough to allow several demanding analyses to take place upon it, one of these ultimately producing a formal specification of the domain and another acting as the basis for the selection of a "classical" representation.

The reason why the elicited representation is not used directly as the basis of these analyses, together with a detailed description of the processes used to develop a primary representation, are given in chapter 4.

The following diagram can be used to help remind us of the stages involved:
These stages can be summarised as:

1. The knowledge elicitation phase produces the elicited representation.
2. The elicited representation is then examined for the transformational characteristics and a trace of these characteristics is produced. This process is known as primary analysis.
3. The primary trace is then compared to the bandwidth diagrams of the primary representation. This is known as trace matching.
4. The trace matching procedure will suggest the most suitable primary representation to use.
5. The elicited representation is transformed into the primary representation formalism.

6.6 The primary representation phase. (For the starter motor domain).

In order to produce a primary representation the five steps given above will now be followed.

Step 1. (The knowledge elicitation phase produces the elicited representation)

This has already been performed and is given above.

Step 2. (The elicited representation is examined for transformational
characteristics and a trace of these characteristics is
produced - this is the primary representation phase.)

Noise: The elicited representation is virtually noise free. This is due to the knowledge elicitation technique used. Thus we will assign the value minimal.

Modularity: The modularity is high. This can be justified by the way the text is partitioned into self-contained units. It is primarily divided into two sections: an area dealing with electrical problems and another dealing with mechanical problems. Within these two major sections are other smaller subtopics such as lighting problems, wiring problems and battery problems, each of which are short and concise making the value high rather than significant.

Linkage: The linkage is significant. This is a resultant of interaction between the highly modular representation in which links are created both internally between data items and externally between procedures.

Operational types: The elicited representation is composed mainly of test, diagnosis and actions. Thus the level of operational types is low.

Sequencing: The sequencing of the knowledge within the elicited representation is significant. The modularity of the knowledge being high along with significant linkage makes for significant sequencing. The sequencing is not high because it is felt that there may be repetition, areas of incompleteness and undefined paths from conditionals.

Justification: The level of justification is low. This is because both the explicit and implicit justification of action is low in the primary representation.

Explanation: The level of explanation is significant as the text contains high amounts of diagnostic advice this being both explicit within the representation and implicit through the form used as the primary representation.

Having examined the elicited representation for these characteristics it is now possible to draw the primary trace:
Steps 3 & 4. (The primary trace is compared to the bandwidth diagrams of the primary representations in order to decide which is the most suitable - this is known as trace matching)

The bandwidth diagram for the flow diagram is as follows:

The bandwidth diagram for the contour diagram is:

The bandwidth diagram for decision tables is as follows:
The reasons for the shape of these bandwidths were given in chapter 4.1.

Step 4. (The trace matching procedure will suggest the most suitable primary representation to use.)

It is clear from the trace and bandwidth diagrams that the flow diagram is the most suitable representation with which to re-represent the elicited representation, the bandwidth of the flow diagram being a better fit than that of the contour diagram or the decision table. The area of the contour diagrams bandwidth is larger and therefore covers more cases and is consequently more general.

Step 5. (The elicited representation is transformed into the primary representation formalism.)

The flow diagram will be created from the elicited representation by the following guidelines:

1. Take a small segment of text and break this down into the components of the flow diagrams - actions, decisions, processes etc.
2. Construct a separate flow diagram for each of the textual points.
3. Unify the separate flow diagrams into one large diagram.
4. Examine the structure for repetitions and redundancies.
5. Examine the structure for deficiencies i.e., undefined paths from decision boxes, incomplete actions upon diagnosis of problems.
6. Re check initial representations to improve the deficiencies.

6.7 The primary representation phase.

The flow diagram of the elicited knowledge:

See figure 8 on page 58.

We can show that the flow diagram is syntactically correct by re-representing it as a pseudo code which adheres to the definitive syntax of our flow diagram. The B.N.F of the language is given in appendix B.3 and the pseudo code is given in appendix B.4

6.8 Using the primary representation.

The creation of the primary representation is a significant point in the development process. It is the first point at which a position of adequacy has been reached, even if the level of adequacy is quite restrictive. (see chapter 4.3 for more details on adequacy). To reach a higher level of adequacy it is necessary to develop from the primary representation a more rigorous representation. This is one of the aims behind the production of a "domain specification", a formal specification of the elicited domain knowledge.

The adequacy of the primary representation also enables it to act as the basis of analysis techniques which indicate the most suitable classical representation in which to carry forward the design of the system.

6.9 Domain specifications.

The primary representation provides us with a more rigorous form than the elicited representation with which to reason about the domain. However, this representation has several drawbacks. The primary representation itself is still far too ambiguous and may contain inconsistencies and incompleteness that can not be spotted due to the structures used. It is the aim of the domain specification to help reduce these problems.

6.10 A domain specification (For the starter motor domain).

The creation of the "Z" specification follows an analysis of the primary representation. The approach taken is to examine each of the operational types in the flow diagram taking
each item in turn and using that as the basis for the predicate part of the "Z" box. This predic­
cate then has suitable data types created for it, if non have already been created. The specifi­
tion is then built up trying to ensure consistency through cross checking the "Z" statements and
enforcing strict, consistent data typing.

The following is a section of the primary representation specified in "Z":

```
car ________________________________________________________________
  | switch-wiring, relay-wiring, wires, starter : {ok,not ok}  
  | lights : {high,dim}                                    
  | battery : {ok,bad}                                    
  | replace-switch-wiring, replace-relay-wiring, replace-battery  
  | replace-teeth, replace-pinion : {yes,no}                  
  | wiring : {good,bad}                                     
  | noises : {loud}                                        
  | solenoid : {shorted,ok}                                 
  | crank : {normal,bad}                                    
  | teeth : {broken,ok}                                     
```

```
car-starts ________
  | car
  | ________________________
  | starter-turns
  | lights-high
  | battery-ok
  | wires-ok
  | ________________________

  starter-turns ____________
  | car
  | ______
  | ______
  | starter = ok
  | ________________________

  lights-high ____________
  | car
  | ______
  | ______
  | lights = high
  | ________________________
```
lights-dim

<table>
<thead>
<tr>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
| lights = dim

---------------------

loud-noises

<table>
<thead>
<tr>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
| noises = loud

---------------------

battery-ok

<table>
<thead>
<tr>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
| battery = ok

---------------------

wires-ok

<table>
<thead>
<tr>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
| wires = ok

---------------------

wiring-1

<table>
<thead>
<tr>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
| not switch-wiring replace-switch-wiring = yes

---------------------

wiring-2

<table>
<thead>
<tr>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
| not relay-wiring replace-relay-wiring = yes

---------------------

switch-wiring

<table>
<thead>
<tr>
<th>car</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
| switch-wire = ok

---------------------
relay

| car
| relay-wire = ok

| car-fails-to-start ___
| car
| not starter-turns
| lights-dim
| battery = bad

| car-fails-to-start-1
| car
| car-fail-to start replace-battery = yes

| car-fails-to-start-2
| car
| car-fails-to-start
| battery = ok
| wiring = bad

| car-fails-to-start-part2
| car
| car-fail-to-start-2 replace-wiring = yes

| car-fails-to-start-3
| car
| car-fails-to-start-2
| battery = ok
| crank = ok
| solenoid = ok
| not switch-wiring
crank-ok

| car
| crank = normal

solenoid-ok

| car
| solenoid = normal

switch-wire-ok

| car
| switch-wiring = ok

---

car-fails-to-start-part3

| car
| car-fails-to-start-3 replace-switch-wiring = yes

---

car-fails-to-start-4

| car
| car-fails-to-start-2
| battery = ok
| crank = ok
| solenoid = ok
| not relay-switch

---

car-fails-to-start-part4

| car
| car-fails-to-start-4 replace-relay-wiring = yes

---

142
car-problem

1

car

1

1 starter-turns

1 loud-noises

---
car-problem-2

1

car

1

1 car

1 car-problem \Rightarrow starter \neq ok

---
starter-problem

1

car

1

1 starter \neq not ok

---
teeth-test

1

car

1

1 starter-problem

1 teeth-broken

---
teeth-test-two

1

car

1

1 teeth-test \Rightarrow replace-teeth = yes

---
teeth-broken

1

car

1

1 teeth = broken

---
6.11 Towards the representation specification.

The next step in the development methodology is to identify which (if any) classical or hybrid representation is the most suitable form around which to base the representation specification. The approach is described in detail in chapter 5. However, we can summarise it through use of the following diagram and points:
1. Creation of a knowledge profile for the elicited knowledge contained within the primary representation. This is known as profile analysis.

2. Comparison of the domains knowledge profile (created in stage one) with profiles for the classical representations. This is the profile matching process.

3. Selection of the most suitable form around which to base the representation specification.

These will now be followed in an attempt to select the most suitable representation.

6.11.1 Profile analysis (Stage 1).

The knowledge profile for the motor car domain.

By considering the knowledge definitions and guidelines given in chapter five, for constructing a profile from a primary representation which is in the form of a flow diagram, the following knowledge profile was constructed for the motor car domain:
Brief resumes of how these levels were derived will now be given.

**Factual.**

The elicited knowledge contained within the primary representation includes a high amount of factual knowledge. This is because the facts and data-items that compose the conditions are each separately defined in the primary representation.

**Control.**

The primary representation contains an intermediate amount of control knowledge. We can state this as the conditionals are total argumental in nature and are found in proportions that are balanced and consistent with the other operation types.

**Heuristic.**

There is very little heuristic knowledge, due to the precise relationships expressed in the primary representation.

**Procedural.**

The flow diagram contains an intermediate level of procedural knowledge. The procedures, characterised as sequences of conditionals with associated actions and responses concerned with an individual topic, can be found in proportions equal to those paths through the diagram that are non procedural.
Conceptual.

The amount of conceptual knowledge associated with the primary representation is minimal as the domain is reasoned about at only one level, without giving any capacity for abstraction.

6.11.2 Profile matching (Stage 2).

Having drawn the domain profile the process of selecting a suitable classical representation can be considered.

The selection process is based upon the following guidelines:

1. Overlay the representation profile and domain profiles.
2. Count the divergences both positive and negative and place them in a table.
3. Summate all the positive divergences and give them a non numeric value.
4. Summate all the negative divergences and give them a non numeric value.
5. Select the representation with the lowest number of negative divergences.
6. If two or more representations have the same number of divergences then examine the representations removing the one with the largest individual negative divergencies repeat this process until only one representation is left or two or more have equal negative divergencies. In this case repeat the process using positive divergences.
7. If two or more representations are equal in all aspects then if after reconsideration of their profiles they are still equal then examine whether a change from the standard representation would aid the matching process.
8. If the divergences are great then investigation of new or hybrid representations may be appropriate.

Step 1.

Comparison one.

Frame representation profile and domain profile.
Comparison two.

Rules profile and domain profile.

Comparison three.

Semantic network profile and domain profile.

Steps 2, 3 & 4.

2. Count the divergencies and place them in a table
3. Summate all the positive divergencies and give them a non-numeric value
4. Summate all the negative divergencies and give them a
Step 5. Select the representation with the lowest number of negative divergencies

This reduces the representations to select from frames and semantic networks.

Step 6. Examine the negative divergencies in the matches

They are both total matches.

Step 7. Examine the positive divergencies

Even though they both have high values of positive divergence, frames have a lower deviation over all knowledge types, whilst also having two matches against the semantic networks one.

We therefore select frames as the representation to use in the development of this domain.

6.12 The representation specification.

The language used as the representation specification can be found in Appendix B.1

6.13 The concrete specification.

6.13.1 The secondary representation.

We will now bring together the formal domain specification and the frame
based form of the representation which has been selected as the representation specification to create a secondary representation from which the domain can be implemented.

```
starter
|
|
|
motor car starter troubleshooting
|
|
|
| (problem)? ref starter turns
| (not problem)? end
|
```

```
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
|
starter-problems

desc: this section deals with starter problems

@test-1? ref elec-problem
@non test-1)? ref starter-motor-problems

Elec-problems

desc: This section deals with electrical problems

@test-2? text1
@not test-2)? text2
@test-3)? text3
@not test-3)? text4

@test-2 [fact]

desc: we need to test the battery
expl: this tries to isolate the
problem to the battery area
query: is the battery ok

@test-3 [fact]

desc: we need to test the wiring
expl: this tries to isolate the
problem to the wiring
query: is the wiring ok

text 1

The battery is ok
replace the battery

---

wiring is ok

---

replace the wiring

---

**starter-motor-problems**

desc: This section tests the area of starter motor problems

(solenoid)? V (crank)? ref starter-diag

**starter-diag**

desc: may be the starter motor

(starter-motor)?

---

solenoid [fact]

desc: we need to test the wiring
expl: This isolates the problem to a wiring circuit
query: are the solenoid terminals shorted together
6.13.2 The control architecture.

In chapter five we examined the various control architectures available for a frame based secondary representation. The standard method of control in diagnostic systems such as MYCIN [Shortliffe, 1976] is that of a backward chaining strategy which employs backtracking. This is also the mechanism that we shall use on our secondary representation. It is advantageous to use this approach with our frame based representation as it utilises fully the predicate features of the frames that are the basis of the inheritance mechanism. It is important therefore that the predicates are manipulated in a sufficiently effective yet efficient way. The efficiency
of this approach is one reason why this control architecture is often used by diagnostic shells such as AGE [Nii, 1979] and EMYCIN [VanMelle, 1979].

6.14 The implementation stage.


In our description of the development methodology in the previous three chapters we have not discussed how the concrete specification should be implemented. The thesis has emphasised the knowledge acquisition methodology rather than implementation as being the more critical and difficult task. However to demonstrate the value of the method we show in the following section that it is possible to implement a system from the concrete specification.

6.14.2 Concrete specification.

The secondary representation was the result of combining the formal domain specification with a frame based language chosen as our representation specification. This frame language was the 'Conceptual language' described and developed in [Plant, 1985] and was designed to assist the knowledge engineer to implement his system in the shell language KRL upon which ESP advisor runs. The shell incorporates a backward chaining architecture but does not facilitate backtracking.


The constructs used in the concrete specification language and those used in KRL are very similar, the difference being that those used in the concrete specification are intended to allow the knowledge engineer to develop and design the knowledge base whilst those of KRL are designed to allow the knowledge base to be translated into Prolog by its compiler.

To show how different types of frame are mapped to KRL we will examine an example of each type. The domain is that of a help system for UNIX.

If the top level section "title" is considered, this is represented by the frame:
This is equivalent to the KRL code:

```kern
title "unix help facility"
{beginner} reference getting-started
{not beginner} reference system-commands
```

These entities have a close correspondence as does the predicate "beginner" in its frame representation:

```kern
beginner [fact]

desc: you are new to this operating system

expl: this is to determine the level of help necessary

query: are you a beginner with UNIX
```

and its KRL code:

```kern
beginner: " you are new to this operating system"

desc: you are new to this operating system

expl: this is to determine the level of help necessary

query: are you a beginner with UNIX
```

The other parameter types also have a close mapping between their model representation and their coded form. For example the category parameter described by the frame:

```kern
option-16 [category]

desc: the calendar, time and date functions

expl: tests to see which function from the chronological ones is to be used.

options: cal, calendar, date, time, expanded format, none

query: please select the function you are interested in
```

(Which) represents the following code:
option-16: "The calendar, time and date functions"
category
explanation
"tests to see which function from the chronological ones" & "is to be used."
options
cal - "cal"
calendar - "calendar"
date - "date"
time - "time"
expanded-format - "expanded format"
none - "quit"
askable
"please select the function you are interested in".

Other function types such as numbers and phrases can be similarly mapped from the frame model into KRL.

Thus it can be seen that each parameter frame maps quite easily into KRL code. The question might be asked that if the mapping is so good, why bother with the frame language at all? The answer to this returns us to the fundamental argument of why it is necessary to have any form of design tools or methods. This topic is discussed in chapters 2 and 5.2. Briefly however, it can be seen that any attempt to design a knowledge base in the KRL language would be less than productive due to its heavy syntax and its tolerance of an unstructured programming style, whereas our development methodology allows us to abstract ourselves from the intricacies of the implementation and reason about each component that influences the system before bringing them together as a whole.

6.15 The implementation.

The guidelines for transforming the frame language into KRL led to the KRL program given in Appendix B.5

Utilising the implementation.

The implementation process was quickly performed (1 day) and is successful in that it allows the user to interrogate the knowledge base to derive a solution. A example dialogue produced was of the form:

(user responses are in uppercase)

> Is there a problem? YES
> Does the starter motor turn over? NO
> Do the headlights dim? YES
> Is the battery ok? WHY

I am asking because I wish to establish whether we need to
test the battery (s4_test2)
Which is a necessary pre-condition for the display of the following paragraph:
"is the battery ok"
> Is the battery ok? EXPLAIN

Further information

-------------------------
This tries to isolate the problem to the battery area.
> Is the battery ok? DIRECTORY

Directory of parameters.

-------------------------

problem: try to find out if this is a new problem (true) s1_turns: if the starter motor turns
then isolate the problem area (false) s2_full_listing: information on the problem area (?) s3_test1: we need to test whether the lights dim (true) s4_test2: we need to test the battery (?) s4_test3: we need to test the wiring (?) s5_solenoid: we need to test the wiring (?) s5_crank: we need to isolate the problem (?) s6_starter_motor: we need to isolate the
problem (?)
> Is the battery ok? YES
> Is the wiring ok? NO
>> replace the wiring <<

6.16 Comments on the development.

In this first part of chapter six we have shown that it is possible to apply the development methodology described in chapters 3, 4 & 5 to a domain and produce an implementation.

The size of the elicited representation used in this case study was kept small in order not to overwhelm the reader with masses of knowledge, information and text. A second reason for keeping the elicited representation size down was that it allows us to illustrate and discuss points that may lose their clarity were the text larger. For example: in selecting the representation specification we used three classical representations in our matching process (frames, rules and semantic networks) from which the guidelines suggested that the frame representations would be the most appropriate, semantic networks a close second with rules the most unsuitable, the frame based system being developed into an implementation. However, due to the 'manageable' size of text used in this study we decided to take our investigation of it one step further and examine what the consequences would be if we were to take a representation specification that was not the most suitable according to the profile matching process and try to
develop the a concrete specification from it. This part of the case study will now be described.

6.17 Further investigation into primary analysis.

6.17.1 Introduction.

In this section we would like to add further discussion to the process of selecting the representation specification. We will do this by taking the motor car domain and developing not the best profile match but one with a poor profile match. This will enable us to examine the validity of our selection technique.

6.17.2 Profile selection.

In our profile selection procedure the profile matches gave us the following results:

frame match:

| n.d. | 0 | 0 | 0 | 0 | 0 | 0 | match |
| p.d. | 0 | 2 | 4 | 0 | 1 | 7 | high |

rule match:

| n.d. | 4 | 0 | 0 | 1 | 0 | 5 | high |
| p.d. | 0 | 2 | 2 | 0 | 2 | 6 | high |

semantic network match:

| n.d. | 0 | 0 | 0 | 0 | 0 | 0 | match |
| p.d. | 0 | 1 | 2 | 1 | 4 | 8 | high |

From which we can see that both frames and semantic networks have similar matches and even though by following the selection guidelines, frames was chosen, the semantic network formalism had a strong claim to be selected. However, the rule based form was indicated by the profile match to be clearly the least suitable, having both high positive and negative divergencies. This poor match was made worse as there was no individual match amongst the separate knowledge types. To see the significance of this we will now take this rule formalism and examine what would be the consequences of developing a secondary representation from it.
6.17.3 The representation specification.

The representation specification is based on the production system model given by Barr and Feigenbaum [Barr. 1981].

6.17.4 The concrete specification.

The development of a rule based secondary representation.

We will proceed by examining each part of the domain specification and its suitability for representation in the rule form.

```
car | switch-wiring, relay-wiring, wires, starter : {ok, not ok}
    | lights : {high, dim}
    | battery : {ok, bad}
    | replace-switch-wiring, replace-relay-wiring, replace-battery
    | replace-teeth, replace-pinion : {yes, no}
    | wiring : {good, bad}
    | noises : {loud}
    | solenoid : {shorted, ok}
    | crank : {normal, bad}
    | teeth : {broken, ok}
```

In the frame system all of the elements are typed. This cannot be done in the standard rule based representation to the same extent.

```
car-starts
```

```
car
  +-----------------
  | car
  +-----------------
  | starter-turns
  | lights-high
  | battery-ok
  | wires-ok
```

This utilises four predicate elements, Boolean in type:

```
starter-turns
```

```
starter = ok
```
lights-high

| car
|    |
| lights = high

battery-ok

| car
|    |
| battery = ok

wires-ok

| car
|    |
| wires = ok

Which along with two other Boolean types:

lights-dim

| car
|    |
| lights = dim

loud-noises

| car
|    |
| noises = loud
can not be defined in our standard rule based representation.

The next schema:

wiring-1

| car
|    |
| not switch-wiring ⇒ replace-switch-wiring = yes

specifies a rule of the form:
RULE: Rule one IF switch-wiring = not ok THEN replace-switch-wiring = yes

Similarly the schema

```
wiring-2 _____________________________
|    |
| car |
|______|
| not relay-wiring ⇒ replace-relay-wiring = yes
```

gives:

RULE: Rule two IF relay-wiring = not ok THEN replace-relay-wiring = yes

We can not define the following facts:

```
switch-wiring __________
|    |
| car |
|______|
| switch-wire = ok
```

```
relay _________________
|    |
| car |
|______|
| relay-wire = ok
```

We now encounter our first complex conditional:

```
car-fails-to-start __
|    |
| car |
|______|
| not starter-turns
| lights-dim
| battery = bad
```

```
car-fails-to-start-1 _____________________________
|    |
| car |
|______|
| car-fail-to start ⇒ replace-battery = yes
```

Which produced our third rule:
RULE: Rule three IF starter-turns = not ok AND
lights = dim AND
battery = bad THEN replace battery = yes

We now encounter a second complex conditional:

```
car-fails-to-start-2
  | car
  | car-fails-to-start
  | battery = ok
  | wiring = bad
```

car-fails-to-start-part2

```
car-fails-to-start-2 => replace-wiring = yes
```

Which is similar to that of rule three:

RULE: Rule four IF starter-turns = not ok AND
lights = dim AND
battery = ok AND
wiring = bad THEN replace-wiring = yes

we can also define the following individual rules:

RULE: Rule five IF battery = bad THEN replace-battery = yes

RULE: Rule six IF wiring = bad THEN replace-wiring = yes

We now come to a further complex condition; defined by the following schema:

```
car-fails-to-start-3
  | car
  | car-fails-to-start-2
  | battery = ok
  | crank = ok
  | solenoid = ok
  | not switch-wiring
```
crank-ok
|  
|  
|  car
|  
|  
|  crank = normal
|  

solenoid-ok
|  
|  
|  car
|  
|  
|  solenoid = normal
|  

switch-wire-ok
|  
|  
|  car
|  
|  
|  switch-wiring = ok
|  

car-fails-to-start-part3
|  
|  
|  car
|  
|  
|  car-fails-to-start-3 \Rightarrow replace-switch-wiring = yes
|  

Which gives the following rule:

RULE: Rule seven IF

- starter-turns = not ok AND
- lights = dim AND
- battery = ok AND
- crank = normal AND
- solenoid = shorted AND
- switch wiring = not ok

THEN replace-switch-wiring = yes

Similarly for car-fails-to-start-4:

car-fails-to-start-4
|  
|  
|  car
|  
|  
|  car-fails-to-start-2
|  
|  battery = ok
|  
|  crank = ok
|  
|  solenoid = ok
|  
|  not relay-switch
|  

-163-
RULE: Rule eight IF starter-turns = not ok AND lights = dim AND battery = ok AND crank = normal AND solenoid = shorted AND relay wiring = not ok THEN replace-relay-wiring = yes

We can obtain two further rules from these texts:

RULE: Rule nine IF switch wiring = not ok THEN replace-switch-wiring = yes

RULE: Rule ten IF relay wiring = not ok THEN replace-relay-wiring = yes

The schema:

car-problem __________________________________________
| car
| __________
| __________
| starter-turns
| loud-noises

_____________________________________________________

car-problem-2 _________________________________________
| car
| __________
| __________
| car-problem starter = not ok

giving:

RULE: Rule eleven IF starter = engages AND noise = loud THEN starter = not ok

Again the starter-problem fact can not be independently given:
We then have two more complex definitions:

```
tooth-test
  car
  
  starter-problem
  teeth-broken

  tooth-test-two
  car
  
  tooth-test replace-teeth = yes

  tooth-broken
  car
  
  teeth = broken

  and

  tooth-test-three
  car
  
  tooth-test
  pinion-break

  pinion-break
  car
  
  pinion = broken
```
Which can be written as:

RULE: Rule twelve IF starter = not ok AND teeth = broken THEN replace-teeth = yes

and

RULE: Rule thirteen IF starter = not ok AND teeth = not broken AND pinion = broken THEN replace-pinion = yes

Also

can become:

RULE: Rule fourteen IF pinion = broken THEN replace-pinion = yes

We can now draw all our rules together to form the secondary representation:

RULE: Rule one IF switch-wiring = not ok THEN replace-switch-wiring = yes

RULE: Rule two IF relay-wiring = not ok THEN replace-relay-wiring = yes

RULE: Rule three IF starter-turns = not ok AND lights = dim AND battery = bad THEN replace battery = yes

RULE: Rule four IF starter-turns = not ok AND lights = dim AND battery = ok AND wiring = bad THEN replace-wiring = yes
RULE: Rule five IF battery = bad THEN replace-battery = yes

RULE: Rule six IF wiring = bad THEN replace-wiring = yes

RULE: Rule seven IF starter-turns = not ok AND lights = dim AND battery = ok AND crank = normal AND solenoid = shorted AND switch wiring = not ok THEN replace-switch-wiring = yes

RULE: Rule eight IF starter-turns = not ok AND lights = dim AND battery = ok AND crank = normal AND solenoid = shorted AND relay wiring = not ok THEN replace-relay-wiring = yes

RULE: Rule nine IF switch wiring = not ok THEN replace-switch-wiring = yes

RULE: Rule ten IF relay wiring = not ok THEN replace-relay-wiring = yes

RULE: Rule eleven IF starter = engages AND noise = loud THEN starter = not ok

RULE: Rule twelve IF starter = not ok AND teeth = broken THEN replace-teeth = yes

RULE: Rule thirteen IF starter = not ok AND teeth = not broken AND pinion = broken THEN replace-pinion = yes

RULE: Rule fourteen IF pinion = broken THEN replace-pinion = yes

6.17.5 Analysis of a rule based representation.

Secondary representation.

The initial development has given us a secondary representation that consists of fourteen rules. We will now try to further evaluate the significance of these findings and relate the merits of the rule based representation to that of 'the original' - frame based - secondary representation.
The rule based representation that has been produced can be used as the basis of a secondary representation. However, several questions that need to be examined are: 'How adequate is this rule based representation'; 'How does the frame formalism differ from the rule based representation, in its applicability to the domain' and 'is the frame representation still the most applicable'.

In order to answer the first of these questions, we will start by examining how well the rules represent each of the five principal knowledge types.

Factual: In the representation of factual knowledge the rules were equal to the profile prediction; in that the representation of facts was minimal. This is due to the definition of our representation. The standard we used does not have in its syntactic definition the ability to represent facts.

Control: The amount of total argumental knowledge within the rule based secondary representation measures up to the domain profile, producing fourteen rules in all. These rules are often long, complex, multi conditional structures, which implies a degree of specialisation over them.

Heuristic: There are no heuristics in the secondary representation which reflects the minimal value given to it in the knowledge profile.

Procedural: The ability to proceduralise rule based systems is limited. Techniques such as partitioning could be applied to divide the electrical problems from the mechanical problems. However, this does not tackle the problem of proceduralising complex statement groupings. Thus, the profiles reflect this, showing that there is an amount of negative deviance between the amount of procedural knowledge needed and the amount that the rules can represent.

Conceptual: The level of conceptual knowledge within the domain is minimal and is reflected in all the forms that represent the domain.

Despite this apparent success of the rule based form, we can see that our decision to use the frame based form was justified as the rule based representation is less well defined and consequently holds the knowledge and information in a weaker format. It is possible however
to change and adapt the standard representations and produce 'variant' representations better suited to representing a problem domain than the original standard one. We will now suggest some way that this could be done to the rule representation. The most outstanding feature of the rule based representation was that of weak knowledge typing. This could be immediately strengthened by allowing Boolean predicates (facts) to be defined, and would remove a significant negative divergence from the profile match. An additional re-representation stage could also be employed that would refine the representation by breaking down complex rules in such a way as to produce a larger number of rules that would cover a larger number of situations.

Control architecture.

Having decided upon a rule based secondary representation the next step is to determine which control architecture would be suitable. Several aspects have to be taken into consideration: firstly, there is a need for a conflict resolution strategy and secondly a suitable search mechanism needs to be found. The conflict resolution strategies available are well documented in the literature [Barr, 1981] [Rich, 1983] as are the search mechanisms (see section 5.4.3). The standard approach used in commercial systems that are diagnostic in nature is to use a backward chaining architecture with backtracking and a conflict resolution strategy that fires the most specific rule, that is the one with the most detailed condition part that matches the context list.

6.17.6 Analysis of the investigation.

Having developed both a frame based and a rule based concrete specification we can now discuss our findings.

The frame based representation has been shown to be a highly suitable medium through which to represent the domain. It has two highly desirable features: one being that it has strong typing and the other that it has a powerful inference mechanism. The strong typing allows us to define our knowledge whilst the frames' inheritance properties allows the knowledge to be proceduralised and allows the knowledge engineer to use different levels of abstraction where necessary.
In the standard rule based system many of the powerful features that make frames desirable were missing, thus reflecting the profile drawn for it. The representation did however, have the potential for being a suitable secondary representation around which to build a concrete specification. In order to achieve this, a 'variant' rule based representation would need to be developed which incorporated features lacking in the standard one. The most important of these features is the incorporation of a stronger typing facility. This could be achieved initially through adding the ability of the representation to represent factual knowledge and Boolean types.

The question (asked earlier) as to "which form is the most suitable to act as the basis of the concrete specification" will now be examined: In our original profile matching process, it was indicated that the frame representation would be a better basis for the concrete specification than the standard rule representation, and this is vindicated in our analysis. We now have to examine whether the ‘variant’ or extended rule representation that included definitions of factual knowledge would be more suitable than frames. If we were to draw a profile for our variant representation then it would be of the form:

![Diagram](fig. 43)

Which gives us a match of:

<table>
<thead>
<tr>
<th>n.d.</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1 = minimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>p.d.</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>6 = high</td>
</tr>
</tbody>
</table>

This is a far better overall match than that of the standard rule based representation and shows there to be an individual match for the factual knowledge type.

The profile match, when used as a basis for representation selection shows that it has
a lower positive deviance than frames but still has a negative deviation on procedural knowledge and it is this that allows frames to remain as the more suitable representation. In order to overcome the problems associated with a lack of representational procedurality, the rule based representation would have to develop a specialised control architecture.

A further consideration that may produce a representation equal to or better than a frame representation is a 'hybrid' representation, combining features of both frames and rules. A representation such as this could if desired be customised to meet the representational requirements of this individual domain, or be made generic enough for a wider applicability of usage. The investigation of this and other possible representations would be an interesting line of research to pursue, however this is beyond the scope of this thesis.

Comment on the investigation.

We have shown in this section that the original choice of representation specification was the correct one for the profile matches that had taken place. The study also revealed that it was possible to develop a variant representation that could be utilised in place of the frames if necessary and that had comparative representational characteristics. Thus, from this investigation we can say that with a larger range of representational profiles to match against, a profile match conflict resolution strategy may be needed. In such a case it may be that the control architectures will play a large role in the selection criteria. Since other things being equal, it may be desirable to use a representation that utilises a more flexible, less specialised control architecture than a more complex one.

6.18 COMMENTS AND CONCLUSIONS.

In this chapter we have tried to show how our development methodology can be used in practice, attempting to illustrate the theoretical aspects through the use of a small case study, a diagnostic system where the domain is that of starting a motor car.

Throughout the development it is possible to see that the three principals given at the beginning of this chapter were adhered to. Firstly, every step in the development was justified: the primary representation that was chosen is justified through the primary analysis while the choice of representation is justified through the profile matching process.
This latter justification is borne out by our investigation into the consequences of using an alternative representation from the best one with which to develop a set of specifications.

The study was also intended to show (if only on a small scale) how elicitation from a literary source can be used as part of a development. This is an important technique to consider as we feel that more extensive use of it will be made in the future, especially in industrial situations where the groundwork and less specialised knowledge required to build the initial layers of the knowledge base can be produced without the domain expert, only bringing him in at a latter stage - making for better resource utilisation.

The particular literary source used here was a technical manual for fault diagnosis in motor cars. It is important to select the best possible textual source for elicitation, in terms of information clarity and yield. The domain expert should be involved in this selection task and if it is known that this elicitation technique is to be used, the literature references can be included in the initial specification. However, sometimes there may be no choice of textual source and the primary analysis will have to take place over a less amenable elicited representation. Through developing other small systems from a diverse range of elicited representations, we found that these techniques for analysing and transforming text could be successfully applied to many types of text.

The case study has also shown some of the benefits that the utilisation of a domain specification gives, in that it assists the knowledge engineer to think about the domain in abstract terms away from the influences of representation methodologies and the intricacies of implementation languages. The domain specification also shows how maintenance, readability, completeness, consistency of the knowledge etc., are all heightened by its use.

The case study and the further analysis upon it showed that the selection of the representation specification is a very important part of the development process, as the result of selecting an unsuitable representation can be detrimental both in development time and capacity of the system to represent knowledge as effectively as possible.

The system ran and performed satisfactorily and as expected, however due to its limited size further comments on the run-time performance can not be given.
In conclusion we can say that the development process worked effectively on this small case study as it did for other examples which we have studied such as i) A verbal report produced by an expert hematologist [Fox, 1985] was taken and from this we developed a series of specifications; as we did with ii) an elicited representation that was the transcript of an interview with an accountant discussing statutory sick pay [Greene, 1987]. Again iii) a development was undertaken on paint selection procedures for industrial applications [Andries].

The effects of a larger domain and elicited representation will be examined in chapter 7. The small case studies raised several points for consideration, three of which were: Firstly, that the selection of the correct representation specification is crucial to the success of a development and that techniques for manipulating representations into a form that will match the systems requirements is an area of major importance to the expert systems field. Secondly, the use of a domain specification to independently reason about the knowledge is vital and thirdly, that the merger of the two representations into a concrete specification can be performed for any representation specification but for the concrete specification to be fully effective the most suitable representation specification is necessary.
CHAPTER 7.

A LARGE SCALE CASE STUDY IN EXPERT SYSTEM DEVELOPMENT.

7.1 Introduction.

Having produced a theory for developing expert systems, we tested it out upon several small domains and modified it accordingly after each one. A small case study is given in chapter six. The experiences gained in developing these cases were enormously beneficial but it was still felt that practical use of the development methodology could not be assessed until a domain several orders of magnitude larger was approached.

In this chapter we present a description of a large case study. It commences with the background to the case. This is followed by brief descriptions of each stage in the development process, the emphasis being on how the methodology applies to a large domain. The conclusions are then presented.

In describing the development of the system we will concentrate on the application of the steps which were described in detail earlier. Also, to add clarity, the full working details of some stages: elicited representation, domain specification, representation specification, concrete specification, plus some intermediate stages have been removed from the descriptive text and placed in the appendices.

7.2 Background to case study.

In examining domains for a large scale case study several criteria had to be present within a domain to make it viable. It had to be large enough to raise questions regarding the applicability of the development methodology to non trivial problems. Secondly, the subject matter had to be of sufficient complexity to be worthy of the term 'expert'. Thirdly, an expert had to be found who was prepared to enter into the project knowing that the project was part of a research environment realising little or no direct commercial gain.

A suitable domain area was jointly proposed by Professor M.J.R. Shave (Dept. of Computer Science, Univ. of Liverpool) and Mr. N. Gardner (Geology Fellow of St. John's, Oxford & Director of the UGC's Computers in Teaching Initiative Support Services): namely, the
problems associated with soil erosion. The domain experts were Prof. J. Thornes (Dept. of Geography, Univ. of Bristol) and Miss. A. Marchington (Doctoral candidate, Dept. of Geography, Univ. of Bristol) both of whom are currently active researchers in the area of soil erosion [Thornes. 1987] [Thornes. 1985] [Thornes. 1977] [Thornes. 1976] as well as being interested in the development of expert systems. A further knowledge engineer was also involved, Mr. J. Barber, a professional knowledge engineer and a member of the Knowledge Engineering group of Unilever Research PLC with whom the Department of Computer Science at The University of Liverpool have close links. His role was to aid in the knowledge elicitation process.

7.3 The Initial specification.

Introduction.

We will now begin by showing how an initial specification can be developed, from which the knowledge engineers can work during the elicitation process.

A detailed description of one possible approach to developing an initial specification is given in section 3.2, and it is these lines we shall follow here.

Step one. (Area outline)

This stage was completed through the exchange of correspondence by Gardner (who was acting as the domain expert's representative in the creation of the initial specification) and the two knowledge engineers (Barber & Plant). In this stage the domain was focused down to that of "soil erosion". With the following literature being used as introductory material: [Holy. 1980] [Kirkby. 1981] [Dadkhah. 1980] [Djorvic. 1980] [Cannell. 1979].

Step two. (Knowledge engineer(s) investigate domain literature)

The knowledge engineers having read the literature provided by Gardner plus some independent study then underwent a 'briefing' on the domain by Gardner. This allowed them to raise and clarify any points arising from their research. The briefing also allowed the domain expert (or his representative in this case) to ask questions regarding the approach likely to be taken by the knowledge engineers as well as general questions on expert systems.
and a.i. An aim of this stage was to promote thoughts on the limitations of the proposed system.

Stage three. (Prepare an initial specification)

In the third stage an iterative cycle occurs where the domain to be developed is proposed and clarified.

Due to time limitations the interactive cycles had to be rapidly accomplished and resulted in the knowledge engineers defining the domain to be: "Soil erosion that occurs in semi-arid environments on field sized plots where no wind erosion occurs", whilst the problem description is: "What are the causes of soil erosion that occur in semi-arid environments on field sized plots where no wind erosion occurs and what preventative measures can farmers take to overcome these problems". The details of this meeting were also reported back to the domain experts (Thornes & Marchington) by Gardner.

7.4 The knowledge elicitation phase.

Introduction.

The process of knowledge elicitation is that of extracting knowledge from the domain expert with only a little amount of consideration given to analysis of that knowledge. The following section describes how the elicitation process in the soil erosion domain occurred.

The knowledge engineers.

In this project it was decided that the use of two knowledge engineers would be advantageous. The reasons behind this decision were that the elicitation process in a complex domain can be highly demanding on one knowledge engineer and it was felt that the load could be better spread over two people. This situation then allows one of the knowledge engineers 'thinking time' while the other is proceeding with the elicitation. This is important as in, say, an interview situation the knowledge engineer has at least four tasks to perform in parallel:
- Listen to the domain expert.
- Relate the new information with the old.
- Formulate new questions.
- Check for consistency and completeness.

In such an interview it was felt that one of the knowledge engineers could act as the 'principal knowledge engineer' leading the questioning and discussion, whilst the 'secondary knowledge engineer' checked the information provided for completeness and consistency. The secondary knowledge engineer would summarise and raise points that were more general, attempting to place the information in context.

Choice of elicitation technique.

Having studied the information gained in the Initial specification the knowledge engineers decided to utilise the semi-structured interview technique. This involves allowing the domain experts to lead the direction of the interview once a few seed questions have been put to them. The knowledge engineers' role is to ensure that the domain expert does not digress too far from the major topic under discussion, and the knowledge engineer tries to regulate the depth the interview takes over a certain period of the elicitation.

This technique was chosen for several reasons: Firstly, we had the availability of two domain experts; secondly this method is likely to be one of the most widely used in industrial projects - the environment we were attempting to simulate.

It would have been possible to utilise the domain experts in other ways such as asking them to 'report' on tasks, for example when they performed modelling in tank tests. However, it was felt that interviewing would provide the widest range of elicited knowledge within the time constraints imposed. The formalised techniques such as repertory grid were not used as these methods work better in extracting specialised information structures and relations.

A full description of knowledge elicitation techniques is given in section 3.3 and includes suggestions of how each technique is best applied.
The domain experts.

The use of two domain experts as well as two knowledge engineers was slightly unusual. An aim of this was to encourage one of them to be the ‘principal domain expert’ taking the lead in answering the bulk of the questions; whilst the other domain expert could examine these answers - suggesting where points had been missed out as well as summarising and assisting the knowledge engineers on points they found difficult by rephrasing the information.

The elicitation process.

The knowledge elicitation process was carried out in two parts. The first part consisted of two, one hour interviews with both Thornes and Marchington answering questions and joining in the discussion. The second part also consisted of two one hour interviews, one with Thornes on his own, the other with Marchington on her own. The interviews were taped and later transcribed.

The elicited representation.

Having transcribed the interviews their size became apparent, there being approximately 40 pages (20 thousand words) in all. Thus, it was decided to concentrate upon one session of the elicitation and develop that through to a concrete specification. The selected elicitation was that of session three - the knowledge engineers interviewing Anne Marchington on her own.

This interview was selected for several reasons: Firstly, the domain expert was by this stage more aware of the elicitation procedure being used by the knowledge engineers than in the previous two interviews. This helped to relax the expert and aid elicitation. Secondly, the area of domain covered was the same as the first two interviews, except that this session elicited as much if not more information than the sum of the previous two. Thirdly, the final interview with Thornes resulted in an elicitation session where the subject under investigation changed away from the practical problems of soil erosion to the theoretical ones of modelling and research into soil erosion. This was not strictly within the boundaries of the initial specification.
The fact that the interview with Marchington was more productive than the previous two matched with expectations gained from previous experiences of interviewing, in that knowledge engineers and domain experts build up a working relationship over a period of time making each session increasingly productive. Barber states that "It is often the case that the lack of this working relationship can prevent the early sessions eliciting as much information as would be desirable and so these sessions are often not used as the main basis of the elicitation" [Barber:personal communication].

Thus, in our 'accelerated environment' of elicitation we had to take these factors into account when selecting the transcript to work from.

The elicited representation for the interview with Anne Marchington is given in appendix C.1.

Comments and conclusions to the knowledge elicitation phase.

During the knowledge elicitation phase it soon became apparent that there were two approaches to performing the task. These approaches stemmed from the assumptions the knowledge engineer made regarding what was the pre-requisite amount of knowledge needed (by the knowledge engineer) in order to perform the elicitation process.

One possible approach was for the knowledge engineer to read the introductory literature acquired during the initial specification phase in order to gain a high level view of the proposed domain and in so doing, isolate the key concepts.

The aim of this approach was to go into the knowledge elicitation session knowing only enough to prompt the domain expert who would then act as the main driving force in the elicitation session. The knowledge engineer would use any feedback to encourage, refocus and help the domain expert expound his knowledge.

An alternative view is for the knowledge engineer to place a strong emphasis on the importance of the introductory literature and research into the domain. This would help the knowledge engineer relate to the domain expert at both the conceptual and deeper specific levels.
Both of these views have their advantages and disadvantages. The first approach of a naive knowledge engineer is based upon the assumption that the knowledge engineer will not be swayed by the literature into false value judgements, also that he will not lead the domain expert off into only those areas he, the knowledge engineer is familiar with. The aim of having these two assumptions is to produce an unbiased elicitation session where the knowledge engineer is lead by the domain expert who guides him along suitable paths into the most applicative areas of the domain. This however seems to have a flaw, in that the assumption is made that the domain expert knows his own meta knowledge completely along with the inter relationships of that knowledge. This can be a dangerous assumption to make as often this is simply not true and can lead to unproductive elicitation.

The approach in practice works through the knowledge engineer viewing the process as acquiring strips of the expert’s knowledge, looking at the whole domain a layer at a time and starting off at the high level knowledge - general strategies, general ideas etc., this being rich in conceptual knowledge and low in specific factual knowledge. Having gained the knowledge at that level the knowledge engineer can then either proceed through developing a whole strip at a time or by taking a part of the whole domain and developing it to a suitable depth. Again the assumption is made that the domain experts know what knowledge they use and which of it is important in the problem solving process. In an ideal situation this may be so, but if the knowledge engineer relies only on the information given to him at a higher level, with which to prompt the expert, then this can cause elicitation problems.

In the second approach, it is seen that the knowledge engineer should have a major influence on the direction that the elicitation should proceed. This is not to say that he has a rigid set of questions from which he can not deviate, but rather that he has a set of goals he would like to achieve. As with the previous approach the knowledge engineer can start his elicitation at a high level which can lead onto a deeper search of the expert’s knowledge. In this second approach the emphasis is that if the knowledge engineer has a more knowledgeable starting point then this can cause the initial stages of the elicitation process to proceed more rapidly and that the knowledge engineer and domain expert can spend more
time discussing the specifics of the problem domain. The knowledge engineer can also use his knowledge to provide knowledge clues upon which the domain expert can act. It has been well documented in the literature that the human memory works better with memory cues [Leplat, 1972] (also see section 3.3). All of these points are especially important when the expert's time for elicitation is limited.

During the four knowledge elicitation sessions both of these approaches were applied to the domain, the results bearing out many of the points given above.

Practical knowledge elicitation techniques.

During this and other case studies, several techniques that aid the elicitation process were developed and are now outlined.

A useful way for the knowledge engineer to help the domain expert understand the problem to be overcome is for the knowledge engineer to provide a mental model of the projected system. This can be done by instructing thus:

"We pretend that you were the computer and that I (the knowledge engineer) was coming along to you and saying that 'I have this problem', what would you have to ask me in order to tackle it?"

The use of such a model promotes discussion and enables the knowledge engineer to see the knowledge engineers' perspective.

During interviews it is important for the knowledge engineer to be aware of the alternative directions that the discussion can take. One way of doing this is by asking the domain expert "what are the possibilities - at this point / causing this factor to occur /..." by doing this the knowledge engineer obtains a list of choices or alternatives that could occur at this point and of course be developed later on. The usual procedure is to take each of the possibilities in turn and then expand upon them to the depth required.

The questions asked of the domain expert by the knowledge engineer are often introspective, but the answer given is to a large extent based upon the information content within the question itself. For example, if the question was:
"we have talked about factor X, what else is important?"

this would give rise to a more general answer than:

"we have talked about factor X, what is the next most important factor?"

which is a more focused and specific question. Some of the information revealed may contain negations, for example:

"X will occur, Y will occur but not Z"

which can provide a useful seed to work from - raising the question:

"what if Z did occur"

It is important to spot negations as exceptions to a rule, case or situations are harder for an expert to identify than positive statements.

In response to generalities within answers these can be turned to advantage by asking the domain expert to elucidate upon them, this will help provide the factual knowledge that can underlie the more abstract, conceptual knowledge.

It is often natural for the knowledge engineer to think in terms of conditions when performing elicitation, in which case the knowledge engineer should try and think of his question in two parts - the condition and the antecedent. Then take one half and think "what knowledge is needed to justify this".

The level of information gathered is often important and the knowledge engineer must be wary of whether the domain expert is covering all levels. For example, it may be that the domain expert 'forgets' to include the knowledge he considers 'trivial', in which case this must be looked for.

Conclusions.

The knowledge elicitation phase of the development was carried out successfully and produced satisfactory results. The use of two knowledge engineers also proved to be highly beneficial, increasing both the productivity and quality of the information obtained. The use of two domain experts was also beneficial. It was felt that the experts were best utilised if they were interviewed both individually and together over a series of
sessions. This gives rise to the possibility of overlapping within the information gained, but comparison of answers can only raise the quality of the information elicited. The use of two domain experts will also increase the breadth and variety of knowledge elicited. When the elicitation involves both experts then the quality of the information extracted will be especially high.

7.5 The Primary analysis phase.

The role of the primary analysis stage is to take the elicited representation and by following guidelines produce the most suitable primary representation.

The method of producing a primary representation is given in chapter four but can be summarised as follows:

1. The elicited representation is examined for the presence of the following characteristics: noise, modularity, linkage, operational types, sequencing and explanations. A graph showing the amount of these characteristics present in the elicited representation is then drawn and this is known as the primary trace.

2. The primary trace is then compared with the bandwidth diagrams of the primary representation in order to decide which is the most suitable, this process is known as trace matching.

3. Finally, the elicited representation is transformed into the chosen primary representation formalism.

In section 4.1.8 ("primary analysis in practice") we described the problems associated with analysing large elicited representations and how best to overcome these problems. These can be summarised as follows:

The process of trace creation, matching and selection of primary representation is based upon the assumption that the elicited representation is either small enough so that the trace is drawn for it is valid for the whole of it or that the elicited representation is uniform in character throughout its entire length.

A proposed solution to the problem encountered when the elicited representa-
tion is large in size or has variable characteristics over its length is to partition the elici-
ted representation into blocks of roughly similar length. Traces can then be drawn for each
block, followed by the other primary analysis phases.

The size of the elicited representation in this study meant that it was necessary to use these techniques.

The first step is to label each adjacency pair in the elicited representation, where an adjacency pair is one knowledge engineer - domain expert interaction. This is done for several reasons. Firstly, the pairs can be numbered which helps in the referencing of information. Secondly, most of the interaction is between a single question and a following adjoining answer, with less frequent referrals back to previously stated information or to information yet to be presented (i.e., just before I state the objections to the motion I will discuss the positive points...). The adjacency pair is then a useful basic unit of text around which to work. It also allows access to theory from the world of communication and linguistics with which to analyse it.

The second step is to partition the adjacency pairs into blocks, the method used in this study being binary decomposition (see section 4.1.8). The total number of adjacency pairs was 43, and this was then divided by two giving integer values of 21 and 22 as our block sizes. The process was then repeated giving four blocks of 10, 11, 11 and 11 adjacency pairs respectively. At this point the subdivision process was stopped as it was felt that these were workable block sizes with stable characteristics.

Diagrammatically this can be viewed as:

<table>
<thead>
<tr>
<th></th>
<th>43</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

fig. 44.
The listing of blocks one - four are given in Appendix C.2

Trace creation.

In this stage we examine the characteristics of each block of the elicited representation in order to draw its trace. We will now consider each block in turn.

A trace for block 1.

Noise.

The elicited representation is virtually noise free. This is due to the clarity of the speaker. Thus, the value minimal will be assigned.

Modularity.

The modularity is significant. This is because in a significant proportion of the adjacency pairs the answer is a direct response from the immediately proceeding question. The level is not higher as the questions start off being quite general evoking general answers.

Linkage.

The linkage is low. The only connection between modules is through the generalities of the early questions.

Operational types.

The block is composed of mainly test, control, action and diagnosis types. This assigns the value intermediate to this parameter.

Sequencing.

There is an intermediate amount of sequencing. This is because the adjacency pairs do sequentially (temporally) flow from one to the other. If the flow had been lower (more disjoint) then a low or minimal value would have been assigned. If there had also been forward and/or backward referencing of adjacency pairs then this would have raised the value to significant or high.
Justification.

There is a low level of justification. This leads from only approximately one fifth of the adjacency pairs containing any form of justification. If all the pairs are justified then this tends towards a high value, if half then this approximates towards intermediate whilst below one fifth the value would tend towards minimal.

Explanation.

There is an intermediate level of explanation given to queries as approximately half of the adjacency pairs are associated with explanations.

Having examined this block of the elicited representation for these characteristics it is now possible to draw the following primary trace:

![Diagram showing levels of noise, modularity, link, and justification]

fig. 45.

Trace for block 2.

Noise.

The noise level of the block is minimal.

Modularity.

There is significant amounts of modularity. The block contains adjacency pairs which are modular in nature.
Linkage.

An intermediate level of linkage is found. The modularity is not always restricted to one adjacency pair, a subject being carried over into 2 or more of the pairs. The linkage between subjects is also an influencing factor.

Operational types.

The number of operational types is low.

Sequencing.

The sequencing information content is low as most of the information is concerned with short descriptive passages in response to queries.

Justification.

There is an intermediate level of justification within this block as responses to question get qualified by justifications.

Explanation.

There is only a low level of explanation given in attempting to expand upon answers given to questions.

This gives us the following trace:

![Diagram of trace levels]

fig. 46.
A trace for block 3.

Noise.

There is only a low amount of noise in the block, this stems mainly from two adjacency pairs being less than complete.

Modularity.

The modularity is significant. This is because the block deals with several points but the information is kept within adjacency pairs.

Linkage.

The linkage between modules is significant. This is because the block deals with mainly one subject (animal husbandry). This subject is divided into modules but there are good linkages leading from one module to another.

Operational types.

There is an intermediate level of operational types in this block. This value is given because action, conditional, numerical and descriptive types are present.

Sequencing.

The block has an intermediate value for knowledge regarding how the information is to be sequenced, this is due to the profitable use of techniques such as ‘recapping’, ‘clarifying’ and ‘cross referencing’ by the knowledge engineer in this block.

Justification.

The amount of justification within the block is significant as an attempt to justify most of the points raised has been given. The amount given to this factor is not higher because some of the justifications are not very strong.

Explanation.

The emphasis in this text was placed more on justifications and the need for full explanations of the areas were not given. This block thus has only a low level of explanation.
This gives us the following trace.

A trace for block 4.

Noise.

There is a low level of noise in this block caused through repetitions within adjacency pairs and also unfinished sentences.

Modularity.

The modularity of the knowledge is low in that there are several points raised during the block, however, these points are discussed throughout its length as opposed to being composed of self-contained, compact modules.

Linkage.

The linkage of the modules is low. This is only to be expected as the modularity level is low. The information is well connected because of the discursive nature of the block.

Operational types.

There is a low level of operational types, the block containing mainly non-numeric tests and descriptions.
Sequencing.

The level of sequencing is intermediary as some of the adjacency pairs (those discussing drought and rill with cattle) contain information on the sequencing of operations that may change the state of the land.

Justification.

The level of justification is low as the operation types are mainly concerned with just giving premises. They are not followed up by actions to take and as a consequence the justifications are only of the premises.

Explanations.

There is a significant amount of explanation in the block giving explanations of points raised within adjacency pairs.

This gives us the following trace:

```
Trace matching.

In the fourth step the primary traces are compared to the bandwidth diagrams of the primary representations to decide which is the most suitable - this is known as trace matching.
```
Decision tables.

The bandwidth diagram for decision tables is:

```
High
Significant
Intermediate
Low
Minimal
```

Having created the traces for each block and reviewed the bandwidths of the three primary representations, the next stage is to find the most suitable representation for each block. Taking each in turn:

Trace matching block 1.

The closest match is with the contour diagram. The flow diagram was the second best match. There was however a negative deviance on the part of the elicited representation regarding linkage of the modules.

Trace matching block 2.

The closest match is with the contour diagram. The second closest match was with the flow diagram, there were however negative deviances in both linkage and sequencing on the part of the elicited representation.

Trace matching block 3.

The trace for block three matches with bandwidths for both the contour diagram and flow diagram. The flow diagram was selected due to it having a closer match.
The bandwidth diagrams.

We will first of all look at the bandwidth diagrams for the primary representations (section 4.1 gives justifications regarding their shape).

Flow diagram.

The bandwidth diagram for the flow diagram is as follows:

![Flow diagram diagram](image)

Contour diagram.

The bandwidth diagram for the contour diagram is:

![Contour diagram diagram](image)
Trace matching block 4.

The closest match is with the contour diagram.

Creating the primary representation.

The matching process has shown that three of the blocks are to be represented by contour diagrams whilst the fourth is to be represented by a flow diagram.

In this section we shall describe the development of one of these contour diagrams and the flow diagram. The resultant primary representations are then developed on to the primary analysis stages.

The first step is to create our primary representations. The techniques for transforming elicited representations into primary representations are discussed in chapter four but can be summarised by the following guidelines:

Elicited representation to contour diagram.

Guideline 1.

The transcript is partitioned into a series of knowledge engineer - domain expert, action response pairs. The pairs are numbered and then marked off from each other.

Guidelines 2 - 15.

Having broken the elicited representation down into adjacency pairs these can then be analysed using the following guidelines in order to produce a contour diagram:

2. Decide upon a set of connectives to use in the decomposition process,
3. Compile a list of keywords to use in the decomposition process,
4. Decide upon a criteria for filtering noise from the text,
5. Remove noise from the text,
6. Examine the knowledge engineers text for conversational coherence and alignment:
   1. Accounts,
   2. Formulations,
   3. Meta talk.
3.1. Clarifying.
3.2. Remediating.
3.3. Directing.
3.4. Requesting.
3.5. Agendizing.
3.6. Side particles.

4. Side sequences.
5. Meta communicative digressions.
6. Digressions.
7. Qualifiers.

7. If necessary break the text into a series of smaller segments approximately a paragraph in size.

8. Decompose the paragraphs into sentences.

9. Take each line in turn and check it for referential characteristics of the form:

   1. That.
   2. Because.
   3. When.
   5. What.

   and where necessary add text that helps to identify the reference.

10. Based upon the position of the keywords and connectives, partition the sentences up into subsentences - removing the connective in the process. Repeat the partitioning until the sentences are "singularities". (contains no connective or keyword) or have only connectives which help form a list of conditions or items. Whilst this is being performed a careful note of the nested structure of the decomposition must be made.

11. After the whole text has been decomposed, the numbering and indentation can be used to create a contour diagram.

12. Examine each section of the contour diagram and try and assign it a type:

   1. Noise.
   2. Sequencing information.
   3. Facts.
   4. Rules.
   5. Queries.
   7. Explanations.
   9. Qualification of other types.
some sections will not be coherent or will be incapable of having a new type invented for them, and these get classified as noise.

13. Remove noise.

14. Each section of the text in the contour diagram is then replaced by a more concise definition.

15. Take the distilled information and draw out as much detail as possible, listing all of the definitions for each type together.

Due to the size of the blocks only the first one was transformed into a primary representation. The analysis of which can be found in Appendix C.3.

Comments on creating the contour diagram.

The resultant contour diagram and associated information list is given in Appendix C.4

Having successfully created the contour diagram we can make the following observations. Firstly, that the contour/indentation effect of the representation does enable a form of structuring to be given to the information and helps preserve the original semantics until the knowledge engineer can create the final distillation of the knowledge. Secondly, that the creation process is a long exercise and involves a large volume of textual manipulation. This has to be scaled up by the number of iterations necessary in order to achieve the final contoured state with its well defined objects. The process would be eased considerably had an automated software environment been available. The creation of such a system is beyond the scope of this work. However a system such as KEATS [Motta. 1986] could be amended and utilised to speed the process up. If the text manipulation process is allowed to take too long, much of the impetus behind the development could be lost.

The guidelines for transforming each stage to the next were found to be comprehensive in their coverage of the situations encountered during the process. This can be attributed to the variety of domains developed in earlier ‘experimental’ case studies.

It may be useful for the guidelines to be extended to include example situations which
could act as 'precedence' for future transformations. It may even be feasible to build some of
the guidelines into a small expert system that analysed the text and suggested the form a con­
tour diagram may take and offer advice at each transformation step. However the development
of such a system is beyond the scope of this work.

Elicited representation to flow diagram.

The following guidelines are a suggested way to produce flow diagrams from text:

1. Decide on any criteria that may be needed for the removal of noise.
2. Remove any noise.
3. Take a small segment of text and break this down into the components of the flow
diagrams and give them types.
4. Construct a separate flow diagram for each of the textual points.
5. Unify the separate flow diagrams into one large diagram.
6. Examine the structures for repetitions and redundancies.
7. Examine the structures for deficiencies i.e., undefined paths from decision boxes, incom­
completed actions upon diagnosis of problems.
8. Recheck elicited representations to improve the deficiencies.

A full discussion of these guidelines and their application can be found in section 4.2.

The flow diagram for block 3 was created by following these guidelines and can be
found in appendix C.5.

Comments on creating a flow diagram.

The creation of the flow diagram drew our attention to the fact that when one or more of
the primary trace characteristics is on the borders of the bandwidth diagram then this indicates
that extra care should be taken in the development of the primary representation. In this case
the low value for the sequencing of the knowledge caused the development to be slower than if
an abundance of sequencing information were available. However, once the flow diagram has
been obtained then the advantages of this temporal sequencing are obtained, which are not
available in all other representations.

7.6 The secondary analysis stage.

Having created our primary representation we can now develop the system towards the domain and representation specifications.

Domain specification.

The domain specification is given in appendix C.6.

It was found that the process of creating a domain specification from the contour diagram was easier than from the flow diagram. This is because of the more explicit way in which the contour diagram ultimately expresses the knowledge and information. The flow diagram even though it has a type structure, caused the section of the domain specification created from it to undergo several iterations until it was felt to be satisfactory.

The use of a formal specification was found to be very advantageous in unifying the knowledge from the two primary representations. It enabled consistency problems to be checked along with other verification measures. If no formal domain specification was used then the process of creating a maintainable knowledge base from two (or more) primary representations would be most difficult.

The domain specification did not cause the "Z" notation to be utilised in any more than a conventional manner. However, the specifications did grow to a size where it would be convenient to utilise some form of development tool or environment to create schemas, definitions and validate the specification. Tools for this purpose are currently under development at The Programming Research Group, Oxford.

The representation specification.

The next step in the development methodology is to identify which (if any) classical or hybrid representation is the most suitable for use as a representation specification.

In order to find the most suitable form several stages have to be gone through:
1. Creation of knowledge profiles for the elicited knowledge contained within the primary representation. This is known as profile analysis.

2. Comparison of the domains knowledge profile (created in stage one) with the profiles for the classical representations. This is the profile matching process.

3. Selection of the most suitable form around which to base the representation specification. These will now be followed in an attempt to select the most suitable representation.

Profile analysis (Stage 1).

By considering definitions of knowledge types as well as guidelines for constructing a profile the following were constructed: The block represented by a contour diagram gives:

![Diagram](image)

fig. 52.

The block represented by a flow diagram gives:
The next step is to merge these two profiles to form one profile that can be used in the matching process. This is done such that the largest value for each of the knowledge types is taken from the individual profiles to form this total domain profile. This then represents the 'total' knowledge needs of the domain.

By doing this we achieve:

We can now undertake the matching process.
Profile matching (Stage 2).

The aim of this stage is to compare the knowledge profile for the domain with profiles of classical representations. These representational profiles are created by firstly having a standard representation of a given classical form and then examining it for the ability to represent the five knowledge types.

From these analyses the profiles are drawn.

Frame profile.

The following frame profile is based on the frame based language described in [Plant. 1985].

![Frame profile diagram](image)

**fig. 55.**

Rule profile.

The following rule based profile is based on the rule based system given in [Barr. 1981], where the knowledge base is purely a set of rules.
Semantic network profile.

The following semantic network profile is based on the procedural semantics for semantic networks given by Levesque and Mylopoulos [Levesque, 1979].

Representation selection (Stage 3).

Having drawn the domain profile and examined the representation profiles the selection of the most suitable representation can now be considered.

In order to make the selection easier the domain and representation profiles are overlaid and divergences are examined.
Comparison one.

Frame representation profile and domain profile.

High
Significant
Intermediate
Low
Minimal

Comparison two.

Rule profile and domain profile.

High
Significant
Intermediate
Low
Minimal

Comparison three.

Semantic network profile and domain profile.
Selection criteria.

The profile matching showed that none of the three representations were perfect matches for the domain, all of them having some deviations in their profiles. In order to try and roughly measure the deviances, the number of units difference were taken and then compared to the following scale:

- High: > 5 units
- Significant: 4 units
- Intermediate: 3 units
- Low: 2 units
- Minimal: 1 unit
- Match: 0 units

This gives us the following results:

frame match:
- negative deviance = 1 = match
- positive deviance = 7 > high

Rule match:
- negative deviance = 1 = match
- positive deviance = 3 = intermediate

semantic network match:
- negative deviance = 1 = minimal
- positive deviance = 10 > high

A further analysis can be performed by examining the deviances for each knowledge...
We will now utilise the following guidelines to select an appropriate form for the representation following the guidelines given in section 5.3.7.

**Selection of the representation specification.**

The selection of the representation specification is performed by following the guidelines given above. Firstly, the profiles are overlaid and the divergencies counted. This showed us that there were no perfect matches for both positive and negative deviations for a single representation. The next step was to follow guideline six and discard those matches containing high deviations, this left us with the rule match as the most appropriate form to act as the representation specification.

The representation specification is given in appendix C.7

**The concrete specification.**

Having specified the system in formal terms within the domain specification and having selected a rule based representation as a suitable form to model the system, these two forms are brought together in the concrete specification.

The concrete specification is given in appendix C.8
7.7 CONCLUSIONS.

The aim of this case study has been to show that the development methodology can be applied to large scale problems within a complex domain.

We feel that this is true as even though the study was carried out under less than ideal time considerations the methodology proved to be i) robust and ii) resulted in a high information yield. Robust in the sense that the guidelines and methodology as a whole covered a high percentage of the situations that occurred in the process of developing the case study. We would not expect it to cover all situations but the study showed that the guidelines covered the principle problems that occur. The second point regards the amount of information distilled out of the elicited text, it is our judgement that a high percentage of the available knowledge has been extracted. Fox [Fox. 1985] raises the point of uncertainty regarding information yield, and when discussing his attempt to analyse a protocol states:

"have we extracted all the knowledge possible from the transcripts? We do not know". [Fox. 1985]

This promotes the area of information yields from protocol analysis as one for future research - where different techniques could be tested against each other to try and establish what techniques elicit knowledge in which areas to what levels of sensitivity. We also feel that the methodology achieved this level of result in a creditable amount of time from the point at which elicitation was terminated (3 - 4 weeks). The case study has also shown that it is possible to produce a concrete specification purely from a transcribed interview with no other domain expert contact. This makes it especially valuable for use in industrial and commercial situations where the domain expert's time is strictly limited. The specifications could if desired be expanded from literary or other sources at a later date.

The next consideration for the methodology is that of applying it to a full scale industrial application. The most obvious need to assist in this task would be for automated environments to be built that would help the knowledge engineers cope with the volume of text manipulation and creation of specifications. We feel that if suitable tools were available then the methodology would create a system that accurately reflected the elicited knowledge, that was maintainable, documented and expandable, four features highly desirable of knowledge based systems
but unfortunately often missing.
CHAPTER 8.

SUMMARY AND CRITICAL COMMENT.

The aim of this thesis was to provide the foundations for a rigorous development methodology for the production of expert systems. This was necessary in order to offer an alternative to the existing practice in developing expert systems, which generally involves weak ad hoc approaches that have little or no theoretical basis.

The approach that we adopted was to adhere to three design principles: Firstly, that every step in the development from the definition of the problem domain to the implementation should be capable of justification. Secondly, that this path should have a point at which the knowledge of the domain, elicited from the expert, can be represented in an implementation independent form and thirdly, that a suitable form be chosen to represent the domain and that this choice be capable of justification.

The first stage in the development methodology is the creation of an initial specification. The role of this is to force the knowledge engineer to define as early in the development cycle as possible: the problem domain, its boundaries and any constraining factors under which the knowledge engineer or system will have to work.

The initial specification, even though it can be used as a base line document throughout the remainder of the development, is principally intended to assist the knowledge engineer in the extraction of knowledge from an expert source - a process we define as 'knowledge elicitation', as distinct from 'knowledge acquisition' which we define as the analysis and organisation of extracted knowledge.

The literature on knowledge elicitation is heavily orientated towards the field of experimental psychology which emphasises the cognitive aspects of elicitation, whilst the literature discussing the relationship between knowledge elicitation and KBS is limited. This forced us to consider some of the techniques available to the knowledge engineer. This investigation indicated to us that the knowledge elicitation technique to be used is in the first instance dependent upon two factors: firstly, the type of problem domain involved and secondly, the accessibility of the expert knowledge source. Once an accessible expert is located then other
factors need to be taken into account when selecting elicitation techniques, for example the
type or depth of knowledge may demand that a certain approach be taken. Our work in this
area indicates that in order to elicit knowledge for a non trivial domain a series of techniques
will need to be used as the knowledge to be elicited changes in type and/or depth. In order to
help the knowledge engineer, a set of guidelines is given for technique selection.

To assist in achieving a rigorous approach the results of our elicitation had to be in a
consistent format, upon which analysis could be performed. We therefore restricted the elici­tation
techniques that can be used in conjunction with our methodology to those capable of pro­
ducing a textual output - this output forming the elicited representation.

The knowledge contained in the elicited representation is refined by re-representing it in
a form that highlights its organisational structure - this form is known as the primary represen­
tation.

It is important that the primary representation selected to rerepresent the elicited
knowledge is a suitable one. In order to determine this suitability the underlying characteristics
of both the primary representation and the elicited knowledge need to be considered. Our
investigations show that the structural organisation of the elicited knowledge can be compared
to the capacity of a primary form to represent that structure. A selection procedure based upon
this principle can therefore be used.

As the primary representations change only the format of the knowledge, this helps
maintain the principle of minimising information loss and changes in semantic meaning. The
primary representations also show the value that the more visual form of representation has
over the plain textual one - for example, making some of the inconsistencies and incomplete­
ness easier to identify.

The production of a primary representation allows the knowledge engineer to move into
the knowledge acquisition phase of the development. It is in this phase that a major concept
behind our development methodology is utilised:- The need to separate the specification of the
elicited knowledge from the selection and specification of the (classical) form of representation
upon which the implementation will be based e.g., frames, rules semantic networks.
The aim behind the formal specification of the elicited knowledge - the domain specification - is to enable an unambiguous specification to be built that is complete and consistent. It is complete in the sense that every item of information present in the elicited and primary representations is also in the domain specification, and consistent in that if an item of knowledge is referenced more than once then the reference mechanism and data types are identical.

The use of a formal specification enables several other advantages to be obtained for those systems that utilise them. The specification can act as the authoritative unambiguous statement around which discussions can occur regarding the intention of the system - these are open to the domain expert, knowledge engineer and the implementer (although some training may be needed to read the specifications). The future maintenance of the system is enhanced by using a formal approach and this helps to overcome a problem that complicates the amendment of many existing knowledge bases.

The use of formal specification techniques for expert system development has received little attention in the literature and in chapter five we therefore gave a comparative discussion of the approaches open to the knowledge engineer.

The selection of a form in which to represent the knowledge has also received little attention in the literature with workers tending to advocate only one representation to be used in many problem situations. Our approach is to ensure that the representation fits the needs of the knowledge engineer rather than forcing the knowledge into a style of representation to which it is not suited.

In order to perform the matching process between the knowledge and the representation, it is necessary to consider what are the principle knowledge types underlying the elicited knowledge and what is the capacity of the classical forms such as frames, rules etc., for representing these knowledge types. Our research indicates that there are five principle knowledge types (factual, control, heuristic, procedural and conceptual) and it is based upon this that we give the first steps towards a taxonomy for knowledge typing.

The taxonomy is used to obtain an indication of the level of each principle knowledge
type within the elicited knowledge. These levels are then compared to the representational ability of several standardised classical representations and a suitable form of representation is selected. If none is found to be suitable then this can also be determined, indicating the need for a special representation to be developed. The definition of the syntax/semantics of the selected classical representation is known as the 'representation specification'.

The domain and representation specifications are combined to form a secondary representation in which the knowledge from the domain specification is represented in the form advocated by the representation specification. This process will be a relatively straightforward and natural one as the representation should accommodate the knowledge types of the domain specification.

The knowledge engineer then has to specify the control architecture to manipulate the secondary representation. This process is complicated due to the specific needs that the individual domains and their associated characteristics demand. The combination of secondary representation and control architecture form the composite object known as the concrete specification, the aim of which is to act as a point from which a system can be implemented.

The development methodology was tested through several small case studies, one of which is given in chapter six, and a large case study presented in chapter seven. The results obtained from these studies indicated several points:

Firstly, that the theory and guidelines could be applied successfully to a variety of domain problems.

Secondly, that although some aspects of the development were time consuming e.g., the transcription process and the analysis process, the increased formality of the results allowed other aspects to be performed quickly and confidently e.g., the creation of the secondary representation.

Thirdly, that the rigor of the development process assisted system upgrades and maintenance. For example, if the system needed to be expanded, the new elicited knowledge could in the initial stages be developed independently of the existing knowledge. The new knowledge could then be added to, and checked against the domain specification whilst the representation
specification was checked to see if its form needed amending - finally if necessary the concrete specification could be amended and the implementation updated. The whole process is performed in a controlled rigorous manner and was illustrated in the large case study of chapter 7.

The case studies also helped to identify areas in which further research would be advantageous. One direction would be an investigation of software tools and environments to support a more automated development of a system, through our methodology. Several software tools that could be adopted or amended to initiate this work are currently in use or under development at various research centres. For example, the KEATS system [Motta. 1986] assists the knowledge engineers to manipulate textual information, whilst "B" and "Forsite" are systems (currently research projects at the Programming Research Group, Oxford) intended to assist software engineers develop and check "Z" specifications.

It would be instructive to take tools of this sort and by using the a.i. tools provided in a system such as KEE, build a development environment for the methodology. For example, figure 63 shows how one such configuration might be linked.

In this configuration the elicitation is executed in a KEATS like environment, the analysis of the elicited knowledge controlled by a series of tools built in KEE, the specification is controlled by Forsite and the implementation is controlled by KEE processes. The separate tools are co-ordinated through a front end built in KEE.

Automation of part or all of the development path may also have the desirable conse-
quence of strengthening the theoretical basis for the guidelines, helping to decrease the subjectivity that is currently present within some aspects of the methodology.

The need for an automated environment or intelligent assistant is high, but there are several fundamental issues which also need to be examined in order to provide a firm theoretical base for such a system.

There needs to be an investigation of elicitation techniques and their applicability to the extraction of knowledge for expert systems. We have in this thesis surveyed many of the available techniques and given a series of initial views, drawn from our experience, on which technique is suitable for which situation. This needs to be followed up by qualitative reviews and experiments - perhaps in conjunction with researchers from other fields such as experimental psychology.

The area of control architectures is also in need of work to determine which mechanism is suitable in which circumstance. A series of references were presented in section 5.4.3 that give the early results of work in this area.

In the thesis we have examined and given an approach to representation selection and this is also an area in need of further research. It would be valuable for research to be undertaken in defining representations at all levels - primary, classical & secondary. For if the characteristics of these representations were formally defined, along with less subjective measurement techniques, this would allow the knowledge engineer greater precision in the selection and use of representations. We have taken a first step towards this by using three standard representations and discussing their characteristics. The next step may be to develop and strictly define the semantics of more and varied representations such as hybrid forms, apply case studies to them and analyse the results.

Another area in which basic research could be useful is that of investigating information yields for the different analytical techniques applied to elicited knowledge. We feel that the techniques used in our methodology produce a high information yield. If the reasons behind these yields could be formally defined this could lead to better or more refined techniques.

The emphasis throughout this need for fundamental research is that of "justification", it is
vital that design decisions in a development can be theoretically justified.

In conclusion, we have attempted to demonstrate through this work what would be a reasonable and achievable set of objectives for an expert system methodology to have. These can be summarised as:

i) Have a high degree of rigour.

ii) Justify each stage in the development.

iii) Justify the transformation of knowledge between stages.

iv) Specify the knowledge in a formal implementation independent way.

v) Select a representation for that knowledge and justify the selection.

vi) Be applicable to a variety of real world problem.

We see the work contained within this thesis as providing the foundations of such a methodology. Further research could lead to the production of an automated development environment which would strengthen the guidelines in the methodology, increase its rigour and decrease the time needed to develop expert systems.
REFERENCES

Addis, T.R. 1985
'Designing knowledge based systems'.
Kogan Page.

'The uses of metaknowledge in ai systems'.
In, ECAI 84. Advances in Artificial Intelligence.

'Knowledge level engineering: Ontological analysis'.

Aho, A.V., Hopcroft, J.E. & Ullman, J.D. 1980
'Data structures and algorithms'.
Addison-Wesley.

Andries, H., Delaney, J.D.P., Irving, G.N. & Medling, H.K.
'Guidelines on paints, Zinc- and Aluminium - Rich coatings'.
Unilever Research Labs, Internal document.

Bainbridge, L. 1979
'Verbal reports as evidence of the process operator’s knowledge'.

Barr, A. & Feigenbaum, E.A. 1981
'The handbook of artificial intelligence'.
Pitman press.

Bennett, J.S 1981
'On the structure of the acquisition process for rule based systems'.
In, Machine Intelligence: Infotech state of the art report,
Series 9, No 3. Pergamon Infotech Ltd, Maidenhead, Berks, pp 189-204.

Berry, D.C. & Broadbent, D.E. 1984
'On the relationship between task performance and associated verbalised
knowledge'.

Bramer, M.A. 1986
'Expert systems in Britain: Progress and prospects'.

Buchanan, B.G., Sutherland, G.L., & Feigenbaum, E.A. 1969
'Heuristic DENDRAL: A program for generating explanatory hypotheses in
organic chemistry'.
In, B.Meltzer and D.Michie (Eds.), Machine Intelligence 4. Edinburgh,

Buchanan, B.G., Sutherland, G.L., & Feigenbaum, E.A. 1970
'Rediscovering some problems of artificial intelligence in the context
of organic chemistry'.
In, B.Meltzer and D.Michie (Eds.), Machine Intelligence 4. Edinburgh,
Edinburgh University Press, 253-280.

Buchanan, B.G., Barstow, D., Bechtal, R., Bennett, J., Clancy, W.,
Kulikowski, C., Mitchell, T., & Waterman, D.A. 1983
'Constructing an expert system'.

Buchanan, B.G., 1986
'Expert systems: Working systems and the research literature'.

Cannell, G.H., & Weeks, L.G. 1979
'Erosion and its control in semi-arid regions',
In, A.E. Hall., G.H. Cannell., H.W. Lawton (eds.) 'Agriculture in semi-arid environments'.
Springer.

Clancy, W.J. 1979
'Transfer of rule based expertise through a tutorial dialogue'.
Report No. STAN-CS-769, Computer Science Dept., Stanford University.

Clancy, W.J. 1983
'The Epistemology of a Rule-Based Expert System - a Framework for Explanation'.

'Programming in prolog'.
Springer Verlag.

Codd, E.F. 1970
'A relational model of data for large shared data banks'.

Constantine, L.L., Myers, G.J., & Stevens, W.P. 1974
'Structured design'.

Cottam, I.D. 1984
'The rigorous development of a system version control program'.

Cunny, X. 1977
'Etude de l'activite de regulation dans une situation de conduite de processus'.

Dadkhah, M. & Gifford, G.P. 1980
'Influence of vegetation, rock cover and trampling, an infiltration rates and sediment production'.
Water resources Bull. 16.

Darlington, J. 1982
'Program transformation'
In, Mathematical logic and programming languages,

Date, C.J. 1981
'An introduction to database systems'.
Addison Wesley.
Davis, R. & Lenat, D. 1982
'Knowledge based systems in artificial intelligence'.

Dietterich, T.G. & Michalski, R.S. 1983
'A comparative review of selected methods for learning from examples'.
In, Machine Learning, (Eds., R. Michalski, J.G. Carbonell, & T.M. Mitchell)
Tioga, Palo Alto, Calif.

Dijkstra, E.W. 1976
'A discipline of programming'.
Englewood Cliffs. Prentice Hall.

Djorvic, M. 1980
'slope effect on run-off and erosion'.
In, M. de Boodt and D. Gabriels (eds.,): 'Assessment of erosion'.
Wiley.

Drayfus, H.L. 1965
'Alchemy and artificial intelligence'.
P-3244, The RAND Corporation, Santa Monica, California.

Duda, R.O., Gashnig, J., Hart, P.E., Konolige, K., Reboh, R., Barrett, P.,
& Slocum, J. 1978
'Development of the PROSPECTOR consultation system for mineral exploration'.
Final report, SRI Projects 5821 and 6415, SRI International Inc., Menlo
Park, Calif.

Englemore, R.S., and Terry, A. 1979.
'Structure and function of the CRYSAIS system'.
IJCAI 6, 250-256

Ericsson, K.A. & Simon, H.A. 1980
'Verbal reports as data'.

Feigenbaum, E.A. 1979
'Themes and case studies of knowledge engineering'.
Edinburgh University Press.

Feigenbaum, E.A., & Fieldman, J. (Eds.) 1963
'Computers and thought'.
McGraw-Hill.

Flanagan, J.C. 1954
'The critical incident technique'.

Floyd, C. 1983
'A systematic look at prototyping'.
In, Approaches to prototyping, (Eds., Buddle, R., Kuhlenkamp, K.,

Floyd, R.W. 1967
'Assigning meanings to programs'.
Proceedings of a symposium in Applied mathematics.
Forgy, C.L. 1981
'The OPS5 user's manual'.
Technical report CMU-CS-81-135

Fox, J., Myers, C.D., Greaves, M.F. & Pegram, S. 1985
'Knowledge acquisition for expert systems: Experience in Leukaemia
diagnosis'.

Gammack, J. & Young, R. 1985
'Psychological techniques for eliciting expert knowledge'.
In, Research and development in expert systems, (Ed., M. Bramer),
Cambridge University Press.

Gashnig, J. 1979
'Performance measurement and analysis of certain search algorithms'.

Goldberg, J.A. 1983
'A move towards describing conversational coherence'.
In, Conversational coherence (Eds. Craig, R.T. & Tracy, K.)
pp 10-25, Sage.

Gordon, M.J.C, 1979a
'The denotational semantics of programming languages'.
Springer Verlag; New York.

'Edinburgh LCF Vol VIII'.
Lecture notes in computer science. No. 78.
Springer-Verlag.

Greene, C.B. 1987
'An advisory expert system on statutory sick pay'.
BSc Honours project, Dept. Comp. Sci. University of Liverpool.

Grover, M.D. 1983
'A pragmatic knowledge acquisition methodology'.
IJCAL 8. pp436-438

Hart, A. 1985
'The role of induction in knowledge elicitation'.

Hayes, I. 1985
'Specification case studies'.
Technical monograph PRG-46
Computing Laboratory, Oxford University.

Hayes, I. 1985b
'Specifying the CICS Application Programmer's interface'.
Technical monograph PRG-47
Computing Laboratory, Oxford University.

'Building Expert Systems'.
Addison-Wesley.
Hehner, E.C.R. 1984
'The logic of programming'.
Englewood Cliffs. Prentice Hall.

'The lispkit manual Vols. I & II'.
Technical monograph PRG-32 (I)(II)
Computing Laboratory, Oxford University.

Hoare, C.A.R. 1962
'Quick sort'.

Hoare, C.A.R. 1969
'An axiomatic basis for computer programming'.
CACM. Vol 12, 576-583.

Hoare, C.A.R. 1982
'Programming is an engineering profession'.
Technical monograph PRG-27
Computing Laboratory, Oxford University.

Hoare, C.A.R. 1983
'Notes on communicating sequential processes'.
Technical monograph PRG-33
Computing Laboratory, Oxford University.

Holy, M. 1980
'Erosion and environment'.
Permagon.

Hughes, S. 1986.
'Lisp'.
Pitman Press.

Jackson, M.A. 1975
'Principals of Program Design'.
Academic Press.

Jacobs, S. & Jackson, S. 1983
'Speech act structure in conversation: Rational aspects'.
In, Conversational coherence (Eds. Craig, R.T. & Tracy, K.) pp 47-66, Sage.

Jones, C.B. 1980
'Software development - a rigorous approach'.
Englewood Cliffs, Prentice Hall.

Kelly, G.A. 1955
'The psychology of personal constructs'
New York, Norton.

Kirkby, M.J., & Morgan, R.P.C. (Eds.) 1981
'Soil erosion'.
Wiley.

Kowalski, R.A. 1980
'Contribution to SIGART newsletter No. 70'.
Special issue on knowledge representation.

Kowalski, R.A. 1983
‘Logic for expert systems’.
In, Expert systems 83, Proceedings of BCS SGES conference 1983,

Kowalski, R.A. 1985
‘The relation between logic programming and logic’.
In, Mathematical logic and programming languages,

Kraft, A. 1984
‘XCON: An expert configuration system at Digital Equipment Corporation’.
In, The AI business, commercial uses of artificial intelligence.
(Eds., Winston, P.H and Prendergast, K.A.) pp41-51, MIT Press Cambridge,
Mass.

Lederberg, J. 1964
‘Computation of molecular formulas for mass spectrometry’.
San Francisco: Holden-Day.

Leibrandt, U., & Schnupp, P. 1983
‘An evaluation of prolog as a prototyping system’.
In, Approaches to prototyping, (Eds., Buddle, R., Kuhlenkamp, K.,

Leplat, J. & Bissaret, A. 1965
‘Analyse des processus de traitement de l’information chez le controleur
de la navigation aerienne’.

‘A procedural semantics for semantic networks’.
In, Associative networks, representation and use of knowledge by

‘Foundations of a functional approach to knowledge representation’.
Artificial Intelligence 23 pp 155-212.

‘The efficiency of algorithms’.

Lindsay, R., Buchanan, B.G., Feigenbaum, E.A., & Lederberg, J. 1980
‘DENDRAL’.

‘Set theory and related topics’.
Schaum’s Outline Series In Mathematics.
McGraw-Hill.

Martelli, A. 1977.
‘On the complexity of admissible search algorithms’.
McCarthy, J. 1962
'Towards a mathematical science of computation'.
Information processing, Proceedings of IFIP congress.

'LISP 1.5 Programmers manual'.

McDermott, J. 1982
'XSEL: A computer sales person's assistant'.

McDougall, R. 1904
'Recognition and recall'.
Jour. of Philosophical Psychology and Scientific Methods, 1. pp 229-233.

Michalski, R.S. & Chilauski, R.L. 1980
'Learning by being told and learning from examples: An experimental
comparison of the two methods of knowledge acquisition in the context
of developing an expert system for soybean diagnosis'.

Michalski, R., Carbonell, J.G., & Mitchell, T.M 1983
'Machine Learning'.
Tioga, Palo Alto, Calif.

Michie, D. 1982
'The state of the art in machine learning'.
In, Introductory readings in expert systems, (Ed., D. Michie)
Gordon and Breach, UK.

Morgan, C. & Sufrin, B.A. 1985
'Specification of the UNIX filing system'.

Morgan, C. & Sorensen, I. 1984b
'The schema language'.
Programming research Group (internal document)
Computing Laboratory, Oxford University.

'KEATS: The knowledge engineer's assistant'
Alvey project IKBS/020. Final project report.

Newell, A. & Simon, H.A. 1961
'GPS, a program that simulates human thought'.
Lernende Automaten, Munich: R.Oldenbourg KG.

Newell, A. & Simon, H.A. 1972
'Human problem solving'.
Englewood Cliffs, Prentice Hall.

'The knowledge level'.
Artificial Intelligence 18, pp 87-127.
Nil, H.P. & Aiello, N. 1979
'AGE (Attempt to generalise): A knowledge based program for building
knowledge based programs'
In, IJCAI 6, pp 645-655

Nilson, N.J. 1982
'Principles of artificial intelligence'
Springer Verlag.

Paxton, W.H. 1976
'A framework for language understanding'.
SRI technical report note 131.
AI centre, SRI International Inc, Menlo Park, Calif.

Plant, R.T. 1985
'An investigation of knowledge based help facilities'.
Masters dissertation
Programming Research Group, Oxford University.

Pople, H. 1977
'The formation of composite hypotheses in diagnostic problem solving - An
exercise in synthetic reasoning'.
IJCAI 5, 1030-1037.

Post, E. 1943
'Formal reductions of the general combinatorial problem'.

Quillian, J.R. 1968
'Semantic memory'.
In, Semantic information processing (Ed. Minsky, M.)

Quillian, J.R. 1979
'Discovering rules from large collections of examples: A case study'.
Edinburgh University Press.

Quine, W.V.O. 1960
'Word and object'
MIT Press, Cambridge M.A.

Reddy, R., Erman, L., Fennell, R. & Neely, R. 1976
'The hearsay speech understanding system: An example of
the recognition process'.

Rich, E. 1983
'Artificial intelligence'.
McGraw Hill.

Reingold, E.M., Nievergelt, J. & Dao, N. 1977
'Combinatorial algorithms: Theory and practice'.
Englewood Cliffs, Prentice Hall.

Russell, B., & Whitehead, A.N. 1910
'Principia Mathematica'
Cambridge University Press.
Shortliffe, E.H. 1976
'Computer-based medical consultations: MYCIN'.
New York: North Holland

Sloman, A. 1984
'Why we need many knowledge representation formalisms'.
In, Research and development in expert systems. (Ed. Bramer, M.A.)

Smith, D.H., Buchanan, B.G., Englemore, R.S., Duffield, A.M., Yeo. A.,
Feigenbaum, B.A.,Lederberg, J., & Djerassi, C. 1972
'Applications for artificial intelligence for chemical inference VIII. An
approach to the computer interpretation of the high resolution mass
spectra of complex modules. Structure elucidation of estrogenic steroids'.
Journal of the American Chemical Society 94:5962.

Smith, D.H 1975
'Applications of artificial intelligence for chemical inference XV.
Constructive graph labelling applied to chemical problems. Chlorinated
hydrocarbons.'
Analytical chemistry 47:1176.

Spivey, M. 1984
'Towards a formal semantics for the Z notation'.
Technical monograph PRG-41
Computing Laboratory, Oxford University.

Stefik, M. 1980
'Planning with constraints'.
Report No. 784, Computer Science Dept. Stanford University.

Stefik, M. 1981a
'Planning with constraints (MOLGEN: Part 1).'
AI 16: 111-139.

Stefic, M. 1981b
'Planning and meta-planning (MOLGEN: Part 2).'
AI 16: 141-169.

Stokes, R., & Hewitt, J.P. 1976
'Aligning actions'.

Stoy, J.E. 1977
'Denotational semantics: The Scott-Strachey approach to programming language
theory'.

Sufrin, B.A. 1984
'Notes for a Z handbook: Part 1 - The Mathematical language'.
Programmin Research Group (Internal document), Oxford University.

Sully, F.K., & Unstead, P.J 1983
'Automobile engines'.
Newnes Technical, London.

Thornes, J.B. 1976
'Semi-arid erosional systems, case studies from Spain'.
London School of Economics and Political Science, London.

Thornes, J.B. & Brunsden, D. 1977
'Geomorphology and time'.

Thornes, J.B. 1985
'River channels'.
MacMillan Educational.

Thornes, J.B. 1987
'Erosional equilibria under grazing'
To appear in, Environmental Archaeology.
(Eds. Biutiff, J. & Shackley, M.)

Turner, D.A. 1982
'Recursion equations as a programming language'
In, Functional programming and its applications
(Eds., Darlington, J., Henderson, P. & Turner, D.)
pp 1-28, Cambridge University Press.

Turner, D.A. 1985a
'Miranda: A non-strict functional language with polymorphic types'.
In Proceedings of the IFIP Conference on Functional Programming Languages

Turner, D.A. 1985b
'Functional programs as executable specifications'.
In, Mathematical logic and programming languages,

Yu, V.L., Buchanan, B.B., Shortliffe, E.H., Wraith, S.M., Davis, R.,
'Evaluating the performance of a computer based consultant'.
Computer programs in Biomedicine 9:95-102

Blum, R.I., Buchanan, B.G., Cohen, S.N. 1979b
'Antimicrobial selection by a computer - A blinded evaluation by infectious
disease experts'.

van Melle, W. 1979
'A domain-independent production-rule system for the consultation programs'.
IJCAI 6, pp 923-925.

Weiss, S.M. & Kulikowski, C.A. 1979
'EXPERT: A system for developing consultation models'.

Weiss, S.M. & Kulikowski, C.A. 1983
'A practical guide to designing expert systems'.
Chapman Hall, London.

Welbank, M. 1983
'A review of knowledge acquisition techniques for expert systems'.
British Telecom Research Labs. Martlesham Consultancy Services.
Wielinga, J.B., & Breuker, J.A. 1983
'Analysis techniques for knowledge based systems: part 1'.
Report 1.1 Esprit Project 12.

Winston, P.H. & Horn, B. 1981
'LISP'
Addison Wesley, Reading, Mass.

Wirth, N. 1976
'Algorithms + Data structures = Programs'
Englewood Cliffs, Prentice Hall.
THE "Z" SCHEMA NOTATION.

1. Schema Definition

A schema groups together some declarations of variables and a predicate relating these variables. There are two ways of writing schemas: vertically, for example:

```
S
|  |
| x : N|
| y : seq N|
|     |
| x < #y|
```

or horizontally, for the same example:

```
S = {x : N; y : seq N | x < #y}
```

2. Inclusion

A schema S may be included within the declarations of a schema T, in which case the declarations of S are merged with the other declarations of T (variables declared in both S and T must be the same type) and the predicates of S and T are conjoined. e.g.

```
T
|  |
| S
| z : N|
|     |
| z < x|
```

is

```
|  |
|x, z : N|
| y : seq N|
|     |
| z < x|
|x < #y|
```
3. Renaming of components:

\[ S[\text{new/old}] \] the schema \( S \) with the component \text{old} renamed to \text{new} in its declaration and every use of that \text{old} within the predicate.

\[
\begin{align*}
\text{e.g. } S[z/x] & \text{ is } [z : N; \ y : \text{seq } N \ | \ z \leq \#y] \\
\text{and } \\
S[y/x, x/y] & \text{ is } [x : N; \ y : \text{seq } N \ | \ y \leq \#x]
\end{align*}
\]

4. Declaration.

Declaration with subscript, superscript, prime, etc.; systematic renaming of the variables declared in the schema.

\[
\begin{align*}
\text{e.g. } S' & \text{ is } [x' : N; \ y' : \text{seq } N \ | \ x' < \#y']
\end{align*}
\]

5. Negation.

\( \neg S \) The Schema \( S \) with its predicate part negated. e.g., \( \neg S \) is

\[
[x : N; \ y : \text{seq } N \ | \ \neg x < \#y]
\]


The schema formed from schemas \( S \) and \( T \) by merging their declarations and and’ing their predicates. Given:

\[
T = [x : N; \ z : \text{PN } \ | \ x \in z]
\]

\( S \land T \) is
7. Disjunction.

The schema formed from schemata S and T by merging their declarations and or'ing their predicates. Given:

\[ T \text{ equin } [x : N; z : PN \mid x \in z] \]

\[ S \lor T \text{ is } \]

\[[x : N\mid y : \text{seq } N\mid z : P \text{N}\mid x < \#y \land x \in z] \]

8. Implication.

The schema formed by schemas S and T by merging their declarations and taking pred S \( \Rightarrow \) pred T as the predicate. e.g., S \( \Rightarrow \) T is similar to S \( \land \) T and S \( \lor \) T except the predicate contains an "\( \Rightarrow \)" rather than an "\( \land \)" or an "\( \lor \)".


The schema formed by schemas S and T by merging their declarations and taking pred S \( \Leftrightarrow \) pred T as the predicate. e.g. S \( \Leftrightarrow \) T the same as S \( \land \) T with "\( \Leftrightarrow \)" in the place of the "\( \land \)".


S \( (v_1, v_2, ..., v_n) \) is the schema S with the variables \( v_1, v_2, ..., v_n \) hidden; the variables listed are removed from the declarations and are existentially qualified in the predicate.
e.g. \( \{ y : \text{seqN} \mid \exists x : \text{N} \cdot x < #y\} \)

A schema may be specified instead of variables in this case the variables declared in that schema are hidden, e.g., \((s \lor 't)\) \(S\) is

\[
\begin{array}{l}
| z: \text{PN} \\
| \\
| (\exists x : \text{N}; \ y : \text{seqN}, x < #y \land x \in z) \\
\end{array}
\]

11. Conventions.

The following conventions are used for variable names in those schemas which represent operations:

- undashed state before the operation,
- dashed state after the operation,
- ending in "?" inputs to the operation, and
- ending in "!" outputs from the operation.

The following schema operations only apply to schemas following the above conventions.

12. Precondition.

\(\text{pre } S\)

All the states after components (dashed) and the outputs (ending in "!" are hidden e.g., given

\[
\begin{array}{l}
\text{S} \\
| x?, s, s', y! : \text{N} \\
| \\
| s' = s - x? \\
| y! = s \\
\end{array}
\]

\(\text{pre } S\) is
13. Postcondition.

This is similar to precondition except all the state before components (undashed) and inputs (ending in '?' ) are hidden.


\[ S O^+ T = (S \land \neg \text{pre } T) \lor T \]

e.g. given \( S \) above and

\[
\begin{align*}
S O^+ T \text{ is} \\
\text{\( x?, s, s': N \) } \\
\text{\( s < x? \) } \\
\text{\( s' = s \)}
\end{align*}
\]

The predicate can be simplified:

If we consider an intermediate state that is both the final state of the operation S and the initial state of the operation T then the composition of S and T is the operation which relates the initial state of S to the final state of T through the intermediate state. To form the composition of S and T we take the state after components of T that have a basename * in common, rename both to new variables, take the schema "and" of the resulting schemas, and hide the new variables.

e.g. $s \circ T$

*) basename is the name with any declaration ("","|","?", etc.) removed.
Appendix A.2
THE MATHEMATICAL NOTATION OF "Z".

1. DEFINITIONS AND DECLARATIONS.

Let \( x, x_k \) be identifiers and \( T, T_k \) sets.

- LHS \( \overset{^\wedge}{=} \) RHS Definition of LHS as syntactically equivalent to RHS.
- \( x: T \) Declaration of \( x \) as type \( T \).
- \( x_1: T_1; x_2: T_2; \ldots; x_n: T_n \) List of declarations.

2. LOGIC

Let \( P, Q \) be predicates and \( D \) declarations.

- \( \neg P \) Negation: "not P".
- \( P \land Q \) Conjunction: "P and Q".
- \( P \lor Q \) Disjunction: "P or Q".
- \( P \Rightarrow Q \) Implication: "P implies Q" or "if P then Q".
- \( \forall x: T \bullet P \) Universal quantification: "for all \( x \) of type \( T \), \( P \) holds".
- \( \exists x: T \bullet P \) Existential quantification: "there exists an \( x \) of type \( T \) such that \( P \)".
- \( \exists! x: T \bullet P \) Unique existention "there exists a unique \( x \) of type \( T \) such that \( P \".

\[ \forall x: T \bullet P \overset{\wedge}{=} (\exists y: T \bullet (y \neq x \land P_y)) \]

\[ \forall x_1: T_1; x_2: T_2; \ldots; x_n: T_n \bullet P \] "For all \( x_1 \) of type \( T_1 \), \( x_sub 2 \) of type \( T_2 \), ..., and \( x_n \) of type \( T_n \), \( P \) holds.

\[ \exists x_1: T_1; x_2: T_2; \ldots; x_n: T_n \bullet P \] Similar to \( \forall \).

\[ \forall D \mid P \overset{\wedge}{=} (\forall D \bullet P \Rightarrow Q) \]

\[ \exists D \mid P \overset{\wedge}{=} (\exists D \cdot P \land Q) \]
t₁ = t₂  Equality between terms.

\[ t₁ \rightarrow = t₂ = \neg (t₁ = t₂). \]

3. SETS.

\[ t \in S \quad \text{Set membership: "t is an element of S"} \]

\[ t \not\in S \quad = \neg (t \in S). \]

\[ S \subseteq T \quad \text{Set inclusion: } = (\forall x:S \cdot x \in T) \]

\[ S \subset T \quad \text{Strict set inclusion: } = S \subseteq T \land s \not\subseteq T \]

\[ \{\} \quad \text{The empty set.} \]

\[ \{x:T \mid P\} \quad \text{The set containing exactly those } x \text{ of type } T \text{ for which } P \text{ holds.} \]

\[ (t₁, t₂, ..., tₙ) \quad \text{Ordered } n\text{-tuple of } t₁, t₂, ... \text{ and } tₙ. \]

\[ T₁ \times T₂ \times ... \times Tₙ \quad \text{Cartesian product: the set of all } n\text{-tuples such that the} \]
\[ \quad \text{kth component is of type } T_k. \]

\[ \{x₁:T₁; x₂:T₂; \ldots ; xₙ:Tₙ \mid P\} \quad \text{The set of } n\text{-tuples } (x₁, x₂, ... , xₙ) \text{ with} \]
\[ \quad \text{each } x_k \text{ of type } T_k \text{ such that } P \text{ holds.} \]

\[ \{D \mid P \cdot t\} \quad \text{The set of } t's \text{ such that given the declarations } D, \text{ P holds.} \]

\[ \{D \cdot t\} \quad = \{D \mid \text{true} \cdot t\}. \]

\[ P S \quad \text{Powerset: the set of all subsets of } S. \]

\[ F S \quad \text{Set of finite subsets of } S: = \{T: P S \mid T \text{ is finite}\}. \]

\[ S \cap T \quad \text{Set intersection: given } S, T: P X, = \{x:X \mid x \in S \land x \in T\} \]

\[ S \cup T \quad \text{Set union: given } S, T: P X, = \{x:X \mid x \in S \lor x \in T\} \]

\[ S - T \quad \text{Set difference: given } S, T: P X, = \{x:X \mid x \in S \land x \not\in T\} \]
Distributed set intersection: given SS:
\[ P(P X), \land \{x : X | (\forall S : SS \cdot x \in s)\} \]

Distributed set union: given SS:
\[ P(P X), \lor \{x : X | (\exists S : SS \cdot x \in s)\} \]

 '\#S' Size (number of distinct elements) of a finite set.

4. NUMBERS.

\[ N \] The set of natural numbers (non negative integers).

\[ N^+ \] The set of strictly positive natural numbers: \( = N - \{0\} \)

\[ Z \] The set of integers (positive, zero, and negative).

\[ m..n \] The set of integers between \( m \) and \( n \) inclusive:
\[ = \{k : Z | m \leq k \land k \leq n\} \]

\[ \min S \] Minimum of a set, \( S : \mathbb{F} N \).
\[ = \{x : S \land (\forall x : S \cdot x \geq \min S)\} \]

\[ \max S \] Maximum of a set, \( S : \mathbb{F} N \).
\[ = \{x : S \land (\forall x : S \cdot x \leq \max S)\} \]

5. Relations.

A relation is modelled by a set of ordered pairs hence operators defined for sets can be used on relations.

Let \( X, Y \) and \( Z \) be sets; \( x : X; y : Y \); and \( R : X \leftrightarrow Y \).

\[ X \leftrightarrow Y \] The set of relations from \( X \) to \( Y \): \( = P(X \times Y) \).

\[ x \mathbin{R} y \] \( x \) is related by \( R \) to \( y \): \( = (x, y) \in R \)

\[ x \mathbin{l-R} y \] \( (x, y) \)
\[
\{x_1 \mapsto y_1, x_2 \mapsto y_2, \ldots, x_n \mapsto y_n\}
\]

The relation: \(\{(x_1, y_1), \ldots, (x_n, y_n)\}\) relating \(x_1\) to \(y_1\), \ldots, and \(x_n\) to \(y_n\).

\textbf{dom} R \quad \text{The domain of a relation:} \quad ^\Delta \{x: X \mid \exists y: Y. x \mathrel{R} y\}.

\textbf{rng} R \quad \text{The range of a relation:} \quad ^\Delta \{y: Y \mid \exists x: X. x \mathrel{R} y\}.

\(R_1 \circ R_2\) \quad \text{Forward relational composition: given } R_1: X \leftrightarrow Y; R_2: Y \leftrightarrow Z,
\[(^\Delta = x: X; z: Z \mid \exists y: Y. x \mathrel{R_1} y \land y \mathrel{R_2} z)\]

\(R_1 \circ R_2\) \quad \text{Relational composition:} \quad ^\Delta = R_2 \circ R_1

\(R^{-1}\) \quad \text{Inverse of relation } R: \quad ^\Delta \{y: Y; x: X \mid x \mathrel{R} y\}

\textbf{id} X \quad \text{Identity function on the set } X: \quad ^\Delta \{x: X \mapsto x\}

\(R^k\) \quad \text{The relation } R \text{ composed with itself } k \text{ times: Given } R: X \leftrightarrow X,
\[R^0 = \text{id} X, R^{k+1} = R^k \circ R\]

\(R^*\) \quad \text{Reflexive transitive closure:} \quad ^\Delta \cap \{n: N \times R^n\}.

\(R^+\) \quad \text{Non reflexive transitive closure:} \quad ^\Delta \cap \{n: N^+ \times R^n\}.

\(R \cap (S \cup I)\) \quad \text{Image: given } S; P X, = \{y: Y \mid \exists x: S. x \mathrel{R} y\}.

\textbf{s} \mid \text{Domain restriction to } S:\n\quad \text{given } S; P X, = \{x: X, y: Y \mid x \in S \land x \mathrel{R} y\}

\textbf{R} \cap \text{Domain subtraction: given } S; P X, = \{x: X \mid x \mathrel{R} y \land y \in Y\}

\textbf{s} \mid \text{Range restriction to } T:\n\quad \text{given } T; P Y, = \{x: X; y: Y \mid x \mathrel{R} y \land y \in Y\}

-236-
R1 $O^+ R2$  Overriding: given $R1, R2: X \leftrightarrow Y$

6. FUNCTIONS.

A function is a relation with the property that for each element in its domain there is a unique element in its range related to it. As functions are relations all the operators defined above for relations also apply to the functions.

$X \downarrow \rightarrow Y$  The set of partial functions from $X$ to $Y$: $\uparrow \downarrow (\forall x: \text{dom } f \cdot (\exists y: Y \cdot x f y))$

$X \rightarrow Y$  The set of total functions from $X$ to $Y$: $\uparrow \downarrow (f: X \downarrow \rightarrow \text{dom } f = X)$

$X \downarrow\downarrow Y$  The set of one-to-one partial functions from $X$ to $Y$: $\uparrow \downarrow (\forall y: \text{rng } f \cdot (\exists x:X \cdot x f y))$

$X \leftrightarrow\leftrightarrow Y$  The set of one-to-one total functions from $X$ to $Y$: $\uparrow \downarrow (f: X \downarrow\downarrow Y \cdot \text{dom } f = X)$

$f t$  The function $f$ applied to $t$.

7. SEQUENCES.

$\text{seq } X$  The set of sequences whose elements are drawn from $X$: $\uparrow \downarrow (A: N^+ \downarrow \rightarrow X \cdot \text{dom } A = 1\ldots\#A)$

$\#S$  The length of sequence $A$

$[]$  The empty sequence.

$[a_1, \ldots, a_n] = \{1 \downarrow a_1, \ldots, n \downarrow a_n\}$

$[a_1, \ldots, a_n] \ast [b_1, \ldots, b_n]$  Concatenation: $\uparrow \downarrow [a_1, \ldots, a_n, b_1, \ldots, b_n]$
head $A^\wedge = A(1)$

last $A^\wedge = A \ (#A)$

tail $[X]^\wedge A = A$

front $A^\wedge [X] = A$

rev $[a_1, ..., a_n]^\wedge$ Reverse: $= [a_n, ..., a_1]$
Appendix B.1
A FRAME BASED REPRESENTATION LANGUAGE.

```
<sec-name>-----------------------------
 |  <desc-spec>
 |  <sec-body>
```

```
|  <para-type>
|  <body-type>
```

```
|  <text-name>
|  <text>
```

```
<sec-spec>::=<sec-name> <desc-spec> <sec-body>
```

```
<sec-name>::=<simple-string>
```

```
<sec-body>::=<pred> => ref <sec-spec> | <pred> => <text-gen>
```

```
<pred>::= ("<para-spec>")? | ("<para-spec> <operator> <action>")? | {
  ("<para-spec>")? <operator> <pred>
}*
```

```
<para-spec>::=<para-type> <body-type>
```

```
<action>::=<pred>-const
```

```
<const>::=<simple-string>
```

```
<para-type>::=<para-name> <type> | <xor> <xor-para>
```

```
<xor-para>::=<para-name> | <para-name> <xor-para>
```

```
<text-gen>::=<text-name> <text>
```

```
<para-name>::=<simple-string>
```

```
<text-name>::=<simple-string> <text-no>
```

```
<type>::=[<type-body>] | <type-double-body>
```

```
<type-body>::= fact | rule | category | number | phrase
```

```
<type-double-body>::=[ category] [rule] [fact] [rule]
```

```
<body-type>::=<fact-body> | <rule-body> | <category>
  <number-body> | <phrase-body> | <category-rule-body> | <fact-rule-body>
```

-340-
<fact-body>::= <desc-spec> <expl-spec> <query-spec>

<rule-body>::= <desc-spec> <expl-spec> <rule-spec> <query-spec>

<category-body>::= <desc-spec> <expl-spec> <options-spec> <query-spec>

<number-body>::= <desc-spec> <expl-spec> <number-spec> <query-spec>

<phrase-body>::= <desc-spec> <expl-spec> <query-spec>

<category-rule-body>::= <desc-spec> <expl-spec> <options-spec>

<rule-spec>::= rule: <rule> {, <rule>}* I rule: <rule> {, <rule>}*

<query-spec>::= query: <text>

<op>::= <relation> I <boolean-op>

<operator>::= <relation> I <boolean-op>
<relation>::= =|<|>|<=|>=
<expression>::= <el> ( <addop> <el>)*
<addop>::= +|-1
<multop>::= */
<boolean-op>::= /1/1 not
<el>::= ( <expression> ) | <valuespec>
/valuespec>::= <name> <number>
<text-no>::= <digit> | text-no <digit>
<number>::= <unsigned-integer> | <sign> <unsigned-number>
<sign>::= +|-1

/unsigned-number>::= <digit-sequence> | <decimal-number> | <exp-number>
<char-sequence>::= <letter> | char-sequence <letter>
<digit-sequence>::= <digit> | digit-sequence <digit>
<decimalal-number>::= <digit-sequence> * <digit-sequence>
<exp-number>::= <digit-sequence> <mantissa-number> | decimal-number <mantissa-number>
<mantissa-number>::= E <sign> <digit-sequence>
Appendix B.2
A REPRESENTATION SPECIFICATION FOR RULES

The following representation specification is based upon the standard one. However, the facility to represent facts has been added.

RULE: <rule-id>: IF <expression> ( AND <expression> ) THEN <action>

HEURISTIC: <heuristic-id>: IF <expression> ( AND <expression> ) THEN <action>

EXAMPLE: <example-id>: IF <expression> ( AND <expression> ) THEN <action>

FACT: <fact-id>: <expression> ( AND <expression> )

RULE: ::= <simple-string>

HEURISTIC: ::= <simple-string>

FACT: ::= <expression>

<rule-id>::= <simple-string>

<heuristic-id>::= <simple-string>

<simple-string>::= <name>kchar-seq>

<name>::= <lowercase>l<letter>knamexdigit

<letter>::= <!lowercase>IA..Z

<lowercase>::= a..z

<digit>::= 0..9

<char-seq>::= <letter>kchar-seqkletter

<expression>::= <simple-expressionkrelational-operatorksimple-expression>

<simple-expression>::= <term>

<term>::= <identifier>

<identifier>::= <letter>l<letter>kdigit>

<relational-operator>::= <> | = | < | > | ≤ | ≥ | boolean

<boolean>::= "true" | "false"

&action>::= <expression>
Appendix B.3
A SYNTAX FOR FLOW DIAGRAMS

<program>::= <program heading> <block>

<program heading>::= program <identifier>;

<identifier>::= <letter> { <letter or digit> }

<letter or digit>::= <letter> | <digit>

<letter>::= <lowercase> | A..Z

<lowercase>::= a..z | " "

<digit>::= 0..9

<block>::= <label declaration part> <procedure declaration part> <statement part>

<label declaration part>::= <empty> | label <label> { , <label> }

<label>::= <unsigned integer> | <letter> { <letter or digit> }

<unsigned integer>::= <digit> { <digit> }

<procedure declaration part>::= { <procedure declaration>; }

<procedure declaration>::= procedure <procedure identifier>

<text>::= <string> { <string> }

<string>::= <simple string> { <simple string> }

<simple string>::= <char sequence>

<char sequence>::= <letter> | <char sequence> <letter>

<statement part>::= <compound statement>

<compound statement>::= begin <statement> { ; <statement> } end

<statement>::= <unlabelled statement> | <label> : <unlabelled statement>

<unlabelled statement>::= <simple statement> | <structured statement>

<simple statement>::= <procedure statement> | <goto statement> | <empty statement> | <undefined statement>

<procedure statement>::= <procedure identifier>

<procedure identifier>::= <action identifier> | <test identifier> | <diagnosis identifier>

'action identifier>::= action<digit> { <digit> }

<test identifier>::= test<digit> { <digit> }

<diagnosis identifier>::= diagnosis<digit> { <digit> }

<goto statement>::= goto <label>
<empty statement>::= <empty>
<empty>::=
<structured statement>::= <compound statement> | <conditional statement>
<compound statement>::= begin <statement> { , <statement> } end
<conditional statement>::= <if statement>
<if statement>::= if <expression> then <statement> | 
                      if <expression> then <statement> else <statement>
<undefined statement>::= undefined | incomplete
<expression>::= condition<digit> { <digit> }

-347-
Appendix B.4
PSEUDO CODE FOR A FLOW DIAGRAM.

Pseudo code for the motor car domain case study which adheres to the syntax of the flow diagram language.

program starter; start:
begin
if condition1
then if condition2
then begin
  diagnosis1
  action1
  test1
  if condition3
  then begin
    action2
    goto start
  end
else begin
  test2
  if condition4
  then undefined
  else begin
    action3
    goto start
  end
end
else if condition5
then if condition6
then goto finish
else begin
  diagnosis2
  if condition7
  then begin
    action4
    incomplete
  end
  else begin
    diagnosis3
  end
end
else begin
  test3
  if condition8
  then begin
    diagnosis4
    test4
test5
  if condition9
  then begin
    diagnosis5
    action5
test6
  if condition10
  then begin
    action6
    goto start
  end
end
else undefined
    end
else begin
    action7
    goto start
    end
end
else begin
    diagnosis6
    action8
    test7
    if condition11
        then begin
            diagnosis7
            action9
            goto start
        end
    else begin
        diagnosis8
        action10
        if condition12
            then begin
                action11
                goto start
            end
        else begin
            undefined
            goto start
        end
    end
end
end
finish
end
-350-
Appendix B.5
title 'Car starter motor problem'.
'The aim of this system is to help people with little mechanical knowledge through the problems of starting a car'.

(problem) reference sl_starter_turns.

quit.

problem: 'try to find out if this is a new problem'
  fact
  explanation
    'this system can examine problems regarding starting a car'
  askable
    'is there a problem'.

/* */
/* SECTION 1: STARTER TURNS */
/* */

section sl_starter_turns:
'This section looks at the symptom of the starter turning'.

{sl_turns} reference s2_grinding_noise.

{not sl_turns} reference s3_starter_problems.

sl_turns:
'if the starter motor turns then isolate the problem area'
  fact
    explanation
      'This test isolates the problem area to the starter motor or its electrics'
  askable
    'does the starter motor turn over'.

/* */
/* SECTION 2: GRINDING NOISE */
/* */

section s2_grinding_noise:
'This section deals with the area of grinding noises from 's within the engine'.

{s2_full_listing or not s2_full_listing}
'information on grinding noises should be given here'.

-352-
s2_full_listing:
  'information on the problem area'
  fact
  explanation
  'gives fuller information on the problem area'
  askable
  'can you hear grinding noises?'.

/*-----------------------------------------------*/
/*  SECTION 3: STARTER PROBLEMS */
/*-----------------------------------------------*/

section s3_starter_problems:
  'This section deals with starter problems'.
  [s3_test1] reference s4_electrical_prob.
  [not s3_test1] reference s5_starter_motor_prob.

s3_test1:
  'we need to test whether the lights dim'
  fact
  explanation
  'this isolates the problem as an electrical one'
  askable
  'do the headlights dim?'.

/*-----------------------------------------------*/
/*  SECTION 4: ELECTRICAL PROBLEMS */
/*-----------------------------------------------*/

section s4_electrical_prob:
  'This section deals with electrical problems'.
  {s4_test2}
    'The battery is ok'.
  {not s4_test2}
    'replace the battery'.
  {s4_test3}
    'wiring is ok'.
  {not s4_test3}
    'replace the wiring'.

s4_test2:
  'we need to test the battery'
  fact
  explanation
  'this tries to isolate the problem to the battery area'
  askable
  'is the battery ok?'.

s4_test3:
  'we need to test the wiring'

-353-
fact
explanation
'this tries to isolate the problem to wiring'
askable
'is the wiring ok?'.

/* */
/* SECTION 5: STARTER MOTOR PROBLEMS */
/* */

section s5_starter_motor_prob:
'This section tests the area of the starter motor problems'.

{s5_solenoid and s5_crank}
'could be the wiring - check the wiring up to the ignition switch &
'and the seat belt interlock relay'.

{s5_solenoid and not s5_crank} reference s6_starter_diag.

s5_solenoid:
'we need to test the wiring'

fact
explanation
'this isolates the problem to a wiring circuit'
askable
'are the solenoid terminals shorted together?'.

s5_crank:
'we need to isolate the problem'

fact
explanation
'isolate the problem in the crank or wiring systems'
askable
'does the crank start normally?'.

/* */
/* SECTION 6: STARTER DIAGNOSIS */
/* */

section s6_starter_diag:
'may be the starter motor'.

{s6_starter_motor}
'replace the starter motor'.

s6_starter_motor:
'we need to isolate the problem'

fact
explanation
'try to decide the problem in the starter motor area'
askable
'is the starter motor itself ok?'.

-354-
Appendix C.1
The elicited representation.

JB: What methods exist for preventing soil erosion?

AM: Right, well if you consider that the critical factors are: the rainfall characteristics, the soil characteristics and the topography, you have got to consider the degree of slope, and if the slope is not very great, then you can probably control erosion by altering the soil characteristics in some way, by say mulching, or putting down one of these cause meshes, something like this, for getting up the organic matter level, something like this, so you are encouraging a greater cohesion of the soil and also by increasing the roughness of the surface with this machine which puts holes in there you are increasing the surface retention, if you like, of water. So on low slope you have probably not got that many problems. Unless, for instance you have got flood irrigation where the land has been put into these terrace so initially there would have been a slope but say you are inheriting this agricultural land which is a terrace formation and then, well, for a start you have got to upkeep these terraces because one of the problems in Spain particularly now is that you have got this movement of the farmers into the cities and the terraces are going into disuse and those front walls are not been upkept and as soon as those start falling down then you have got this horrific problem of the pore pressures being increased then the water suddenly just gushes; bursts, through the wall. So if you have inherited that sort of thing then you have got to keep the upkeep going - you have to put in the man hours basically, if you are inheriting that, that's the problem with that system. It is quite intensive with man hours.

JB: These 2 techniques you mentioned, one, mulching and the other actually putting down some thing that actually physically cover the surface. Now would you describe these as functionally equivalent procedures or are there different circumstances under which you would rather use one than the other.

AM: I think certainly, that if you were going to put down this mesh as I was saying, that you probably would not do that for a lower slope unless you for say, had this flood irrigation problem, in which case you might start to introduce that. The key thing would be to keep your ground cover, so that, if you have got Olive trees to keep a ground cover around it as well, otherwise you are going to get this problem. Ah if you have got fallow, if you have got a natural vegetation then you probably have not got very many problems, what's happening at the moment is that there perceiving that there is a problem, and so on the steeper slopes if we go onto those, they perceive that there is a problem and then bulldozing all of the natural vegetation away and putting in terraces and then putting in forest, trees, but if the trees don't survive which is quite likely in semi arid environments, they die and you get these terraces which are bare and the whole system goes berserk because that is exactly how you get a concentration of water in that the water concentrates at the back of the terraces and in any drains and unless you have designed it so that it is also resistant to a freak storm, which you can get every 20, 30 or 40 years, then its just that freak storm which will tip it over the balance and once you have got these channels, these gullies or whatever, then you have lost your gamble if you like.

JB: So one key under those circumstances would be insuring that the vegetation does survive.

AM: Either that the natural vegetation survives or that you get a vegetation established in time, that you do not leave the system vulnerable.

JB: So something that you could put into the soil which would make it more permeable and assist vegetation would help you on both fronts.

AM: Certainly, yes.

JB: So we could add to our list some sort of fertiliser

AM: Yes
JB: Do you know what sort of characteristics that would have to have?

AM: I'm afraid that I don't know much on fertilisers, I must admit, what I would say is that a natural organic mulch is better than a fertiliser.

RP: Peat?

AM: I don't know that you would get peat in a semi arid environment, as such, animal...

MS: Organic manure?

AM: Yes, organic manure.

RP: Straw?

AM: Yes, straw is very good, because it does get that binding, and that is, you do see that, but obviously if you are talking about remotest hill slopes you are going to get into problems.

MS: Is there any possibility of overcoming any of the difficulties you have been describing, by minor civil engineering so to speak, in other words building more robust banks to the terraces or is that too expensive.

AM: Well, if the funding is there well yes, I mean, certainly there are places where they are growing vines and so forth, where obviously it is very cost effective to make it entirely into a man made environment and so they have got the terraces built up and they have got all the drainage worked out, and its all up kept, and so they are putting the man hours and machinery in, and provided that they keep it like that they are ok, but as soon as that um work was not in put then you would start to get lots of problems built up.

MS: And any thing like that is presumably substantially more expensive than planting schemes.

AM: Yes, it really does depend upon the land use that you are trying to target up, once you start to talk about growing the cash crops you probably can start to go onto this sort of thing, but I think that if you are talking about a system where there is not the availability of manpower, then there is certainly a move not to create terraces as such, but if you are using, if you are growing olive trees or whatever to use this sprinkler system, where you have got a pipe and then you have got this jet to the specific tree. I mean the problem, obviously, with irrigation is that you have also got to avoid salinization because, obviously again, when you get into that problem then you have to thinking about the chemical and not killing the plant off.

JB: Am I right in thinking that irrigation is itself a preventative measure for soil erosion. In the right context.

AM: In the right context, yes, provided you don't over irrigate so that you get salinization problems, provided, I mean this bucare flood irrigation scheme, in that you are putting across the land the very medium which will cause the soil erosion if you don't control it. With the terrace system you may very well get sheet erosion which brings all of the sediment down to the bottom of the terrace, and if in the next season you plough it up and spread it back across the slope then that's fine, so what you have to avoid is loosing that soil completely off the field so the other thing to do is to have some strip of denser vegetation cover which will actually catch the sediment or even a mesh, some sort of mesh which will capture that sediment so that you don't loose it.

RP: So, the angle on the terrace is not necessarily zero degrees? AM: The angle on the terrace is indeed important.

RP: The terraces are not necessarily horizontal then...

AM: Oh, no, there is flood irrigation they can't be horizontal because what they are trying to do is the whole terrace such that it tilts to the exit point, it will only be a couple of degrees. I mean otherwise if you have it above that you will start getting into the problem of rilling and then leading to gullying so the whole terrace is actually constructed so that the water will actually pass across it to the exit point.

-357-
RP: To distribute the water

AM: To distribute the water entirely across the field. Well that’s one scheme, another scheme is where there is a wall at the front and then the water ponds across the surface and then it does actually stay there for the few hours or whatever that it takes to drain or seep into the soil. And that’s where you can get the problem that you are building up the pore pressures in the soil and you can get this sudden almost explosion through the wall that is not strong enough to restrain that pressure.

RP: So you are trying to balance the moisture content all the time of the soil by either drainage or irrigation.

AM: Absolutely, if you have got irrigation it is important that the soil itself is well drained so you have got this dichotomy where you are pumping water in but you are making sure that it is well drained as well.

JB: Is flood irrigation a poor man’s solution to the problems as opposed to using sprinklers?

AM: Yes, I would say so yes.

MS: You mentioned the possibility of having a barrier of vegetation, a barrier, a growth, to prevent the soil being washed over the edge so to speak, this is something that has not had a lot of attention in the discussions we have had. Why is that do you think?

AM: I think that perhaps it should have the attention, umm, I don’t know about Spain in this instance, but I know that one of the effective ways that if you have got a terrace which has a longer slope, a longer deeper terrace, that one thing you can do is to have strips of different crops, bands of different crops, so that if you want to have wheat or a serial crop, you then intersperse that with denser vegetation a lower one which will have a greater matt, and of course within this you have also got to think very much about root Matt, and so that is another consideration. If say you are going to clear your vegetation, is that, are you going to disturb the root matt, because if say, you burnt the vegetation your not necessarily disturbing the matt, where as if you plough it up the you loose your root matt. That is one consideration you have to bear in mind. In umm, We are going on to tropical countries, I'm sorry, but when we were talking about Brazil yesterday, the cases we were talking about where the trees were burnt rather than the big heavy machinery which also rips out the roots as well that’s the case where you can avoid the situation of this massive erosion because you have still got the root matt, which is protecting, holding the soil together and if you can get that, then provided that you don’t have a rain intensity that overcomes that as well, then you are elevating the problem.

RP: So an ideal preventative measure is to have a low level vegetation of the soil with a heavy root mat.

AM: Well, you see what one thing that you could do, is if you plough up your soil then you had sowed is to actually have a mesh a fine mesh over the soil and the seedlings can grow through that, and then you would be holding the soil as well ok, until the seedlings themselves have got established.

RP: Any compaction?

AM: No, no

RP: Just to stop the rain splash?

AM: Yes so that it disperses that kinetic energy and that it holds the soil together.

MS: Is it possible to quantify, in any way, the width of vegetation that you need at the edge of the terrace in relation, for instance, to slope or any other factors in order to get a significant degree of soil retention.

AM: I’m sure that it would be, it would obviously depend upon the slope angle and the slope length so the amount of the water you are expecting to cross that terrace and also the sediment concentration that you were expecting, if its very high concentration the it is going to fall out very much quicker than if it is very fine, if its fine then you probably have not got a problem anyway, but um there are going to be those factors relating in the calculation, I can’t come up with it now but it is certainly something
that you could calculate.

JB: Would there be any conditions in which I would be right in saying, that, you have planted a crop to help retain the soil, and the roots are doing that, but you suddenly come across a condition in which the roots are breaking up the soil and actually increasing the amount of erosion?

AM: I would not have thought so, if you were talking about tree crops you would only get that in a very extreme case.

JB: OK

AM: That is irrelevant.

JB: So, just a recap then, you have got the following methods: mulching, putting down some cover, terracing, adding fertiliser, irrigation - two types of irrigation, flood, and sprinkler, we have discussed the problems of flood irrigation putting down barriers to control the heads of water, such as walls or vegetation. Is there anything else, that would help as a preventative measure?

AM: Well so far we have been dealing with cultivation, if you started talking about grazing. Animal husbandry, does that sound fair.

JB: Yes.

AM: What you have got to ensure is that the animals do not concentrate on one particular localised area, you have got to keep them moving, this is if you are on a vulnerable slope.

RP: So don't overgraze?

AM: So don't overgraze, you don't want them to lower the vegetation below that critical 30 percent threshold because if you do then you are getting into problems. Not only that but you are also talking about breaking up the soil surface, even if you are at 40% vegetation you have probably still got bare soil, what usually you find there is a surface crust, but if they break up that surface crust, or goats or whatever do actually have these kind of hollows or what ever, and if they start to create that sort of thing then you are going to get a flow convergence, so you have always got to get it so that they keep moving.

MS: Are there particular types of animals which would be better or worse from this point of view?

AM: Certainly, cattle are probably not a very good idea on steep slopes or vulnerable slopes.

MS: Is that because of the steepness or the way that they graze?

AM: Because of the way that they graze, and the nature of there... to be honest you don't get many cattle as far as I know but that is in Spain, I'm thinking on a rather limited level there, if you talk about Africa where of course you have got, but cattle anyway, you have got humm but cattle anyway you would not have on steep slopes, you would avoid the larger animal putting them on the vulnerable slopes.

RP: Larger animal more weight.

AM: Yes, I don't really know all the details here but it will depend on the type of trampling that they will put onto the soil, so if you have got, umm species of cattle that does have hoofs or what ever that are more...

RP: Is it possible to work out the capacity of the land for supporting a certain amount of cattle, is that possible.

AM: Yes, that would be possible certainly, you would calculate that from average slope, from level of vegetation, from vegetation type, so for each animal you were thinking of you would go through and see which was the most suitable.

JB: Can animals ever have a beneficial effect?
AM: Yes, in that their manure is going to be a fertiliser.

RP: Is the compaction caused by their presence or their hoofs cause more damage?

AM: Well you see, if you have got compaction you will be decreasing the infiltration, that's part of the problem, is that in fact where you have compacted soil you may actually get a concentration of the flow, this is one of the, I think with grazing animals on of the difficulties is that they can produce the conditions which may cause a concentration of flow and my bias is obviously in channelised flow, erosion and that may have come through by the way I'm talking and as far as I'm concerned that is the very condition that you try to avoid.

RP: So they can encourage rills to turn into gullies?

AM: Yes, yes., having said which of course they can also, if you have got a rill or a gully they destroy, they destroy the rill by there trampling effect, so in that sense, I would have thought that was an extreme case because rills anyway are ephemeral, because if you have got the situation where a rill will form, it will form again in the next storm event anyway and the trampling will cause compaction and decrease infiltration, having said which if there is an excess of trampling then their manure production will be of benefit, provided you have got the balance of vegetation growth your alright, its just as soon as you tip that balance.

MS: There is a value from the manure of animals on the land, there is also presumably a value from the fact that if you just leave the land completely fallow you get nothing from it, where as if you animals grazing you do get some benefits from that point of view, in other respects, the use of husbandry appears to be anti-beneficial, is that right?

AM: Only if, it is causing too greater stress on the environment.

MS: There presence inevitably causes stress

AM: Depends how resistance that environment is, if the resistance of the environment is such that it can take that level of stress from that given breed of animal then you are alright.

RP: There is a threshold level

AM: Considering your level of force, your level of resistance, basically if you take it on such a broad level, you know, don't think about all the various factors, but if can assess forces which may be put onto that environment/system, and the resistance that system can offer, and providing you have got either the resistance well you want that resistance to be greater than the force that is being put onto the system and once you start getting the two values get closer and closer obviously you are getting towards a greater risk and that's really the point, that, once you do then you have got greater risk from a freak storm or whatever.

JB: It seems to me at this point now that there is room for exploring many interactions for example, if you have got something covering the soil presumably you can not have anything grazing on that surface as well...

AM: What I have forgotten to say, is if you have a drought, I mean that is another possibility, as well, if you have a drought happen, I mean that I was talking about freak storms well that's being a little one side, if you have a drought as well then you are decreasing the vegetation cover so that's another factor that you have got to consider, is that through out the year the resistance offered by the environment may in fact change, so at the end of the summer when the vegetation is at its lowest its also at its most vulnerable, and its at the end of the summer when you may get these freak October storms in Spain, you know, In Spain, but also you will get the seasonal change in all semi arid environments where at the end of the summer you have got this very critical, you may well have conditions of the soil in which infiltration is not going to be very fast, although overland flow may in fact in turn may be very rapid.
RP: So, back to keeping the moisture content of the soil......

AM: Yes, you have got to be very careful that the end of the summer you have not created the situation where in that environment is not going to be very vulnerable to the first seasonal storms. What works for you in some ways is that during the summer you may in actual fact, depending upon the soil type, the soil may have in fact of broken up, there may be clods or whatever that in actual fact be picked up immediately and take up all of the transporting capacity of the overland flow but in many case what happens is when you get that first rain all the soil surface just goes into, a kind of, it just seals all of the cracks seal and everything and you just get a sheet going right across and if you have got that happening, and you have got a low vegetation cover because it has been dying off during the summer then that system is very vulnerable and you have got to make quite sure that what you have done through your animal husbandry, or your cultivation or whatever has not made it such that it will be put across the threshold to excessive soil loss. There is actually, you can calculate the amount of soil that you can afford to lose off a given field if you know the rate of soil production. I think, one that the Americans came up with when they were investigating the ULSE it was something like 25 tons per, although that does not sound quite right........

RP: Soil production as in......

AM: The actual creation, formation of soil itself, you can actually calculate the balance so that you never want the soil loss to be greater than the formation of the soil itself.

MS: What leads to soil creation?

AM: Ah,

MS: What encourages that?

AM: Ah, well, again ah the introduction of organic matter is critical.

MS: Is there any major preventative measure that we have not covered?

AM: Ummm, no ok...

End of transcript.

KB : Principal knowledge engineer (JB) SKE: Secondary knowledge engineer (RP,MS) DE : Domain expert (AM)
Appendix C.2
BLOCK ONE - ADJACENCY PAIRS 1 - 10.

KE1: What methods exist for preventing soil erosion?

DEI: Right, well if you consider that the critical factors are: the rainfall characteristics, the soil characteristics and the topography, you have got to consider the degree of slope, and if the slope is not very great, then you can probably control erosion by altering the soil characteristics in some way, by say mulching, or putting down one of these cause meshes, something like this, for getting up the organic matter level, something like this, so you are encouraging a greater cohesion of the soil and also by increasing the roughness of the surface with this machine which puts holes in there you are increasing the surface retention, if you like, of water. So on low slope you have probably not got that many problems. Unless, for instance you have got flood irrigation where the land has been put into these terrace so initially there would have been a slope but say you are inheriting this agricultural land which is a terrace formation and then, well, for a start you have got to upkeep these terraces because one of the problems in Spain particularly now is that you have got this movement of the farmers into the cities and the terraces are going into disuse and those front walls are not been upkept and as soon as those start falling down then you have got this horrific problem of the pore pressures being increased then the water suddenly just gushes; bursts, through the wall. So if you have inherited that sort of thing then you have got to keep the upkeep going - you have to put in the man hours basically, if you are inheriting that, that's the problem with that system. It is quite intensive with man hours.

---------

KE2: These 2 techniques you mentioned, one, mulching and the other actually putting down some thing that actually physically cover the surface. Now would you describe these as functionally equivalent procedures or are there different circumstances under which you would rather use one than the other.

DE2: I think certainly, that if you were going to put down this mesh as I was saying, that you probably would not do that for a lower slope unless you for say, had this flood irrigation problem, in which case you might start to introduce that. The key thing would be to keep your ground cover, so that, if you have got Olive trees to keep a ground cover around it as well, otherwise you are going to get this problem. Ah if you have got fallow, if you have got a natural vegetation then you probably have not got very many problems, what's happening at the moment is that there perceiving that there is a problem, and so on the steeper slopes if we go onto those, they perceive that there is a problem and then bulldozing all of the natural vegetation away and putting in terraces and then putting in forest, trees, but if the trees don't survive which is quite likely in semi arid environments, they die and you get these terraces which are bare and the whole system goes berserk because that is exactly how you get a concentration of water in that the water concentrates at the back of the terraces and in any drains and unless you have designed it so that it is also resistant to a freak storm, which you can get every 20, 30 or 40 years, then its just that freak storm which will tip it over the balance and once you have got these channels, these gullies or whatever, then you have lost your gamble if you like.

---------

KE3: So one key under those circumstances would be insuring that the vegetation does survive.
DE3: Either that the natural vegetation survives or that you get a vegetation established in time, that you do not leave the system vulnerable.

KE4: So something that you could put into the soil which would make it more permeable and assist vegetation would help you on both fronts.

DE4: Certainly, yes.

KE5: So we could add to our list some sort of fertiliser

DE5: Yes.

KE6: Do you know what sort of characteristics that would have to have?

DE6: I'm afraid that I don't know much on fertilisers, I must admit, what I would say is that a natural organic mulch is better than a fertiliser.

SKE7: Peat?

DE7: I don't know that you would get peat in a semi arid environment, as such, animal...

SKE8: Organic manure?

DE8: Yes, organic manure.

SKE9: Straw?

DE9: Yes, straw is very good. because it does get that binding, and that is, you do see that, but obviously, if you are talking about remoter hill slopes you are going to get into problems.

SKE10: Is there any possibility of overcoming any of the difficulties you have been describing, by minor civil engineering so to speak, in other words building more robust banks to the terraces or is that too expensive.

DE10: Well, if the funding is there well yes, I mean, certainly there are places where they are growing vines and so forth, where obviously it is very cost effective to make it entirely into a man made environment and so they have got the terraces built up and they have got all the drainage worked out, and its all up kept, and so they are putting the man hours and machinery in, and provided that they keep it like that they are ok, but as soon as that um work was not in put then you would start to get lots of
problems built up.

BLOCK TWO - ADJACENCY PAIRS 11 - 21

SKE11: And any thing like that is presumably substantially more expensive than planting schemes.

DEI1: Yes, it really does depend upon the land use that you are trying to target up, once you start to talk about growing the cash crops you probably can start to go onto this sort of thing, but I think that if you are talking about a system where there is not the availability of manpower, then there is certainly a move not to create terraces as such but if you are using, if you are growing olive trees or whatever to use this sprinkler system, where you have got a pipe and then you have got this jet to the specific tree. I mean the problem, obviously, with irrigation is that you have also got to avoid salinization because, obviously again, when you get into that problem then you have to thinking about the chemical and not killing the plant off.

KE12: Am I right in thinking that irrigation is itself a preventative measure for soil erosion. In the right context.

DE12: In the right context, yes, provided you don’t over irrigate so that you get salinization problems, provided, I mean this bucare flood irrigation scheme, in that you are putting across the land the very medium which will cause the soil erosion if you don’t control it. With the terrace system you may very well get sheet erosion which brings all of the sediment down to the bottom of the terrace, and if in the next season you plough it up and spread it back across the slope then that’s fine, so what you have to avoid is loosing that soil completely off the field so the other thing to do is to have some strip of denser vegetation cover which will actually catch the sediment or even a mesh, some sort of mesh which will capture that sediment so that you don’t loose it.

SKE13: So, the angle on the terrace is not necessarily zero degrees?

DE13: The angle on the terrace is indeed important.

SKE14: The terraces are not necessarily horizontal then...

DE14: Oh, no, there is flood irrigation they can’t be horizontal because what they are trying to do is the whole terrace such that it tilts to the exit point, it will only be a couple of degrees. I mean otherwise if you have it above that you will start getting into the problem of rilling and then leading to gully so the whole terrace is actually constructed so that the water will actually pass across it to the exit point.
SKE15: To distribute the water

DE15: To distribute the water entirely across the field. Well that’s one scheme, another scheme is where there is a wall at the front and then the water ponds across the surface and then it does actually stay there for the few hours or whatever that it takes to drain or seep into the soil and that’s where you can get the problem that you are building up the pore pressures in the soil and you can get this sudden almost explosion through the wall that is not strong enough to restrain that pressure.

SKE16: So you are trying to balance the moisture content all the time of the soil by either drainage or irrigation.

DE16: Absolutely, if you have got irrigation it is important that the soil itself is well drained so you have got this dichotomy where you are pumping water in but you are making sure that it is well drained as well.

KE17: Is flood irrigation a poor man’s solution to the problems as opposed to using sprinklers?

DE17: Yes, I would say so yes.

SKE18: You mentioned the possibility of having a barrier of vegetation, a barrier, a growth, to prevent the soil being washed over the edge so to speak, this is something that has not had a lot of attention in the discussions we have had. Why is that do you think?

DE18: I think that perhaps it should have the attention, umm, I don’t know about Spain in this instance, but I know that one of the effective ways that if you have got a terrace which has a longer slope, a longer deeper terrace, that one thing you can do is to have strips of different crops, bands of different crops, so that if you want to have wheat or a serial crop, you then intersperse that with denser vegetation a lower one which will have a greater Matt, and of course within this you have also got to think very much about root Matt, and so that is another consideration. If say you are going to clear your vegetation, is that, are you going to disturb the root matt, because if say, you burnt the vegetation your not necessarily disturbing the matt, where as if you plough it up you loose your root matt. That is one consideration you have to bear in mind. In umm, We are going on to tropical countries, I’m sorry, but when we were talking about Brazil yesterday, the cases we were talking about where the trees were burnt rather than using the big heavy machinery which also rips out the roots as well that’s the case where you can avoid the situation of this massive erosion because you have still got the root matt, which is protecting, holding the soil together and if you can get that, then provided that you don’t have a rain intensity that overcomes that as well, then you are elevating the problem.

SKE19: So an ideal preventative measure is to have a low level vegetation of the soil with a heavy root mat.
DE19: Well, you see what one thing that you could do, is if you plough up your soil then you had sowed is to actually have a mesh a fine mesh over the soil and the seedlings can grow through that, and then you would be holding the soil as well ok, until the seedlings themselves have got established.

SKE20: Any compaction?

DE20: No, no

SKE21: Just to stop the rain splash?

DE21: Yes so that it disperses that kinetic energy and that it holds the soil together.

BLOCK THREE - ADJACENCY PAIRS - 22 - 32.

SKE22: Is it possible to quantify, in any way, the width of vegetation that you need at the edge of the terrace in relation, for instance, to slope or any other factors in order to get a significant degree of soil retention.

DE22: I'm sure that it would be, it would obviously depend upon the slope angle and the slope length so the amount of the water that you are expecting to cross that terrace and also the sediment concentration that you were expecting, if its very high concentration the it is going to fall out very much quicker than if it is very fine, if its fine then you probably have not got a problem anyway, but um there are going to be those factors relating in the calculation, I can't come up with it now but it is certainly something that you could calculate.

KE23: Would there be any conditions in which I would be right in saying, that, you have planted a crop to help retain the soil, and the roots are doing that, but you suddenly come across a condition in which the roots are breaking up the soil and actually increasing the amount of erosion?

DE23: I would not have thought so, if you were talking about tree crops you would only get that in a very extreme case

KE24: OK

DE24: That is irrelevant
KE25: So, just a recap then, you have got the following methods: mulching, putting down some cover, terracing, adding fertiliser, irrigation - two types of irrigation flood, and sprinkler, we have discussed the problems of flood irrigation putting down barriers to control the heads of water, such as walls or vegetation. Is there anything else, that would help as a preventative measure?

DE25: Well so far we have been dealing with cultivation, if you started talking about grazing. Animal husbandry, does that sound fair.

KE26: Yes

DE26: What you have got to ensure is that the animals do not concentrate on one particular localised area, you have got to keep them moving, this is if you are on a vulnerable slope.

SKE27: So don't overgraze?

DE27: So don't overgraze, you don't want them to lower the vegetation below that critical 30 percent threshold because if you do then you are getting into problems. Not only that but you are also talking about breaking up the soil surface, even if you are at 40% vegetation you have probably still got bare soil, what usually you find there is a surface crust, but if they break up that surface crust, or goats or whatever do actually have these kind of hollows or what ever, and if they start to create that sort of thing then you are going to get a flow convergence, so you have always got to get it so that they keep moving.

SKE28: Are there particular types of animals which would be better or worse from this point of view?

DE28: Certainly, cattle are probably not a very good idea on steep slopes or vulnerable slopes.

SKE29: Is that because of the steepness or the way that they graze?

DE29: Because of the way that they graze, and the nature of there... to be honest you don't get many cattle as far as I know but that is in Spain, I'm thinking on a rather limited level there, if you talk about Africa where of course you have got, but cattle anyway, you have got humm but cattle anyway you would not have on steep slopes, you would avoid the larger animal putting them on the vulnerable slopes.

SKE30: Larger animal more weight

DE30: Yes, I don't really know all the details here but it will depend on the type of trampling that they will put onto the soil, so if you have got, umm species of cattle that does have hoofs or what ever that are more...
SKE31: Is it possible to work out the capacity of the land for supporting a certain amount of cattle, is that possible.

DE31: Yes, that would be possible certainly, you would calculate that from average slope, from level of vegetation, from vegetation type. so for each animal you were thinking of you would go through and see which was the most suitable.

KE32: Can animals ever have a beneficial effect?

DE32: Yes, in that their manure is going to be a fertiliser.

BLOCK FOUR - ADJACENCY PAIRS - 33 - 44.

SKE33: Is the compaction caused by their presence or their hoofs cause more damage.

DE33: Well you see, if you have got compaction you will be decreasing the infiltration, that's part of the problem, is that in fact where you have compacted soil you may actually get a concentration of the flow, this is one of the, I think with grazing animals on of the difficulties is that they can produce the conditions which may cause a concentration of flow and my bias is obviously in channelised flow, erosion and that may have come through by the way I'm talking and as far as I'm concerned that is the very condition that you try to avoid.

SKE34: So they can encourage rills to turn into gullies?

DE34: Yes, yes. having said which of course they can also, if you have got a rill or a gully they destroy, they destroy the rill by there trampling effect, so in that sense, I would have thought that was an extreme case because rills anyway are ephemeral, because if you have got the situation where a rill will form, it will form again in the next storm event anyway and the trampling will cause compaction and decrease infiltration, having said which if there is an excess of trampling then their manure production will be of benefit, provided you have got the balance of vegetation growth your alright, its just as soon as you tip that balance.

SKE35: There is a value from the manure of animals on the land, there is also presumably a value from the fact that if you just leave the land completely fallow you get nothing from it, where as if you animals grazing you do get some benefits from that point of view, in other respects, the use of husbandry appears to be anti-beneficial, is that right?

DE35: Only if, it is causing too greater stress on the environment.
SKE36: There presence inevitably causes stress

DE36: Depends how resistant that environment is, if the resistance of the environment is such that it can take that level of stress from that given breed of animal then you are alright.

SKE37: There is a threshold level

DE37: Yes, there will be a threshold level, if you like, you are always considering your level of force, your level of resistance, basically if you take it on such a broad level, you know, don't think about all the various factors, but if you can assess forces which may be put onto that environment/system, and the resistance that system can offer and providing you have got either the resistance well you want that resistance to be greater than the force that is being put onto the system and once you start getting the two values get closer and closer obviously you are getting towards a greater risk and that's really the point, that, once you do then you have got greater risk from a freak storm or whatever.

KE38: It seems to me at this point now that there is room for exploring many interactions for example, if you have got something covering the soil presumably you can not have anything grazing on that surface as well.

DE38: What I have forgotten to say, is if you have a drought, I mean that is another possibility, as well, if you have a drought happen, I mean that I was talking about freak storms well that's being a little one side, if you have a drought as well then you are decreasing the vegetation cover so that's another factor that you have got to consider, is that through out the year the resistance offered by the environment may in fact change, so at the end of the summer when the vegetation is at its lowest its also at its most vulnerable, and its at the end of the summer when you may get these freak October storms in Spain, you know, In Spain, but also you will get the seasonal change in all semi arid environments where at the end of the summer you have got this very critical, you may well have conditions of the soil in which infiltration is not going to be very fast, although overland flow may in fact in turn may be very rapid.

SKE39: So, back to keeping the moisture content of the soil......

DE39: Yes, you have got to be very careful that the end of the summer you have not created the situation where in that environment is not going to be very vulnerable to the first seasonal storms, What works for you in some ways is that during the summer you may in actual fact, depending upon the soil type, the soil may have in fact of broken up, there may be clods or whatever that in actual fact be picked up immediately and take up all of the transporting capacity of the overland flow but in many case what happens is when you get that first rain all the soil surface just goes into, a kind of, it just seals all of the cracks seal and everything and you just get a sheet going right across and if you have got that happening, and you have got a low vegetation cover because it has been dying off during the summer then that system is very vulnerable and you have got to make quite sure that what you have done through your animal husbandry, or your cultivation or whatever has not made it such that it will be put across the threshold to excessive soil loss. There is actually, you can calculate the amount of soil that you can afford to lose off a given field if you know the rate of soil production. I think, one that the Americans came up with when they were investigating the ULSE it was something like 25 tons per...
does not sound quite right.........

SKB40: Soil production as in....

DE40: The actual creation, formation of soil itself, you can actually calculate the balance so that you never want the soil loss to be greater than the formation of the soil itself.

SKB41: What leads to soil creation?

DE41: Ah,

SKB42: What encourages that?

DE42: Ah, well again the introduction of organic matter is critical.

KE43: Is there any major preventative measure that we have not covered?

DE43: Ummm, no ok...
Appendix C.3
Pair one (KEl & DEI): Using technique one.

Step 1. Examine the knowledge engineers text for conversational coherence.

Knowledge engineer.

KEl: What methods exist for preventing soil erosion?

Alignment: Type 3.3. Directing.

- The knowledge engineer outlines the area of the domain to be focused upon - the current context.

Domain expert.

DEI: Right, well if you consider that the critical factors are: the rainfall characteristics, the soil characteristics and the topography, you have got to consider the degree of slope, and if the slope is not very great, then you can probably control erosion by altering the soil characteristics in some way, by say mulching, or putting down one of those cause meshes, something like this, for getting up the organic matter level, something like this, so you are encouraging a greater cohesion of the soil and also by increasing the roughness of the surface with this machine which puts holes in there you are increasing the surface retention, if you like, of water. So on low slope you have probably not got that many problems. Unless, for instance you have got flood irrigation where the land has been put into these terrace so initially there would have been a slope but say you are inheriting this agricultural land which is a terrace formation and then, well, for a start you have got to upkeep these terraces because one of the problems in Spain particularly now is that you have got this movement of the farmers into the cities and the terraces are going into disuse and those front walls are not been upkept and as soon as those start falling down then you have got this horrific problem of the pore pressures being increased then the water suddenly just gushes; bursts, through the wall. So if you have inherited that sort of thing then you have got to keep the upkeep going - you have to put in the man hours basically, if you are inheriting that, that's the problem with that system. It is quite intensive with man hours.

Step 2. [break down the text into smaller segments - approx. a paragraph in size]

DEI.1: Right, well if you consider that the critical factors are: the rainfall characteristics, the soil characteristics and the topography, you have got to consider the degree of slope, and if the slope is not very great, then you can probably control erosion by altering the soil characteristics in some way, by say mulching, or putting down one of these cause meshes, something like this, for getting up the organic matter level, something like this, so you are encouraging a greater cohesion of the soil and also by increasing the roughness of the surface with this machine which puts holes in there you are increasing the surface retention, if you like, of water.
DEI.2: So on low slope you have probably not got that many problems. Unless, for instance you have got flood irrigation where the land has been put into those terrace so initially there would have been a slope but say you are inheriting this agricultural land which is a terrace formation and then, well, for a start you have got to upkeep these terraces because one of the problems in Spain particularly now is that you have got this movement of the farmers into the cities and the terraces are going into disuse and those front walls are not been upkept and as soon as those start falling down then you have got this horrific problem of the pore pressures being increased then the water suddenly just gushes; bursts, through the wall.

DEI.3: So if you have inherited that sort of thing then you have got to keep the upkeep going - you have to put in the man hours basically, if you are inheriting that, that's the problem with that system. It is quite intensive with man hours.

Step 3. [Take the linear/flat elicited representation and produce a structured representation.]

[Refine DEI.1]

[Rename as 1]

1. Right, well if you consider that the critical factors are: the rainfall characteristics, the soil characteristics and the topography, you have got to consider the degree of slope, and if the slope is not very great, then you can probably control erosion by altering the soil characteristics in some way, by say mulching, or putting down one of these cause meshes, something like this, for getting up the organic matter level, something like this, so you are encouraging a greater cohesion of the soil and also by increasing the roughness of the surface with this machine which puts holes in there you are increasing the surface retention, if you like, of water.

   [Remove any noise:"Right, well if you consider that"]

1. Right, well if you consider that the critical factors are: the rainfall characteristics, the soil characteristics and the topography, you have got to consider the degree of slope, and if the slope is not very great, then you can probably control erosion by altering the soil characteristics in some way, by say mulching, or putting down one of these cause meshes, something like this, for getting up the organic matter level, something like this, so you are encouraging a greater cohesion of the soil and also by increasing the roughness of the surface with this machine which puts holes in there you are increasing the surface retention, if you like, of water.

[Divide up 1. on the connective AND]

1.1 the critical factors are: the rainfall characteristics, the soil characteristics and the topography, you have got to consider the degree of slope

1.2 and if the slope is not
very great, then you can probably control erosion by altering the soil characteristics in some way, by say mulching, or putting down one of these cause meshes, something like this, for getting up the organic matter level, something like this, so you are encouraging a greater cohesion of the soil and also by increasing the roughness of the surface with this machine which puts holes in there you are increasing the surface retention, if you like, of water.

1.2.1.1 and if the slope is not very great, then you can probably control erosion by altering the soil characteristics in some way, by say mulching, or putting down one of these cause meshes, something like this, for getting up the organic matter level, something like this,

1.2.2 so you are encouraging a greater cohesion of the soil and also by increasing the roughness of the surface with this machine which puts holes in there you are increasing the surface retention, if you like, of water.

1.1 the critical factors are: the rainfall characteristics, the soil characteristics and the topography, you have got to consider the degree of slope

1.2.1.2 or putting down one of these cause meshes, something like this, for getting up the organic matter level, something like this,

1.2.2 so you are encouraging a greater cohesion of the soil and also by increasing the roughness of the surface with this machine which puts holes in there you are increasing the surface retention, if you like, of water.

1.1 the critical factors are: the rainfall characteristics, the soil characteristics and the topography, you have got to consider the degree of slope

1.2.1.2 or putting down one of these cause meshes, something like this, for getting up the organic matter level, something like this,
1.2.2.1 so you are encouraging a greater cohesion of the soil

1.2.2.2 and also by increasing the roughness of the surface with this machine which puts holes in there you are increasing the surface retention, if you like, of water.

[Divide up 1.2.1.2. on the keyword FOR]

1.1 the critical factors are: the rainfall characteristics, the soil characteristics and the topography, you have got to consider the degree of slope

1.2.1.1 and if the slope is not very great, then you can probably control erosion by altering the soil characteristics in some way, by say mulching,

1.2.1.2.1 or putting down one of these cause meshes, something like this,

1.2.1.2.2 for getting up the organic matter level, something like this,

1.2.1.2 so you are encouraging a greater cohesion of the soil

1.2.2.2 and also by increasing the roughness of the surface with this machine which puts holes in there you are increasing the surface retention, if you like, of water.

[Divide up 1.2.1.1. on the keywords BY SAY]

1.1 the critical factors are: the rainfall characteristics, the soil characteristics and the topography, you have got to consider the degree of slope

1.2.1.1.1 and if the slope is not very great, then you can probably control erosion by altering the soil characteristics in some way,

1.2.1.1.2 by say mulching,

1.2.1.2.1 or putting down one of these cause meshes, something like this,

1.2.1.2.2 for getting up the organic matter level, something like this,

1.2.2.1 so you are encouraging a greater cohesion of the soil

1.2.2.2 and also by increasing the roughness of the surface with this machine which puts holes in there you are increasing the surface retention, if you like, of water.

[contour]
characteristics and the topography, you have got to consider the degree of slope.

And if the slope is not very great, then you can probably control erosion by altering the soil characteristics in some way, by say mulching, or putting down one of these cause meshes, something like this, for getting up the organic matter level, something like this, so you are encouraging a greater cohesion of the soil and also by increasing the roughness of the surface with this machine which puts holes in there you are increasing the surface retention, if you like, of water.

**Distillation**

**Fact:** critical_factors are = \{rainfall_cht, soil_cht, topography_cht\}

**Fact:** degree_of_slope eof topography

**Heuristic:** IF degree_of_slope < great THEN erosion_control = soil_cht.

**Fact:** erosion_control = mulching,

**Fact:** erosion_control = cause_meshes

**Rule:** IF erosion_control = cause_mesh THEN organic_matter_level = increases

**Rule:** IF organic_matter_level = increases THEN soil_cohesion = increases

**Rule:** IF surface_roughness = increase THEN surface_retention_ofwater = increase

**Heuristic:** IF degree_of_slope < great THEN erosion_control = soil_cht.

[redefine DL1.2]
So on low slope you have probably not got that many problems. Unless, for instance you have got flood irrigation where the land has been put into these terrace so initially there would have been a slope but say you are inheriting this agricultural land which is a terrace formation and then, well, for a start you have got to upkeep these terraces because one of the problems in Spain particularly now is that you have got this movement of the farmers into the cities and the terraces are going into disuse and those front walls are not been upkept and as soon as those start falling down then you have got this horrific problem of the pore pressures being increased then the water suddenly just gushes; bursts, through the wall.
formation and then, well, for a start you have got to upkeep these terraces

2.2.2 because one of the problems in Spain particularly now is that you have got this movement of the farmers into the cities and the terraces are going into disuse and those front walls are not been upkept and as soon as those start falling down then you have got this horrific problem of the pore pressures being increased then the water suddenly just gushes; bursts, through the wall.

[divide up 2.2.2 on the connective AND]

2.1.1 So on low slope you have probably not got that many problems.

2.1.2 Unless, for instance you have got flood irrigation where the land has been put into these terrace so initially there would have been a slope

2.2.1 but say you are inheriting this agricultural land which is a terrace formation and then, well, for a start you have got to upkeep these terraces

2.2.2.1 because one of the problems in Spain particularly now is that you have got this movement of the farmers into the cities and the terraces are going into disuse and those front walls are not been upkept

2.2.2.1 and as soon as those start falling down then you have got this horrific problem of the pore pressures being increased then the water suddenly just gushes; bursts, through the wall.

[contour]

So on low slope you have probably not got that many problems.

 Unless, for instance you have got flood irrigation where the land has been put into these terrace so initially there would have been a slope

 but say you are inheriting this agricultural land which is a terrace formation and then, well, for a start you have got to upkeep these terraces

 because one of the problems in Spain particularly now is that you have got this movement of the farmers into the cities and the terraces are going into disuse and those front walls are not been upkept

 and as soon as those start falling down then you have got this horrific problem of the pore pressures being increased then the water suddenly just gushes; bursts, through the wall.

[Distillation]

Heuristic: IF section_of_slope = low THEN erosion_problem = low

Noise:
RULE: IF erosion_control = terrace AND upkeep_needed = true

IF erosion_control = terrace AND unkeep_applied = false
  THEN front_wall = not upkept

RULE: IF erosion_control = terrace AND unkeep_applied = false
  THEN pore_pressure = increase

RULE: IF pore_pressure > limit THEN Wall_collapse = true

Heuristic: IF section_of_slope = low THEN erosion_problem = low

RULE: IF erosion_control = terrace
  THEN upkeep_needed = true

RULE: IF erosion_control = terrace AND unkeep_applied = false
  THEN front_wall = not upkept

RULE: IF erosion_control = terrace AND unkeep_applied = false
  THEN pore_pressure = increase

RULE: IF pore_pressure > limit
  THEN Wall_collapse = true

3. So if you have inherited that sort of thing then you have got to keep the upkeep going - you have to put in the man hours basically, if you are inheriting that, that's the problem with that system. It is quite intensive with man hours.

3.1. So if you have inherited that sort of thing then you have got to keep the upkeep going - you have to put in the man hours basically.

3.2. If you are inheriting that, that's the problem with that system. It is quite intensive with man hours.
Heuristic: erosion_control = terraces THEN upkeep_hours = high

Fact1: critical_factors are = \{rainfall_ccht, soil_ccht, topography_ccht\}
Fact2: degree_of_slope of topography
Fact3: erosion_control = mulching,
Fact4: erosion_control = cause_meshes

Rule1: IF erosion_control = cause_mesh
   THEN organic_matter_level = increases
Rule2: IF organic_matter_level = increases
   THEN soil_cohesion = increases
Rule3: IF surface_roughness = increase
   THEN surface_retention_of_water = increase
Rule4: IF erosion_control = terrace
   THEN upkeep_needed = true
Rule5: IF erosion_control = terrace AND upkeep_applied = false
   THEN front_wall = not_upkept
Rule6: IF erosion_control = terrace AND upkeep_applied = false
   THEN pore_pressure = increase
Rule7: IF pore_pressure > limit
   THEN Wall_collapse = true

Pair two. (KE2 & DE2).

Step 1. [examine the knowledge engineer’s text for conversational coherence] 

Knowledge engineer.

KE2: These 2 techniques you mentioned, one, mulching and the other actually putting down some thing that actually physically cover the surface. Now would you describe these as functionally equivalent procedures or are there different circumstances under which you would rather use one than the other.

Alignment: Type 3.1. Clarifying.

Domain expert.

DE2: I think certainly that if you were going to put down this mesh as I was saying, that you probably would not do that for a lower slope unless you for say, had this flood irrigation problem, in which case you might start to introduce that. The key thing would be to keep your ground cover, so that, if you have got Olive trees to keep a ground cover around it as well, otherwise you are going to get this problem. Ah if you have got fallow, if you have got a natural vegetation then you probably have not got very many problems, what’s happening at the moment is that there perceiving that there is a problem, and so on the steeper slopes if we go onto those, they perceive that there is a problem and then bulldozing all of the natural vegetation away and putting in terraces and then putting in forest, trees, but if the trees don’t survive which is quite likely in semi arid environments, they die and you get these terraces which are bare and the whole system goes berserk because that is exactly how you get a concentration of water in that the water concentrates at the back of the terraces and in any drains and unless you have designed it so that
it is also resistant to a freak storm, which you can get every 20, 30 or 40 years, then it's just that freak storm which will tip it over the balance and once you have got these channels, these gullies or whatever, then you have lost your gamble if you like.

Step 2. [break down the text into smaller segments ~ approx. a paragraph in size]

DE2.1: I think certainly, that if you were going to put down this mesh as I was saying, that you probably would not do that for a lower slope unless you for say, had this flood irrigation problem, in which case you might start to introduce that. The key thing would be to keep your ground cover, so that, if you have got Olive trees to keep a ground cover around it as well, otherwise you are going to get this problem.

DE2.2 Ah if you have got fallow, if you have got a natural vegetation then you probably have not got very many problems, what's happening at the moment is that there perceiving that there is a problem, and so on the steeper slopes if we go onto those, they perceive that there is a problem and then bulldozing all of the natural vegetation away and putting in terraces and then putting in forest, trees, but if the trees don't survive which is quite likely in semi arid environments, they die and you get these terraces which are bare and the whole system goes berserk because that is exactly how you get a concentration of water in drains and unless you have designed it so that it is also resistant to a freak storm, which you can get every 20, 30 or 40 years, then it's just that freak storm which will tip it over the balance and once you have got these channels, these gullies or whatever, then you have lost your gamble if you like.

Step 3. [Take the linear/flat elicited representation and produce a structured representation]

[redefine DE2.1]

[rename as 1.]

1. I think certainly, that if you were going to put down this mesh as I was saying, that you probably would not do that for a lower slope unless you for say, had this flood irrigation problem, in which case you might start to introduce that. The key thing would be to keep your ground cover, so that, if you have got Olive trees to keep a ground cover around it as well, otherwise you are going to get this problem.

[remove any noise: none]

[divide up 1. on the marker "."]

1.1 I think certainly, that if you were going to put down this mesh as I was saying, that you probably would not do that for a lower slope unless you for say, had this flood irrigation problem, in which case you might start to introduce that.

1.2. The key thing would be to keep your ground cover, so that, if you have got Olive trees to keep a ground cover around it as well, otherwise you are going to get this problem.

[divide up 1.1 on the keyword UNLESS]
1.1.1 I think certainly, that if you were going to put down this mesh as I was saying, that you probably would not do that for a lower slope.

1.1.2 unless you for say, had this flood irrigation problem, in which case you might start to introduce that.

1.2. The key thing would be to keep your ground cover, so that, if you have got Olive trees to keep a ground cover around it as well, otherwise you are going to get this problem.

RULE: IF erosion_control = mesh THEN section_of_slope = not lower

RULE IF section_of_slope = lower

AND erosion_problem = flood_irrigation

THEN erosion_control = mesh

RULE: IF ground_cover = high THEN erosion_rate = low

EXAMPLE: IF crop = Olive_trees AND ground_cover = low

THEN erosion_problem = high

RULE: IF erosion_control = mesh THEN degree_of_slope = not lower

RULE: IF degree_of_slope = lower AND erosion_problem = flood_irrigation

THEN erosion_control = mesh

RULE: IF ground_cover = high THEN erosion_rate = low
EXAMPLE: IF crop = Olive_trees AND ground_cover = low
THEN erosion_problem = high

[redefine 2.2]

[rename as 2.]

2. Ah if you have got fallow, if you have got a natural vegetation then you probably have not got very many problems, what's happening at the moment is that there perceiving that there is a problem, and so on the steeper slopes if we go onto those, they perceive that there is a problem and then bulldozing all of the natural vegetation away and putting in terraces and then putting in forest, trees, but if the trees don't survive which is quite likely in semi arid environments, they die and you get these terraces which are bare and the whole system goes berserk because that is exactly how you get a concentration of water in drains and unless you have designed it so that it is also resistant to a freak storm, which you can get every 20, 30 or 40 years, then its just that freak storm which will tip it over the balance and once you have got these channels, these gullies or whatever, then you have lost your gamble if you like.

[remove any noise]

2. if you have got fallow, if you have got a natural vegetation then you probably have not got very many problems, what's happening at the moment is that there perceiving that there is a problem, and so on the steeper slopes if we go onto those, they perceive that there is a problem and then bulldozing all of the natural vegetation away and putting in terraces and then putting in forest, trees, but if the trees don't survive which is quite likely in semi arid environments, they die and you get these terraces which are bare and the whole system goes berserk because that is exactly how you get a concentration of water in drains and unless you have designed it so that it is also resistant to a freak storm, which you can get every 20, 30 or 40 years, then its just that freak storm which will tip it over the balance and once you have got these channels, these gullies or whatever, then you have lost your gamble if you like.

[divide up 2. on keyword WHATS]

2.1 if you have got fallow, if you have got a natural vegetation then you probably have not got very many problems,

2.2. what's happening at the moment is that there perceiving that there is a problem, and so on the steeper slopes if we go onto those, they perceive that there is a problem and then bulldozing all of the natural vegetation away and putting in terraces and then putting in forest, trees, but if the trees don't survive which is quite likely in semi arid environments, they die and you get these terraces which are bare and the whole system goes berserk because that is exactly how you get a concentration of water in drains and unless you have designed it so that it is also resistant to a freak storm, which you can get every 20, 30 or 40 years, then its just that freak storm which will tip it over the balance and once you have got these channels, these gullies or whatever, then you have lost your gamble if you like.

[divide up 2.2. on BECAUSE]
2.1 if you have got fallow, if you have got a natural vegetation then you probably have not got very many problems,

2.2.1 what’s happening at the moment is that there perceiving that there is a problem, and so on the steeper slopes if we go onto those, they perceive that there is a problem and then bulldozing all of the natural vegetation away and putting in terraces and then putting in forest, trees, but if the trees don’t survive which is quite likely in semi arid environments, they die and you get these terraces which are bare and the whole system goes berserk

2.2.2 because that is exactly how you get a concentration of water in drains and unless you have designed it so that it is also resistant to a freak storm, which you can get every 20, 30 or 40 years, then its just that freak storm which will tip it over the balance and once you have got these channels, these gullies or whatever, then you have lost your gamble if you like.

[divide up 2.2.2 on the keywords "AND UNLESS"]

2.1 if you have got fallow, if you have got a natural vegetation then you probably have not got very many problems,

2.2.1 what’s happening at the moment is that there perceiving that there is a problem, and so on the steeper slopes if we go onto those, they perceive that there is a problem and then bulldozing all of the natural vegetation away and putting in terraces and then putting in forest, trees, but if the trees don’t survive which is quite likely in semi arid environments, they die and you get these terraces which are bare and the whole system goes berserk

2.2.2 because that is exactly how you get a concentration of water in drains

2.2.2.2 and unless you have designed it so that it is also resistant to a freak storm, which you can get every 20, 30 or 40 years, then its just that freak storm which will tip it over the balance and once you have got these channels, these gullies or whatever, then you have lost your gamble if you like.

[divide up 2.2.1 on the connectives "AND SO"]

2.1 if you have got fallow, if you have got a natural vegetation then you probably have not got very many problems,

2.2.1.1 what’s happening at the moment is that there perceiving that there is a problem,

2.2.1.2 and so on the steeper slopes if we go onto those, they perceive that there is a problem and then bulldozing all of the natural vegetation away and putting in terraces and then putting in forest, trees, but if the trees don’t survive which is quite likely in semi arid environments, they die and you get these terraces which are bare and the whole system goes berserk

2.2.2 because that is exactly how you get a concentration of water in drains
and unless you have designed it so that it is also resistant to a freak storm, which you can get every 20, 30 or 40 years, then it's just that freak storm which will tip it over the balance and once you have got these channels, these gullies or whatever, then you have lost your gamble if you like.

[contour]

if you have got fallow, if you have got a natural vegetation then you probably have not got very many problems,

what's happening at the moment is that there perceiving that there is a problem,

and so on the steeper slopes if we go onto those, they perceive that there is a problem and then bulldozing all of the natural vegetation away and putting in terraces and then putting in forest, trees, but if the trees don't survive which is quite likely in semi arid environments, they die and you get these terraces which are bare and the whole system goes berserk

because that is exactly how you get a concentration of water in

drains

and unless you have designed it so that it is also resistant to a freak storm, which you can get every 20, 30 or 40 years, then it's just that freak storm which will tip it over the balance and once you have got these channels, these gullies or whatever, then you have lost your gamble if you like.

[distillation]

[rule: IF ground_cover = natural_vegetation THEN erosion_rate = low]

[rule: IF erosion_control = terrace AND weather = freak_storm AND terrace_design = poor THEN erosion_rate = significant]

Pair three. (KE3, DE3)

Step one. [examine the knowledge engineers text for conversational coherence]
KE3: So one key under those circumstances would be insuring that the vegetation does survive.

Alignment: Type 3.1. Clarifying.

KE3: So one key under [bulldozing all of the natural vegetation away and putting in terraces and then putting in forest, trees, but if the trees don't survive which is quite likely in semi arid environments, they die and you get these terraces which are bare and the whole system goes berserk] circumstances would be insuring that the vegetation does survive.

Domain expert.

DE3: Either that the natural vegetation survives or that you get a vegetation established in time, that you do not leave the system vulnerable.

Step 2. [Break down the text into smaller segments: unnecessary]

Step 3. [Take the linear/flat elicited representation and produce a structured representation.]

[rename as 3]

3. Either that the natural vegetation survives or that you get a vegetation established in time, that you do not leave the system vulnerable.

[divide up 3. on the connective OR]

3.1 Either that the natural vegetation survives

3.2 or that you get a vegetation established in time, that you do not leave the system vulnerable.

FACT: erosion_control = vegetation_cover

HEURISTIC: IF surface_cover = bare THEN erosion_problem = vulnerable
FACT: erosion_control = vegetation_cover

HEURISTIC: IF surface_cover = bare THEN erosion_problem = vulnerable

Pair four. (KE4 & DE4)

Step 1. (examine KE4 text for conversational coherence)

Knowledge engineer.

KE4: So something that you could put into the soil which would make it more permeable and assist vegetation would help you on both fronts.

Alignment: Type 3.1 Clarifying.

DE4: Certainly, yes.

Step 2. (Break down the text into smaller segments; unnecessary)

Step 3. (Take the linear/flat elicited representation and produce a structured representation; unnecessary)

[reformulate KE4 as DE4 answer is confirmation of KE4 question]

[rename as 4]

[divide 4. on the OR]

4.1 So something that you could put into the soil which would make it more permeable and assist vegetation would help you on insuring vegetation survives

4.2 or that you get vegetation established

[contour]

|So something that you could put into the soil which would make it more | permeable and assist vegetation would help you on insuring vegetation |
|survives|
|or that you get vegetation established|

[distillation]
FACT: erosion_control = soil_additive

RULE: IF erosion_control = soil additive THEN ground_cover = increases

Pair five. (KE5 & DE5)

Step 1. [examine KE5 text for conversational coherence]

Knowledge engineer.

KE5: So we could add to our list some sort of fertiliser

Alignment: Type 7. Qualifier.

Domain expert.

DE5: Yes

Step 2. [Break down the text into smaller segments: unnecessary]

Step 3. [Take the linear/flat elicited representation and produce a structured representation]

[reformulate KE5 as DE5 answer is confirmation of KE5 question]

[rename as 5]

So we could add to our list some sort of fertiliser

[contour]

FACT: erosion_control = fertiliser

[distillation]

FACT: erosion_control = fertiliser

KE6: Do you know what sort of characteristics that would have to have?
Alignment: Type 3.1 Clarifier

[referral of KE6 on THAT]

KE6: Do you know what sort of characteristics that [fertiliser] would have to have?

Domain expert.

------------

DE6: I'm afraid that I don't know much on fertilises, I must admit, what I would say is that a natural organic mulch is better than a fertiliser.

Step 2. [Break down the text into smaller segments: unnecessary]

Step 3. [Take the linear/flat elicited representation and produce a structured representation]

[rename as 6]

6. I'm afraid that I don't know much on fertilises, I must admit, what I would say is that a natural organic mulch is better than a fertiliser.

[remove any noise: none]

[divide 6 on WHAT]

6.1 I'm afraid that I don't know much on fertilises, I must admit,

6.2 what I would say is that a natural organic mulch is better than a fertiliser.

[contour]

-----------------------------------------------
[I'm afraid that I don't know much on fertilises, I must admit,]

[What I would say is that a natural organic mulch is better than a]

[fertiliser.

[distillation]

-----------------------------------------------
[Noise]

-----------------------------------------------
[FACT: erosion_control = natural_organic_mulch]

[FACT: erosion_control = fertiliser]

[HEURISTIC: IF erosion_control = needed]

[THEN natural_organic_mulch > erosion_control = fertiliser]

[ list]

FACT: erosion_control = natural_organic_mulch
FACT: erosion_control = fertiliser
HEURISTIC: IF erosion_control = needed
Then natural_organic_mulch > erosion_control = fertiliser

Pair seven. (KE7 & DE7)
----------

Step 1. [Examine KE7 text for conversational coherence]
----------

Knowledge engineer.
-------------------

SKE7: Peat?

Alignment: Type 2. Formulation.
-------------------

Domain expert.
-------------------

DE7: I don't know that you would get peat in a semi arid environment, as such. Animal...

Step 2. [Break down the text into smaller segments: unnecessary]
----------

Step 3. [Take the linear/flat elicited representation and produce a structured representation]
----------

[Name as 7]

7. I don't know that you would get peat in a semi arid environment, as such. Animal...

[Contour]

RULE: IF environment = semi arid THEN soil_type not = peat

[Distillation]

RULE: IF environment = semi arid THEN soil not = peat

Pair eight. (KE8 & DE8)
----------

Step 1. [Examine KE8 text for conversational coherence]
----------

Knowledge engineer.
-------------------

SKE8: Organic manure?
Alignment: Type 2. Formulation.

Domain expert.

---

DE8: Yes, organic manure.

Step 2. [Break down the text into smaller segments: unnecessary]

Step 3. [Take the linear/flat elicited representation and produce a structured representation]

[rename as 8]

8. Yes, organic manure.

[contour]

FACT: erosion_control = organic_manure

Pair nine. (KE9 & DE9)

Step 1. [examine KE9 text for conversational coherence]

Knowledge engineer.

---

KE9: Straw?

Alignment: Type 2. Formulation.

Domain expert.

---

DE9: Yes, straw is very good because it does get that binding, and that is, you do see that, but obviously if you are talking about remoter hill slopes you are going to get into problems.

Step 2. [Break down the text into smaller segments: unnecessary]

Step 3. [Take the linear/flat elicited representation and produce a structured representation]
9. Yes, straw is very good, because it does get that binding, and that is, you do see that, but obviously if you are talking about remoter hill slopes you are going to get into problems.

[divide 9 on the BECAUSE]

9.1 Yes, straw is very good.

9.2 because it does get that binding, and that is, you do see that, but obviously if you are talking about remoter hill slopes you are going to get into problems.

[divide 9.2 on the connective AND]

9.1 Yes, straw is very good.

9.2.1 because it does get that binding,

9.2.2 and that is, you do see that, but obviously if you are talking about remoter hill slopes you are going to get into problems.
SKE10: Is there any possibility of overcoming any of the difficulties you have been describing, by minor civil engineering so to speak, in other words building more robust banks to the terraces or is that too expensive.

Alignment: Type 3.4 Requesting.

Domain expert.

DE10: Well, if the funding is there well yes, I mean, certainly there are places where they are growing vines and so forth, where obviously it is very cost effective to make it entirely into a man made environment and so they have got the terraces built up and they have got all the drainage worked out, and it's all up kept, and so they are putting the man hours and machinery in, and provided that they keep it like that they are ok, but as soon as that um work was not in put then you would start to get lots of problems built up.

Step 2. [Break down the text into smaller segments: unnecessary] 

Step 3. [Take the linear/flat elicited representation and produce a structured representation]

[rename as 10]

10. Well, if the funding is there well yes, I mean, certainly there are places where they are growing vines and so forth, where obviously it is very cost effective to make it entirely into a man made environment and so they have got the terraces built up and they have got all the drainage worked out, and its all up kept, and so they are putting the man hours and machinery in, and provided that they keep it like that they are ok, but as soon as that um work was not in put then you would start to get lots of problems built up.

[divide up 10 on the connective "AND SO"]

10.1 Well, if the funding is there well yes, I mean, certainly there are places where they are growing vines and so forth, where obviously it is very cost effective to make it entirely into a man made environment

10.2 and so they have got the terraces built up and they have got all the drainage worked out, and its all up kept, and so they are putting the man hours and machinery in, and provided that they keep it like that they are ok, but as soon as that um work was not in put then you would start to get lots of problems built up.

[divide up 10.2 on the connective BUT]

10.1 Well, if the funding is there well yes, I mean, certainly there are places where they are growing vines and so forth, where obviously it is very cost effective to make it entirely into a man made environment

10.2.1 and so they have got the terraces built up and they have got all the drainage worked out, and its all up kept, and so they are putting the man hours and machinery in, and provided that they keep it like that
they are ok,

10.2.2 but as soon as that work was not put then you would start to get lots of problems built up.

[contour]

They are ok, but as soon as that work was not put then you would start to get lots of problems built up.

[distillation]

HEURISTIC: IF crop = vines THEN man_made_environment = very_cost_effective

HEURISTIC: IF erosion_control = man_made_environment
AND man_made_environment_level = total
THEN terraces = built_up AND drainage = worked_out
AND drainage = all_round AND upkeep_applied = true
AND man_hours = full AND machinery = true

RULE: IF erosion_control = man_made_environment
AND man_made_environment_level = total
THEN erosion_rate = controlled

RULE: IF erosion_control = man_made_environment
AND man_made_environment_level = not total
THEN erosion = not controlled AND erosion_rate = increases
Appendix C.4
Contour diagram.

**Fact:** critical_factors are \{rainfall_{cht}, soil_{cht}, topography_{cht}\}

**Fact:** degree_of_slope is an element of topography

**Heuristic:** IF degree_of_slope < great THEN erosion_control = soil_{cht}.

**Fact:** erosion_control = mulching,

**Fact:** erosion_control = cause_{meshes}.

**Rule:** IF erosion_control = cause_{mesh} THEN organic_matter_level = increases.

**Rule:** IF organic_matter_level = increases THEN soil_cohesion = increases.

**Rule:** IF surface_roughness = increase THEN surface_retention_of_water = increase.

**Heuristic:** IF section_of_slope = low THEN erosion_problem = low.

**Rule:** IF erosion_control = terrace AND upkeep_needed = true THEN front_wall = not upkept.

**Rule:** IF erosion_control = terrace AND upkeep_applied = false THEN pore_pressure = increase.

**Rule:** IF pore_pressure > limit THEN Wall_collapse = true.

**Heuristic:** erosion_control = terraces THEN upkeep_hours = high.

**Rule:** IF erosion_control = mesh THEN section_of_slope = not lower.

**Rule:** IF section_of_slope = lower AND erosion_problem = flood_irrigation THEN erosion_control = mesh.

**Rule:** IF ground_cover = high THEN erosion_rate = low.

**Example:** IF crop = Olive_trees AND ground_cover = low THEN erosion_problem = high.

**Rule:** IF ground_cover = natural_vegetation THEN erosion_rate = low.

**Noise**
RULE: IF erosion_control = terrace AND weather = freak_storm
AND terrace_design = poor
THEN erosion_rate = significant

FACT: erosion_control = vegetation_cover

HEURISTIC: IF surface_cover = bare THEN erosion_problem = vulnerable

FACT: erosion_control = soil_additive

RULE: IF erosion_control = soil additive THEN ground_cover = increases

FACT: erosion_control = fertiliser

Noise

FACT: erosion_control = natural_organic_mulch
FACT: erosion_control = fertiliser

HEURISTIC: erosion_control = natural_organic_mulch >

RULE: IF environment = semi_arid THEN soil_type not = peat

FACT: erosion_control = organic_manure

FACT: erosion_control = straw

RULE: IF erosion_control = straw THEN soil_binding = high

Noise

HEURISTIC: IF crop = vines THEN man_made_environment = very_cost_effective

HEURISTIC: IF erosion_control = man_made_environment
AND man_made_environment_level = total
THEN terraces = built_up AND drainage = worked_out
AND drainage = all_round AND upkeep_applied = true
AND man_hours = full AND machinery = true

RULE: IF erosion control = man_made_environment
AND man_made_environment_level = total
THEN erosion_rate = controlled

RULE: IF erosion_control = man_made_environment
AND man_made_environment_level = not total
THEN erosion = not controlled AND erosion_rate = increases

Noise

HEURISTIC: IF crop = vines THEN man_made_environment = very_cost_effective
THEN erosion = not controlled

RULE5: IF erosion_control = man_made_environment
AND man_made_environment_level = not total
THEN erosion = controlled

RULE6: IF erosion_control = man_made_environment
AND man_made_environment_level = total
THEN erosion rate = increases

FACT1: critical_factors = {rainfall_cht, soil_cht, topography_cht}
FACT2: degree_of_slope of topography
FACT3: erosion_control = mulching
FACT4: erosion_control = vegetation
FACT5: erosion_control = soil_additive
FACT6: erosion_control = fertiliser
FACT7: erosion_control = natural_organic_mulch
FACT8: erosion_control = man_made_environment
FACT9: erosion_control = fertiliser
FACT10: erosion_control = straw

Heuristic: IF section_of_slope = low THEN erosion_problem = low

RULE11: IF erosion_control = cause_mesh
THEN organic_matter_level = increases
RULE12: IF organic_matter_level = increases
THEN soil_cohesion = increases
RULE13: IF surface_roughness = increase
THEN surface_retention_of_water = increase
RULE14: IF erosion_control = terrace
THEN upkeep_needed = true
RULE15: IF erosion_control = terrace AND upkeep_applied = false
THEN front_wall = not upkept
RULE16: IF erosion_control = terrace AND upkeep_applied = false
THEN pore_pressure = increase
RULE17: IF pore_pressure > limit
THEN wall_collapse = true
RULE18: IF erosion_control = mesh
THEN section_of_slope = not lower
RULE19: IF section_of_slope = lower AND erosion_problem = flood_irrigation
THEN erosion_control = mesh
RULE20: IF ground_cover = high
THEN erosion_rate = low
RULE21: IF ground_cover = natural_vegetation
THEN erosion_rate = low
RULE22: IF erosion_control = terrace AND weather = freak_storm
AND terrace_design = poor
THEN erosion_rate = significant
RULE23: IF erosion_control = soil_additive
THEN ground_cover = increases
RULE24: IF environment = semi_arid
THEN soil_type not = peat
RULE25: IF erosion_control = straw
THEN soil_binding = high
RULE26: IF erosion_control = man_made_environment
AND man_made_environment_level = total
THEN erosion = controlled
RULE27: IF erosion_control = man_made_environment
AND man_made_environment_level = not total
THEN erosion = not controlled
RULE28: IF erosion_control = man_made_environment
AND man_made_environment_level = not total
THEN erosion – increases

HEURISTIC1: IF degree_of_slope < great THEN erosion_control = soil_clt.

HEURISTIC2: IF ground_cover = bare THEN erosion_problem = vulnerable

HEURISTIC3: erosion_control = natural_organic_mulch >

erosion_control = fertiliser

HEURISTIC4: IF crop = vines THEN man_made_environment = very_cost_effective

HEURISTIC5: IF erosion_control = man_made_environment

AND man_made_environment_level = total

THEN terraces = built_up AND drainage = worked_out

AND drainage = all_round AND upkeep_applied = true

AND man_hours = full AND machinery = true

HEURISTIC6: IF erosion_control = terraces THEN upkeep_hours = high

HEURISTIC7: IF section_of_slope = low THEN erosion_problem = low

HEURISTIC8: IF crop = vines THEN man_made_environment = very_cost_effective

HEURISTIC9: IF erosion_control = man_made_environment

AND man_made_environment_level = total

THEN terraces = built_up AND drainage = worked_out

AND drainage = all_round AND upkeep_applied = true

AND man_hours = full AND machinery = true

EXAMPLE1: IF crop = Olive_trees AND ground_cover = low

THEN erosion_problem = high
Appendix C.5
Conditions.

1. Veg cover < 30%
2. Grass verge needed
3. Soil crust break up
4. Animal erosion problem
5. Large animals
6. < critical value

Results.

1. Preventative measures needed.
2. Net vegetation cover.
3. Flow convergence.
4. Prevent rain impact.
5. Erosion.
6. No erosion.

List

1. Mulching cover...

Actions.

1. Factors: Angle, length, volume, slope.
2. Reduce overgrazing.

Tests

1. Slope angle, vegetation level, vegetation type.
Appendix C.6
The domain specification.

```
Soil

erosion_control: {mulching, cause_meshing, vegetation_cover, soil_additive,
                   fertiliser, natural_organic_mulch, straw, terrace,
                   man_made_environment, soil_clt, ground_cover, irrigation}

organic_matter_level, soil_cohesion: increases

soil_binding, rainfall_erosion: {low, high}

surface_roughness: increases

degree_of_slope: great

section_of_slope: lower

erosion_problem: {flood_irrigation, controlled, variable}

ground_cover: {high, natural, vegetation, increases}

weather: freak_storm

terrace_design: poor

erosion_rate: {low, significant, increases}

environment: semi-arid

soil_type: peat

man_made_environment_level: total

upkeep_needed, upkeep_applied, wall_collapse: Bool

frontwall: upkept

pore_pressure: increase, limit

erosion_control_needed, verge_needed, soil_crust_break_up, flow_convergence: {yes, no}

general_erosion_capacity: N x N x N

frequent_movement_of_animals: {needed}

animals: {large, small}

f1, f2: N x N x N -> N

slope_angle, slope_length, water_volume, veg_level, veg_type, critical_value,
animal_erosion_capacity, veg_cover: N

animals: {cattle, sheep, goats}
```
mulching control

soil

erosion_control = mulching

cause mesh control

soil

erosion_control = cause_mesh

vegetation control

soil

erosion_control = vegetation_cover

additive control

soil

erosion_control = soil_additive

fertilise control

soil

erosion_control = fertiliser

natural control

soil

erosion_control = natural_organic_mulch

straw control

soil

erosion_control = straw
terrace control

soil

erosion control = terrace

rule one

soil

cause mesh control => organic matter increases

organic matter increases

soil

organic matter level = increases

rule two

soil

cause matter increases => soil cohesion = increases

modus ponem gives us (rule one => rule two)

rule three

soil

surface roughness => increases = surface retention of water = increases

rule four

soil

terrace control => upkeep needed

upkeep needed

soil

upkeep = true
upkept

| soil
| ___
|_
| upkeep applied = true

rule five

| soil
| ___
|_
| unkept terrace => front wall not upkept

rule six

| soil
| ___
|_
| unkept terrace => pore pressure = increase

unkept terrace

| soil
| ___
|_
| terrace control
| not upkept

rule seven

| soil
| ___
|_
| pore pressure > limit => wall collapse = true

rule eight

| soil
| ___
|_
| cause mesh control => section of slope not = lower

rule nine

| soil
| ___
|_
| section of slope = lower * erosion problem = flood irrigation
| => cause mesh control
rule ten

soil

ground cover = high => erosion rate = low

rule eleven

soil

ground cover = natural vegetation => erosion rate = low

rule twelve

soil

poor terrace conditions => significant erosion factor

significant erosion factor

soil

erosion rate = significant

poor terrace conditions

soil

terrace control
weather = freak storm
terrace design = poor

rule thirteen

soil

additive control => ground cover = increases

rule fourteen

soil

environment = semi arid => soil type = not peat
rule fifteen
soil
straw control => soil binding = high

rule sixteen
soil
good man made environment => controlled erosion

good man made environment
soil
erosion control = man made environment
man made environment level = total

poor man made environment
soil
erosion control = man made environment
man made environment level = not total

controlled erosion
soil
erosion problem = controlled

rule seventeen
soil
poor man made environment => not controlled erosion

rule eighteen
soil
poor man made environment => increase erosion rate
increase erosion rate

soil

erosion rate = increase

heuristic one

soil

degree of slope < great => erosion control = soil cht.

heuristic two

soil

ground cover = bare => erosion problem = vulnerable

heuristic three

soil

natural control > fertiliser control

heuristic four

crop = vines => man made environment = very cost effective

heuristic five

soil

good man made environment => good terrace conditions
good terrace conditions

- soil

- terraces = built up
- drainage = worked out
- drainage = all round
- upkeep applied = true
- man hours = full
- machinery = true

example one

- soil

- crop = olive trees * ground cover = low => erosion problem = high

cover control

- soil

- erosion_control = ground_cover

irrigation control

- soil

- erosion_control = irrigation

barrier control

- soil

- erosion_control = barrier

animal control

- soil

- erosion_control = animal_husbandary
preventative measure
|
| mulching_control
| cover_control
| terrace_control
| fertiliser_control
| irrigation_control
| barrier_control
| animal_control
|

erosion problem
|
| soil
| ____________
| erosion_control-needed = yes
|

bellow critical veg
|
| soil
| ________________
| veg_cover < 30% => erosion_problem
|

grass verge
|
| soil
| ________________
| verge_needed = yes
|

grass verge factors
| slope_angle, slope_length, water_volume
| erosion_capacity = f1(slope_angle, slope_length, water_volume)
|

crust break ok
|
| soil
| ________________
| soil_crust_break_up = yes
flow convergence ok

| soil
|
| flow_convergence = yes

flow problem

| soil
|  
| flow_convergence => erosion_problem

rain_impact_ok

| soil
|  
| not crust_break_ok => impact_resistance

impact resistance

| soil
|  
| rainfall_erosion = low

overgrazing problem

| soil
|  
| overgrazing => erosion_problem

overgrazing

| animal_erosion_critical
|  
| reduce overgrazing

| soil
|  
| overgrazing => frequent_movement_of_animals = needed
Large_animal

soil

animals = large

animal erosion

soil

large_animals => erosion_problem

animal_erosion_factors

soil

animal_erosion_capacity = f2(slope_angle, veg_level, veg_type)

animal erosion critical

soil

animal_erosion_factors > critical_value => erosion_problem
Appendix C.7
The representation specification.

RULE: <rule-id>: IF <expression> { AND <expression> } THEN <action>

HEURISTIC: <heuristic-id>: IF <expression> { AND <expression> } THEN <action>

EXAMPLE: <example-id>: IF <expression> { AND <expression> } THEN <action>

FACT: <fact-id> : <expression> { AND <expression> }

<rule-id>::= <simple-string>

<heuristic-id>::= <simple-string>

<simple-string>::= <name>kchar-seq><letter>knamexdiagit

<name>::= <lowercase>kname><letter>kname><diagit>

<letter>::= <lowercase>A..Z

<lowercase>::= a..z

<digit>::= 0..9

<char-seq>::= <letter>kchar-seq><letter>

<expression>::= <simple-expression>ksimple-expression><relational-operator>

<simple-expression>::= <term>

<term>::= <identifier>

<identifier>::= <letter> {<letter>kdigit>}

<relational-operator>::= <> | = | <= | >= | l < l > | l <= l => l <boolean>

<boolean>::= "true" | "false"

?action>::= <expression>
Appendix  C.8
The concrete specification.

RULE: rule one IF cause mesh control THEN organic matter increases

RULE: rule two IF cause matter increases THEN soil cohesion = increases

RULE: rule three IF surface roughness = increases
    THEN surface retention of water = increases

RULE: rule four IF terrace control THEN unkeep needed

RULE: rule five IF unkept terrace THEN front wall not upkept

RULE: rule six IF unkept terrace THEN pore pressure = increase

RULE: rule seven IF pore pressure > limit THEN wall collapse = true

RULE: rule eight IF cause mesh control THEN section of slope not = lower

RULE: rule nine IF section of slope = lower
    AND erosion problem = flood irrigation
    THEN cause mesh control

RULE: rule ten IF ground cover = high THEN erosion rate = low

RULE: rule eleven IF ground cover = natural vegetation THEN erosion rate = low

RULE: rule twelve IF poor terrace conditions THEN significant erosion factor

RULE: rule thirteen IF additive control THEN ground cover = increases

RULE: rule fourteen IF environment = semi arid THEN soil type = not peat

RULE: rule fifteen IF straw control THEN soil binding = high

RULE: rule sixteen IF good man made environment THEN controlled erosion

RULE: rule seventeen IF poor man made environment THEN not controlled erosion

RULE: rule eighteen IF poor man made environment THEN increase erosion rate

HEURISTIC: heuristic one IF degree of slope < great
    THEN erosion control = soil cht.

HEURISTIC: heuristic two IF ground cover = bare
    THEN erosion problem = vulnerable

HEURISTIC: heuristic three IF natural control THEN fertiliser control

HEURISTIC: heuristic four IF crop = vines
    THEN man made environment = very cost effective

HEURISTIC: heuristic five IF good man made environment
    THEN good terrace conditions

EXAMPLE: example one IF crop = olive trees AND ground cover = low
    THEN erosion problem = high
FACT: mulching control: erosion_control = mulching

FACT: cause mesh control: erosion_control = cause_mesh

FACT: vegetation control: erosion_control = vegetation_cover

FACT: additive control: erosion_control = soil_additive

FACT: fertilise control: erosion_control = fertiliser

FACT: natural control: erosion_control = natural_organic_mulch

FACT: straw control: erosion_control = straw

FACT: terrace control: erosion_control = terrace

FACT: organic matter increase: organic matter level = increases

FACT: upkeep_needed: upkeep = true

FACT: upkept: upkeep applied = true

FACT: unkept terrace: erosion control = terrace AND upkeep = false

FACT: significant erosion factor: erosion rate = significant

FACT: poor terrace conditions: erosion control = terrace
  AND weather = freak storm AND terrace design = poor

FACT: good man made environment: erosion control = man made environment
  AND man made environment level = total

FACT: poor man made environment: erosion control = man made environment
  AND man made environment level = not total

FACT: controlled erosion: erosion problem = controlled

FACT: increase erosion rate: erosion rate = increase

FACT: good terrace conditions: terraces = built up AND drainage = worked out
  AND drainage = all round AND upkeep applied = true AND man hours = full
  AND machinery = true