Rigorous approach to the development of knowledge-based systems

Robert T Plant

Knowledge-based systems have often been criticized for the limited theoretical base on which they are constructed. The partially valid view, held by many, is that systems are often constructed in an ad hoc, *individual way, which leads to unmaintainable, unreliable, and unrigorous systems. This holds even though there have been several attempts at producing development methodologies to assist the knowledge engineer in the construction process*[1–7]*. A large contributing reason for the limited applicability of these methodologies is that they often too closely follow the waterfall model approach used for the development of conventional software systems*[8]*. This approach forces developers to make large jumps in the system state during development, which is not necessarily the most conducive way to model the domain accurately.*

The paper therefore aims to introduce a new alternative approach that shows the benefits of taking a software engineering philosophy towards the development of knowledge-based systems. The methodology breaks down the process of creating a knowledge-based system into constituent parts and discusses ways of creating rigorous specifications for those parts, as applicable. This includes specifying the knowledge base, the representation, the control architecture, etc., thus promoting quality systems that are better specified, more reliable, and easier to maintain.

Keywords: knowledge-based systems, system development, specification

The very nature of the problems that knowledge-based systems typically attempt to solve (e.g., complex yet poorly structured problems within a large domain) prevents in all but trivial cases a total formal specification from being constructed (i.e., the full specification of the problem for all cases). However, it is possible to

Department of Computer Information Systems, University of Miami, Coral Gables, Florida, USA

Paper received 22 June 1990. Revised paper received 14 August 1991

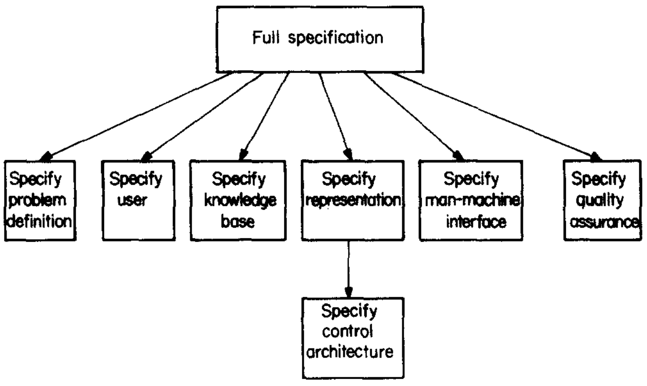
Full specification
Specify problem definition
Specify user
Specify knowledge base
Specify representation
Specify man-machine interface
Specify quality assurance
Specify control architecture

Figure 1. Composite specification

produce a partial, composite specification for a system, in that it is possible to specify several key points in the development process and link these together in a rigorous manner, such as that indicated by Jones, who advocates developing 'a program in a style which separates decisions, making the subsequent tasks more manageable and, therefore more accurate'[9].

A complete system specification would contain seven aspects, as illustrated in Figure 1. The first is the specification of the problem definition, the production of which is extremely difficult for nontrivial knowledge-based systems due to their inherent lack of procedural, deterministic, algorithmic structure.

The second area where specification is needed is that of the intended user. This can be performed through the creation of a behavioural model.

The third part is the specification of the knowledge base. It is possible to model this aspect, as the knowledge elicited from the domain expert/knowledge source is finite, and through the use of transformational processes this can be specified formally. The specification of the knowledge base is vital if the system is to be maintained, while the representational independence of the specification promotes clarity and flexibility.

Fourth, it is vital that a suitable representation is selected, and this is done by analysing the representational needs of the knowledge base. Once selected, the syntax and semantics of the representation can be specified. Having specified the representation it is then possible to select an appropriate control architecture, the operation of which can also be specified.

The fifth aspect that needs to be specified is the man–machine interface. This can be fully specified through the use of formal techniques, and several examples of such specifications have been documented[10,11]. The specification of the interface allows the knowledge engineer and user to have an unambiguous frame of reference, through which interactions with the system can be viewed.

The sixth component of the composite specification is the need to specify the validation and verification requirements. The definition of these will enable the knowledge engineer to be able to judge whether the levels of quality reached are adequate for system use.

It is possible therefore formally to specify several aspects of an expert system, each of which are fundamental to its construction, and this paper shows how, from an initial specification of the problem definition, these points can be rigorously reached and combined together to form a concrete specification from which the system can be implemented.

OVERVIEW

The methodology as a whole can be introduced by considering Figure 2. The development commences with an initial specification, which acts as an informal software requirements document. This gives a broad outline of the systems parameters and boundaries, to be used by the knowledge engineer as the basis of both the knowledge elicitation phase and the creation of the user model.

In the knowledge elicitation phase the most suitable knowledge elicitation technique is selected with which to extract knowledge from the domain expert. The knowledge engineer then uses this extracted knowledge as the basis of the elicited representation, an unprocessed representation that usually has a textual form. The elicited representation, however, is too coarse in nature to act as the specification for an implementation and so it is necessary for the representation to undergo a refinement process. The result of this is a more adequate representation, termed the primary representation. It is adequate in the sense that an adequate level of completeness and consistency has been reached, to allow major knowledge processing of the representation to be performed. The first process is to transform the primary representation into a formal representation, this being a mathematical specification written in the Z specification language[12,13].

The second process is an analysis that examines what constituent characteristics are present in the primary representation, before attempting to match these with the characteristics of the 'classical' representations, such as frames, production systems, and semantic networks. From this matching process a specification of a suitable representation language can be produced. This is known as the representation specification. Following this, the domain and representation specifications are drawn together to form the secondary representation in which the domain knowledge from

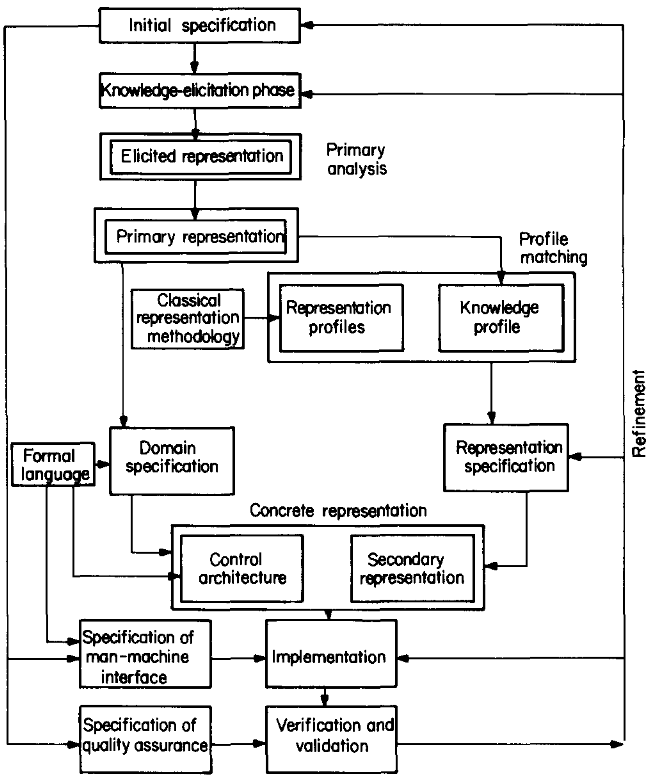

Figure 2. Rigorous development methodology

the domain specification is represented into the form advocated by the representation specification; this plus the specification of the control architecture forms the concrete specification. This acts as a specification for the implementation of the knowledge base, which when combined with the man–machine interface specification (which allows the human–computer interaction considerations to be understood) provides the basis for implementing the whole system.

The remainder of the paper examines these stages in greater detail, illustrating them through an actual case study in which the problems associated with preventing soil erosion are examined. The case study involved a domain expert, two assistant domain experts, and two knowledge engineers.

INITIAL SPECIFICATION

An expert system can be viewed as a type of computer system, and as such the development constraints that software engineers place on conventional software systems can also be applied to the expert system life-cycle. The first stage in this life-cycle is often termed by software engineers the 'requirements analysis and definition'[14], which is here termed the 'initial specification'.

The aim of the initial specification is to establish boundaries on the solution space of the problem. This is important in a knowledge-based software system, for the domains of such systems lack much of the definition associated with their more traditional counterparts. This lack of definition does not mean that producing an initial specification is impossible, but merely that more effort is required, with perhaps a change of emphasis. A series of steps is proposed that can be followed to assist the knowledge engineer to arrive at an outline for the 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 1: 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:

- the proposed project area
- the reasoning behind the need for a system
- references that introduce literature on the projected area

In the case study the area outline was achieved through an exchange of ideas between the knowledge engineers and the project instigators. This resulted in the area of focus being determined as 'soil erosion prevention', with introductory material being given to the knowledge engineering team[15-19].

Step 2: knowledge engineer surveys area

The knowledge engineer takes the system proposal from step 1 and by following up the literature references surveys the area. The survey is aimed to produce two results. First, the knowledge engineer can gain further insight into the problem area and, second, it provides initial views towards feasibility. The first of these results is a necessary prerequisite of the second. This stage is primarily a familiarization one, which in the case study was performed by following up the literature survey by a 'briefing' with the assistant domain experts. This allowed the knowledge engineers to raise and clarify points arising from their research, as well as allowing the domain specialists to ask questions about the approach likely to be taken by the knowledge engineers, in addition to general questions on knowledge-based systems and artificial intelligence. This dialogue is useful in building a mutual trust and understanding between parties. The domain expert was not initially involved due to availability and being a scarce resource.

Step 3: prepare initial specification

The knowledge engineer, having assessed the system feasibility, now tries by discussion with both the initiator of the project and, if possible, the domain expert to focus on 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 two-fold. First, to clarify the knowledge engineer's understanding of the domain, so that the knowledge engineer can appreciate the project put forward by the domain expert. Second, the domain expert gains insight into 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 4: clarification

The initial specification produced in step 3 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 that defines as clearly as possible the area under investigation and especially the boundaries to that area and defines what problem types within the area are applicable.

To give a full initial specification is beyond the scope of this paper, but the following can act as the basis for the case study and subsequent discussion:

> 'The detailed examination of soil erosion that occurs in semi-arid environments on field-sized plots where no wind erosion occurs and the preventative measures available to combat such erosion.'

Having produced an initial specification, this can then act as a baseline document that the knowledge engineer can refer back to during any part of the development. It will also be useful in the post-development stages, for instance, in maintenance.

KNOWLEDGE ELICITATION PHASE

Definition

The creation of an expert or knowledge-based system is not by definition possible without knowledge, and it is the extraction, gathering, and articulation of that knowledge by the knowledge engineer from the domain expert in a particular area of interest which is termed knowledge elicitation. This is different to knowledge analysis, in which the knowledge engineer considers the content and inter-relationship of the information provided by the domain expert gathered during the elicitation phase. The knowledge analysis process shall be examined later in the paper.

The two, however, are not 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 for the interview to be discursive and continuous and to have a dialogue rather than be a set of disjointed questions and answers.

Elicitation techniques

There are several different approaches that the knowledge engineer can take to the problem of knowledge elicitation[20-23], for example:

- verbal transfer of knowledge, e.g., interviewing – structured, focused, and unstructured[24]
- reporting techniques, e.g., online, offline, and hybrid[24]
- psychological techniques, e.g., repertory grid[25], inference structure[26], goal decomposition[2], and distinguishing evidence[27]
- knowledge engineer investigates literature

One of the aims of the methodology is that the knowledge elicitation phase should result in a textual form of elicited representation, from which a refinement process can proceed. This allows a permanent record of the knowledge to be kept in the form in which it was elicited from the domain expert. Knowledge engineers can then assess if the refined knowledge used in the knowledge base is semantically equivalent to its original form. Further, if the system has to be maintained at any time then this document is available. (As shall be seen later, the knowledge can be represented in a formal language, and this can be used advantageously for maintenance purposes.)

During the elicitation phase, a variety of techniques can be used to elicit knowledge at many levels and of a variety of types. For example, the elicitation may commence with the knowledge engineer performing a series of unstructured interviews to extract high-level conceptual knowledge. This may then be followed by structured interviews where the relationship of the domain, its structure, and more detailed information are obtained. This may then be followed by a series of focused interviews to fill in the low-level information of a fine grain size.

The methodology described is designed to allow the knowledge engineer to update the system incrementally, in that all the knowledge in the system goes through the same elicitation, acquisition, and formalization processes, with the specifications being updated. This allows for the new knowledge to be correctly integrated into the specifications, where this may mean that the specifications are added to, modified, or have knowledge items removed from them.

Application of knowledge elicitation

In the soil erosion 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 to the old.
- Formulate new questions.
- Check for consistency and completeness.

As a result, one of the knowledge engineers could act as the 'principal knowledge engineer', leading the questioning and discussion, while the 'secondary knowledge engineer' checked the information provided for completeness and consistency. The secondary knowledge engineer would summarize and raise points that were more general, attempting to place the information in context.

The elicitation process itself was unusual in that several domain experts were used. 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, while the other domain experts could examine these answers, suggesting where points had been missed out as well as summarizing and assisting the knowledge engineers on points they found difficult, by rephrasing the information.

Elicited representation

As stated, the result of the knowledge elicitation phase is the elicited representation, the form of which is intended to be textually based. For example, a transcript of an interview between a domain expert and a knowledge engineer could act as an elicited representation.

In the case of the project the interviews were transcribed and a vast amount of text was produced, averaging 10 pages of text an hour or approximately 5000 words an hour.

KNOWLEDGE ANALYSIS PHASE

The next stage in the methodology is that of knowledge analysis: the process of breaking down the knowledge extracted during the knowledge elicitation phase into its composite parts and examining the relationship between these parts.

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 on it. One of these ultimately produces a formal specification of the domain and another acts as the basis for the selection of the high-level 'classical' representation, e.g., a production system, which ultimately will be used to represent the domain knowledge held in the formal specification.

The reason that the elicited representation is not used directly as the basis of these analyses is that the elicited representation is in the form of English text, which is far too ambiguous in nature. Also the text may be noisy, suffer from problems of continuity or high modularity, or have poor linkage between areas of knowledge – problem areas that are not always apparent in unanalysed English text.

In the soil erosion project a semi-structured elicitation session resulted in an elicited representation of the form:

> **Knowledge engineer:** What methods exist for preventing soil erosion?
> **Domain expert:** 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.

As stated, the transcripts tend to be extremely long and verbose. It is therefore imperative to use an elicitation technique that maximizes the amount of knowledge and information extracted and minimizes digression.

Primary representations

As stated, the aim of this phase is to produce from the elicited knowledge a more rigorous, primary representation. The characteristics of a primary representation are that it has a simple syntax and semantics and is weakly typed. The reasoning behind this is to make the best compromise between rigour and flexibility. Rigour is used in the sense that the language/form used for the primary representation should not encourage ambiguity or highlight any lack of consistency or completeness, yet be flexible enough to mould the primary representation around a variety of elicited representations.

The aim of having only a simple syntax and semantics is that due to their simplicity they will cover a large range of textual situations, while because they have some formal basis the text of the elicited representation can be transformed into a more rigorous form with which to reason about the domain. The weak typing is also an attempt to capture and categorize as much text as possible and instill some formality on it. The elicited representations purposely do not have a complex syntax or semantics or strong data types. This is because the philosophy of the approach is one of representation refinement where only small changes are made at each stage, enabling justifications to be made, which helps prevent information loss or change of semantic meaning. It is not possible to adhere to this philosophy if large jumps in the representation are made.

Further characteristics of the primary representation are that it be amenable to the transformation of large quantities of textual information, yet that it possesses little or no basis for inference. The reasoning behind the first of these considerations is due to the sizeable transcripts and texts that are obtained during the knowledge elicitation phase. The reasoning behind the need for little inference capacity in the primary representation is that the representation will not be required to act as a basis for inference, and the inclusion of these facilities would only add to the complexity of the language or form, Features that prevent information loss or change are to be encouraged, as translations between representations can sometimes influence a change in semantic meaning.

Having compiled this set of characteristics describing the form a primary representation should take, several representations which adhered to them can be used, two of which are the flow diagram and the decision table.

Flow diagram

The flow diagram is based on the flowchart idea, but instead of detailing the flow of control for a program, an attempt is made to try to establish the knowledge flow contained within the elicited representation.

Decision table

The decision table is used generally for elicited representations which have a coarser grain size of textual information than that used by the flow diagram. Decision tables allow for the domain knowledge to be easily checked for consistency and completeness and can be easily reduced in a formalized manner[28,29].

Transformational characteristics

To select an appropriate primary representation into which the elicited knowledge could be transformed, the characteristics of that knowledge have to be matched against those of the representation to ensure that the data types, knowledge types, and control characteristics are adequate. One set of characteristics that could be used to indicate a suitable primary representation is the following:

- noise
- modularity of knowledge
- linkage of knowledge
- operational types
- sequencing
- justifications
- explanations

The characteristics above are sequenced specifically, commencing with what could be termed low-level characteristics and progressing to higher-level ones. The lowest level of all the characteristics is noise, which is defined as 'words or segments of text within the elicited representation that are undefined or unrecognisable', and which has to be removed before analysis.

The second two characteristics, modularity and linkage, are highly inter-related. However, before describing this inter-relationship, definitions of each are needed. Modularity can be defined as: 'A module of knowledge can be described as a collection of knowledge statements with the following characteristics: (i) the knowledge statements are related in subject, this being a subset of the domain, and (ii) the knowledge statements are in close physical proximity to each other on the transcript or other form of elicited representation.' In relation to the primary representation, modularity has to have a slightly different definition: 'Modularity is the ability of the primary representation to allow statements that are conceptually related and in close physical proximity to each other within the elicited representation, to keep this relationship.'

Linkage can broadly be defined as 'the relationships among modules', and is a measure of their independence.

Now the role of modularity and linkage within the elicited representation can be discussed. Elicited representations that have highly modular knowledge are easier to represent in any of the primary representations than non-modular ones, for the knowledge is focused and the context of that knowledge is then clearer for the knowledge engineer to understand.

The linkage of knowledge is important with respect to justifications and explanations. If the elicited representation is highly modular and has high linkage then this makes explanations and justifications easier to accomplish. If linkage, modularity, or sequencing are low, however, then the generation of implicit explanations and justifications is harder to achieve.

The operational type characteristic is based on the different types of operations that are performed within the elicited representation, for example, condition, action, diagnosis, test, etc. The more operational types there are the more complex the primary representation will need to be.

Sequencing refers to the relationship that individual items of knowledge have with each other. The sequencing of information and knowledge within an elicited representation can also be regarded as a complexity indicator for that representation.

The justifications and explanations characteristics present within an elicited representation may be either explicit or implicit. Justifications and explanations included in the text by the domain expert are considered explicit.

Thus each of these characteristics must be taken into consideration before deciding on a primary representation. This is because each of the primary representations has a different capability to accommodate different characteristics. For example, the flow diagram cannot represent any elicited knowledge that has a low linkage of knowledge because this goes against one of the basic principles that the representation is founded on. Decision tables, on the other hand, do not require the information to be structured nor do they require the knowledge to be linked together, although this can be accommodated along with the ability to modularize the knowledge.

To aid the decision of which primary representation to select for a given elicited representation, a graphical selection system has been created in which two types of diagram are used: bandwidth diagrams and trace diagrams.

Traces are diagrams that outline the amount of each characteristic present within the elicited representation, while bandwidth diagrams show what range of values a primary representation is most suitable for representing. The aim is to select the primary representation that best encloses (matches) a trace. For example, Figure 3 shows the characteristics that are associated with the soil erosion elicited representation. It can be seen that the domain would be better represented in a flow diagram (see Figure 4) than in a decision table (see Figure 5). This follows as the closer the fit of the bandwidth, the more suitable the constructs in the primary representation will be for representing the needs of the elicited information.

Summary of knowledge analysis phase

Figure 6 summarizes the process of selecting a primary representation:

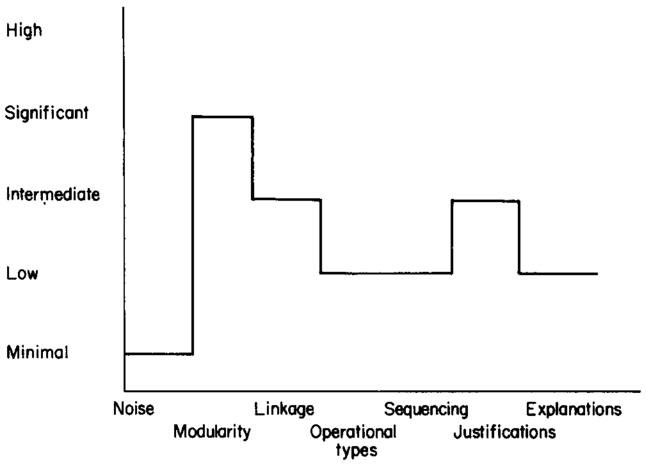

Figure 3. Trace for soil erosion characteristics

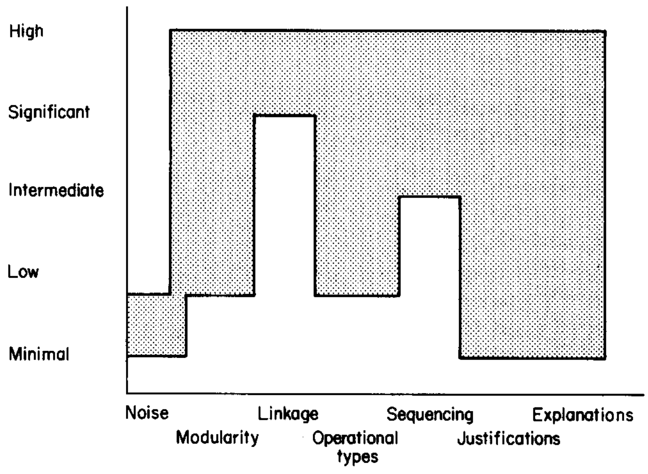

Figure 4. Bandwidth for flow diagram

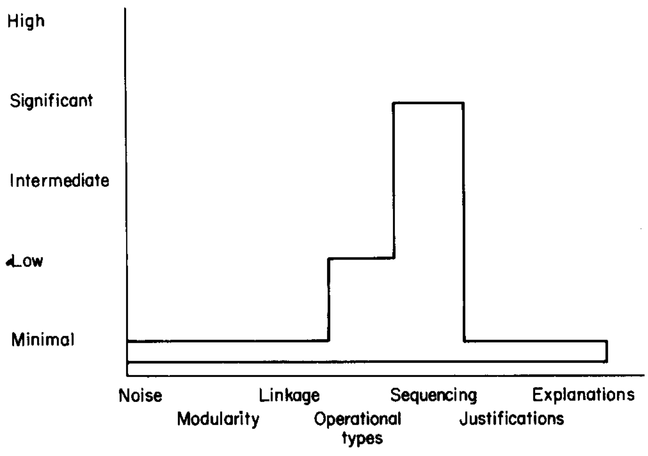

Figure 5. Bandwidth for decision table

(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.

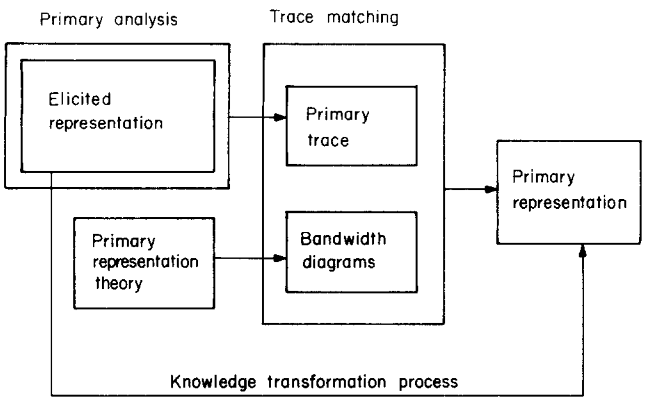

Figure 6. Knowledge analysis process

(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.

CREATION AND USE OF PRIMARY REPRESENTATION

Following the selection of the primary representation, which for the case study is to be a flow diagram as illustrated in Figure 7, the knowledge engineer then has to perform a series of transformations on the elicited knowledge.

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. 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 for analysis techniques that indicate the most suitable classical representation in which to carry forward the design of the system. This is known as the 'representation specification'. The process involved in creating these two specifications will now be examined.

DOMAIN SPECIFICATIONS

The primary representation provides 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 cannot be spotted due to the structures used. It is the aim of the domain specification to help reduce these problems.

The domain specification therefore will have to have mathematics as its basis, and this led to the adoption of the Z notation, a formal specification language developed at the Programming Research Group, Oxford, UK[12–13,30,31].

Specifications in Z consist of formal text and natural language text. The former provides a precise specification, while the latter introduces and explains the formal

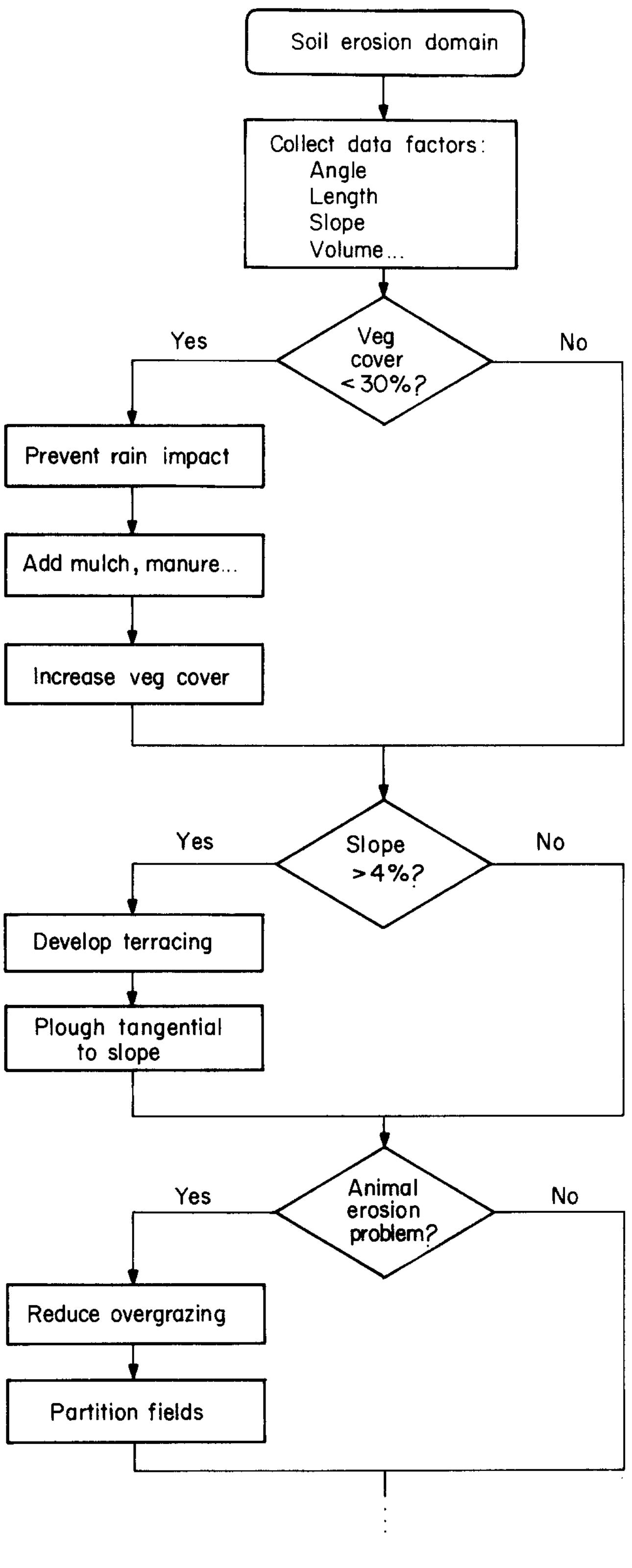

Figure 7. Flow diagram of soil erosion process

parts. Specifications are developed via small pieces of mathematics that are built up by using the schema language to allow specifications to be structured. This leads to formal specifications that are more readable than a specification presented in mathematics alone.

The domain specification plays several roles. First, it acts as a rigorous specification of the elicited knowledge. Second, the specification can act as a basis for

```
Soil = erosion__control: {mulching, Cause meshing,
vegetation__cover,
fertiliser, irrigation, soil__cht, man__made__environment};
soil__binding, rainfall__erosion: {high,low},
degree__of__slopeN; erosion__rate: {Low, Significant, High}
vegetation__control = Soil | erosion__control = cause__mesh
additive__control = Soil | soil__additive
rule__one = soil | cause__mesh__control ⇒ organic__matter__
increases
rule__two = soil | cause__matter__increases ⇒ soil__cohesion =
increases
rule__three = soil | surface__roughness ⇒ surface__retention =
increases
rule__four = soil | terrace__control ⇒ maintenance = yes
rule__five = soil | pore__pressure> limit ⇒ wall__collapse = true
rule__six = soil | (section__of__slope = lower)
     & (erosion__type = flood__irrigation) ⇒ cause__mesh__control
rule seven = soil | ground__cover = natural__vegetation ⇒
erosion__rate = low
rule eight = soil | environment = semi__arid ⇒ soil__type= not
peat
```

Figure 8. Domain specification

the future maintenance of the knowledge base. The role of correct maintenance is important and the procedures for updating the knowledge base can themselves be formally specified. Third, the Z specification can act as a medium for communication between the knowledge engineer and the domain expert as well as between the knowledge engineer and the implementor of the system.

Thus the use of a formal language in the development of a knowledge base is advantageous. Figure 8 illustrates a section of the domain specification for the case study.

TOWARDS 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, where the classical representations are 'frames', 'production systems', 'semantic networks', etc.

To find the most suitable form, several considerations have to be taken into account. First, the underlying needs of the system, in terms of data and knowledge types, have to be assessed. This assessment is based on the degree of presence of the five underlying knowledge types found to some extent in all systems: factual, heuristic, control, procedural, and conceptual.

The second step is then to assess which of the representations, e.g., frames, rules, etc., has a structure that best accommodates the knowledge types and control needs of the knowledge to be represented.

These two steps are performed in a similar way to the selection of the primary representation, in that a graphical system is used to assist the knowledge engineer to determine the best representational match. Three steps are performed. First, a knowledge profile (histogram) for the elicited knowledge contained within the primary representation is created. This details the amount of each knowledge type present in the primary representation. This is known as profile analysis. Second, the domain's knowledge profile (created in stage one) is compared with profiles for the classical representations. The profiles of the classical representations attempt to show the ability of these forms (a standardized version) to capture and represent these knowledge types. This is the profile-matching process. Third, the best match is selected around which to base the representation specification. These stages will now be described in more detail.

Profile analysis

First, profile analysis is defined: 'Profile analysis is the process of producing a knowledge profile for the elicited knowledge contained within the primary representation, where a knowledge profile is a histogram that indicates the amounts of certain knowledge types within the domain.'

Graph structure

The graph structure around which the profiles are constructed is given in Figure 9. The knowledge engineer plots the amount of each knowledge type present within the primary knowledge representation. The vertical axis is not numeric as giving numeric values to the amount of a knowledge type in a domain is felt to be unrealistic. This process is far more subjective, hence the vertical scale. The horizontal axis is based on a set of knowledge types that compose the majority of knowledge within any given elicited representation.

Knowledge profile

By considering definitions for the knowledge types, an assessment of the degree to which these types are present within the primary representation can be made from which a profile can be created. For example, the case study has the form given in Figure 10. This is the profile of a primary representation that contained high amounts of factual knowledge, significant amounts of procedural knowledge, intermediate amounts of control knowledge, but only minimal amounts of heuristic and conceptual knowledge.

Profile matching

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 first having a standard representation of a given classical form, such as frames, and then examining it for the ability to represent the five knowledge types. From these analyses, profiles are drawn and matched against the needs of the knowledge profile of the domain under consideration; the representation having a profile closest to that of the domain is selected.

The selection itself is made by taking the domain and representation profiles, overlaying them, and looking for the differences. For example, if the representation profile in Figure 11 was under consideration and overlaid with the domain profile from Figure 1, this would give Figure 12.

Figure 12 shows that the representation is deficient (negative deviance) in the representation of control knowledge and that when the representation specification is constructed this should be taken into account. The representation has more representational power in

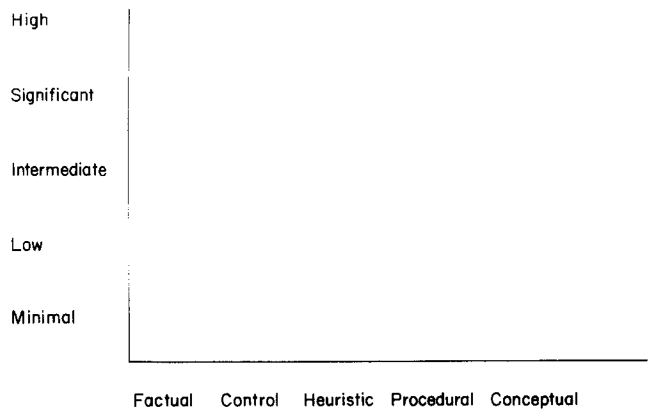

Figure 9. Graphical structure of profile

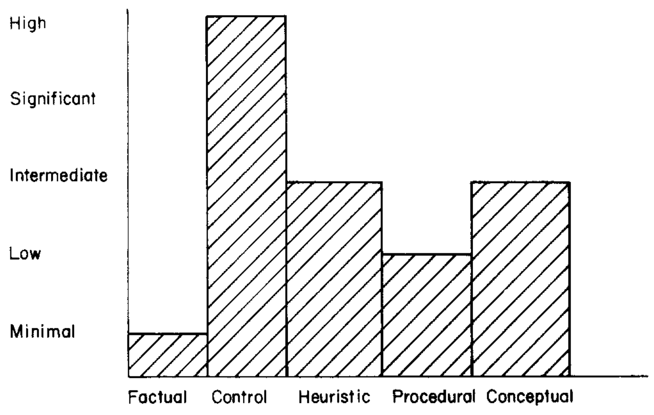

Figure 11. Representation profile of production system

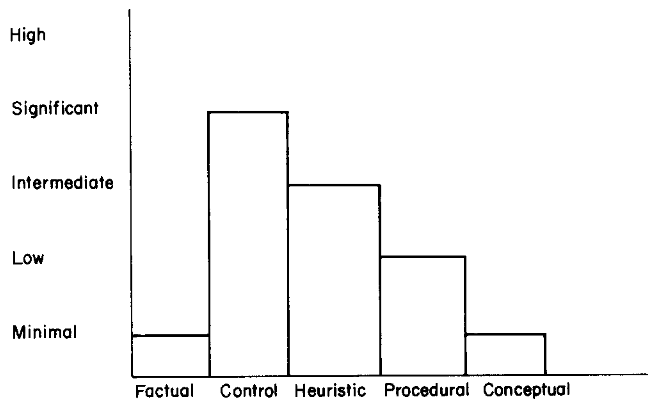

Figure 10. Knowledge profile

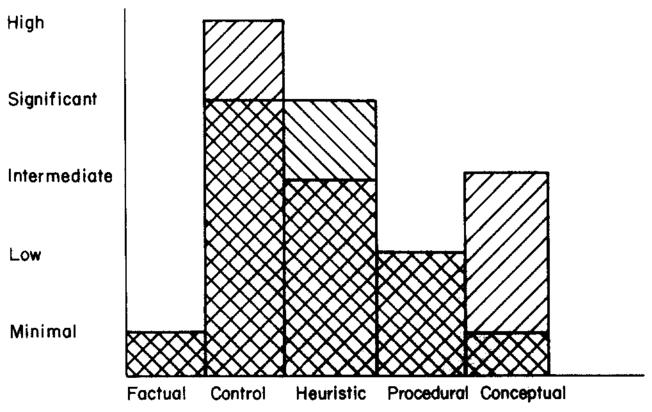

Figure 12. Profile matching

representing procedural knowledge (positive deviance) than is needed in this particular case.

Representation specification

Having performed the matching process, the knowledge engineer can assess the results to produce a formal syntax and a denotational semantics for the representation suggested in the matching process.

The form advocated by the profile matching indicated that a rule-based system would be the most suitable match for the soil erosion domain. A section of the representation specification for a rule-based formalism is given in Figure 13.

CONCRETE SPECIFICATION

Having created the domain and representation specifications, these two specifications can be combined into a form that will allow a move towards implementation. This stage is known as the concrete specification.

The concrete specification itself has two parts: a secondary representation in which the domain knowledge is transformed from its Z specification into the form advocated by the representation specification, and a formal specification of the control architecture associated with the representation; this is also written in Z.

This is not an implementation as the representation is a cross between a high-level version of what is to be implemented (i.e., minus the syntactic sugar of the implementation language) and a formal specification in the style suggested by the syntax and semantics of the representation specification.

Thus the domain specification for the soil erosion case study can be re-represented as rules in the syntax and semantics as formalized in the representation spec-

```
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>|<char_seq>
<name>::= <lowercase>|<name><letter>|<name><digit>
<letter>::<lowercase>|A..Z
<lowercase>::= a..z
<digit>::= 0..9
<char_seq>::= <letter><char_seq><letter>
<expression>::= <simple_expression> |
        <simple_expression> <relational_operator> <simple_
        expression>
<simple_expression>::=<term>
<term>::= <identifier>
<identifier>::= <letter> {<letter>|<digit>}
<relational_operator>::= <> | = | < | > | ⇐ | ⇒ | <boolean>
<boolean>::= "true"| "false"
<action>::= <expression>
```

Figure 13. Representation 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 sùrface retention of water = increases
RULE: rule four IF ground cover = high THEN erosion rate = low
RULE: rule five IF environment = semi-arid THEN soil type = not peat
HEURISTIC: heuristic one IF degree of slope < 4 degrees THEN erosion = low
HEURISTIC: heuristic two IF crop = vines THEN erosion prevention = very cost effective
EXAMPLE: Example one IF crop = oil trees AND ground cover = low
    THEN erosion problem = high
FACT: poor environment: erosion control = man made terrace AND Terrace slope > 4 degrees

Figure 14. Concrete specification

ification. Doing so obtains a concrete representation of the form shown in Figure 14.

IMPLEMENTATION

Having created the concrete specification, this is then used as the basis of the knowledge-base implementation. The interface issues are resolved by referral to the man–machine interface specification.

The implementation of the system should be the most straightforward of all the stages, due to the high degree of structuring and refinement that has been performed on the domain knowledge.

The mechanism through which the system is implemented is left open to the knowledge engineer as this is considered a straightforward step once the specifications have been developed.

VALIDATION AND VERIFICATION

The final specification that has to be created and adhered to is that of the testing strategy for the system. The area of validation and verification for knowledge-based systems is one of active research, and preliminary results have shown that the process of validation is extremely difficult[32–37]. A useful approach that the knowledge engineer could adopt is a testing strategy that has a wide test coverage, to locate as many sources of software failure as possible[34]. In addition to this, a detailed plan of critical testing that attempts to locate errors in extreme or critical areas should be prepared. This strategy should help in raising the reliability of the system[38].

MAINTENANCE AND REFINEMENT

A major benefit that follows from the production of the specifications is that they facilitate an ability to maintain easily the system as the knowledge engineer has a complete and unambiguous record of all stages in the development process[39]. Thus when modifications have to be made then the changes commence at the specification level and the system is updated from the new specifications – the system is thus specification driven.

The specifications being independent of implementational forms allows decisions to be made in such a way that if the knowledge engineer were, say, forced to change the implementation language, it would be easier for a new implementation to be created from a series of specifications rather than, for example, a large volume of Lisp code.

The specifications therefore have many roles, but acting as the basis for the entire system documentation is among the most vital of them.

SUMMARY

The aim of this paper has been to show that it is possible to develop a set of specifications from which an implementation can be produced and that the whole process of developing an expert system does not have to be hit or miss but can in fact be formal and rigorous.

The methodology described above basically consists of four parts. First, the textual elicited representation gets transformed through representation refinement into the primary representation. This representation is central to the whole process, as it is this more rigorous representation that acts as the basis for the second major part – the creation of the formal domain specification. This formal specification being in the Z language acts as a means of removing ambiguity and checking for adequacy in the completeness and consistency of the elicited knowledge. The primary representation also acts as the foundation of the mechanism through which the representation is selected and subsequently specified.

The fourth part of the development is the creation of the concrete specification, of which the secondary representation forms one part and the formal specification of the control architecture in Z the other. Finally, the implementor takes the concrete specification and the man–machine interface considerations plus the user specification to implement the system.

Thus, rather than have an *ad hoc* approach to developing systems, this paper suggests a methodology that uses several formal specifications and attempts to justify rigorously the processes used to achieve them.

ACKNOWLEDGEMENTS

The author thanks the referees for their comments, and Professor Thornes and the Geography Department at the University of Bristol, UK, Professor Shave at the University of Liverpool, UK, and Professor Moralee's Research Group at Unilever Research for their comments and insights.

REFERENCES

1 **Davis, R** 'Expert systems: where are we? and where do we go from here?' *MIT-AIM-665* MIT, Cambridge, MA, USA (1982)
2 **Grover, M D** 'A pragmatic knowledge acquisition methodology' in *Proc. IJCAI 8*(1983) pp 436–438
3 **Buchanan, B G, Barstow, D, Bechtal, R *et al.*** 'Constructing an expert system' in **Hayes-Roth, F, Waterman, D A and Lenart, D G (eds)** *Building Expert Systems* Addison-Wesley, USA (1983)
4 **Wielinga, J B and Breuker, J A** 'Analysis techniques for knowledge-based systems: part 1' *Report 1.1 Esprit Project 12* CEC, Belgium (1983)

5 **Weiss, S M and Kulikowski, C A** *A Practical Guide to designing Expert Systems* Chapman and Hall, UK (1984)

6 **Alexander, J H, Freiling, M J, Shulman, S J *et al.*** 'Knowledge level engineering: ontological analysis' in *Proc. AAAI5* Philadelphia, PA, USA (August 1986) pp 963–968

7 **Weitzel, J R and Kershberg, L** 'Developing knowledge-based systems: reorganising the system development life cycle' *Commun. ACM* Vol 32 No 4 (1989) pp 482–490

8 **Royce, W W** 'Managing the development of large software systems' in *Proc. WESTCON* CA, USA (1970)

9 **Jones, C B** *Software Development: A Rigorous Approach* Prentice Hall, USA (1980)

10 **Jacob, R J K** 'Using formal specifications in the design of a human-computer interface' *Commun. ACM* Vol 26 No 4 (April 1983)

11 **Sufrin, B A and He, J** 'Specification, analysis and refinement of interactive processes' in **Harrison, M and Thimbleby, H (eds)** *Formal Methods in Human-Computer Interaction* Cambridge University Press (1990) pp 153–200

12 **Spivey, J M** *Understanding Z: A Specification Language and its Semantics* Cambridge University Press, UK (1989)

13 **Spivey, J M** *The Z Notation* Prentice Hall, USA (1990)

14 **Summerville, I** *Software Engineering* Addison-Wesley, UK (1989)

15 **Holy, M** *Erosion and Environment* Pergamon Press, UK (1980)

16 **Kirkby, M J and Morgan, R P C (eds)** *Soil Erosion* John Wiley, USA (1981)

17 **Dadkhah, M and Gifford, G F** 'Influence of vegetation, rock cover and trampling, on infiltration rates and sediment production' *Water Resources Bull.* Vol 16 (1980)

18 **Djorvic, M** 'Slope effect on run-off and erosion' in **de Boot, M and Gabriels, D (eds)** *Assessment of Erosion* John Wiley, USA (1980)

19 **Cannell, G H and Weeks, L G** 'Erosion and its control in semi-arid regions' in **Hall, A E, Cannell, G H, and Lawton, H W (eds)** *Agriculture in Semi-Arid Environments* Springer-Verlag, Germany (1979)

20 **Burton, A M, Shadbolt, N R, Hedgecock, A P and Rugg, G** 'A formal evaluation of knowledge elicitation techniques for expert systems: domain 1' in **Moralee, D S (ed)** *Research and Development in Expert Systems IV* Cambridge University Press, UK (1987)

21 **Martin, M, Sanguesa, R and Cortes, U** 'Knowledge acquisition combining analytical and empirical techniques' in *Proc. Expert Systems and Their Applications 11* Avignon, France (27–31 May 1991)

22 **Nawana, H S, Paton, R C, Bench-Capon, T J M and Shave, M J R** 'Facilitating the development of knowledge-based systems: a critical review of acquisition techniques and their techniques' in *Proc. Expert Systems and Their Applications 11* Avignon, France (27–31 May 1991)

23 **Bullemer, P, Bloom, C and Richardson, R** 'Knowledge acquisition for large procedural knowledge-based systems: getting the most out of your experts' in *Proc. 4th Int. Conf. Industrial and Engineering Applications of Artificial Intelligence and Expert Systems* Koloa, HA, USA (2–5 June 1991)

24 **Bainbridge, L** 'Verbal reports as evidence of the process operator's knowledge' *Int. J. Man-Mach. Stud.* Vol 11 (1979) pp 411–436

25 **Kelly, G A** *The Psychology of Personal Constructs* Norton, USA (1955)

26 **Bennett, J S** 'On the structure of the acquisition process for rule-based systems' in *Machine Intelligence: Infotec State of the Art Report Series 9 No 3* Pergamon Infotech, UK (1981) pp 189–204

27 **Welbank, M** *A Review of Knowledge Acquisition Techniques for Expert Systems* British Telecom Research Labs, Martlesham Consultancy Services, UK (1983)

28 **Pollack, S L** *Decision Tables: Theory and Practice* Wiley-Interscience, USA (1971)

29 **Hicks, R and Lee, R** *VP-Expert for Business Applications* Holden Day Press (1988)

30 **Morgan, C** *The Schema Language* Programming Research Group, Oxford University, Oxford, UK (1984)

31 **Sufrin, B** *Mathematics for system specification* Programming Research Group, Oxford University, Oxford, UK (1985)

32 **O'Keefe, R M, Balci, O and Smith, E P** 'Validating expert system performance' *IEEE Expert* Vol 2 No 4 (1987) pp 81–90

33 **O'Leary, D E** 'Validation of expert systems – with applications to auditing and accounting expert systems' *Decision Sci.* Vol 18 (1987) pp 468–486

34 **Rushby, J** 'Quality measures and assurance for AI software' *NASA Contractor Report 4187* (1988)

35 **Constantine, M M and Ulvila, J W** *Testing Knowledge-Based Systems: The State of the Practice and Suggestions for Improvement: Expert Systems with Applications Vol 1* (1990) pp 237–248

36 **Chang, C L, Stachowitz, R A and Combs, J B** 'Validation of nonmonotonic knowledge-based systems' in *Proc. AAAI Workshop Notes on Validation and Verification* Boston, MA, USA (29 July 1990)

37 **Botten, N and Raz, T** 'Knowledge base verification using matrices' in *Proc. 4th Int. Conf. Industrial and Engineering Applications of Artificial Intelligence and Expert Systems* Koloa, HA, USA (2–5 June 1991)

38 **Plant, R T** 'Reliability of expert systems' in *Proc. TIMS/ORSA Joint Nat. Conf.* Las Vegas, NV, USA (May 1990)

39 **Plant, R T** 'Factors in software quality for knowledge-based systems' *Inf. Soft. Technol.* Vol 33 No 7 (1991) pp 527–536