Int. J. Human-Computer Studies (1997) **46,** 523–547

Using meta-knowledge within a multilevel framework for KBS development

Robert T. Plant

Department of Computer Information Systems, University of Miami, Coral Gables, FL 33124, USA

Rose Gamble

Department of Mathematical and Computer Science, University of Tulsa, OK 74145, USA

(Received 6 November 1996 and accepted in revised form 6 December 1996)

This paper describes a multilevel development life cycle of representation refinement for knowledge-based systems that incorporates meta-knowledge at each level. The methodology uses formal techniques in the specification of the domain knowledge, the cognitive aspects and the representation. The methodology provides the knowledge engineer with a dynamic perspective of the system which can be used in conjunction with the static aspects found in the representation abstractions. To provide perspective, the paper details the refinement of one of the levels called the intermediate level, in which an implementation-independent representation is created by the use of a knowledge filter.

© 1997 Academic Press Limited

1. Introduction

Research and development in the area of methodologies for knowledge-based system (KBS) development has often been approached in a piece-meal fashion. Frequently, researchers focus upon single development areas such as knowledge acquisition (Madan, 1995), prototyping (Jones, 1995), verification and validation (Gamble, Roman, Ball & Cunnigham, 1994; O'Leary, 1994; Plant & Preece, 1996), or on the development of a particular application (Waterman, 1986). Recently, researchers have been placing increased emphasis on more comprehensive development methodologies for KBSs focusing upon the static functionality of the systems. In this paper, we move away from a static perspective towards an alternative philosophy of system design, using *meta-knowledge*, that relies upon formality and specification refinement. The paper will present an overview of this approach and indicate how meta-knowledge can be obtained and used to assist in the design and validation processes.

1.1. KBS DEVELOPMENT METHODOLOGIES

An examination of previous research in the development of methodologies for knowledge-based systems identifies eight major models: Buchanan's (Buchanan *et al.*, 1983), Davis's (Davis & Lenat, 1982), Grover's (1983), Alexander's (1986), Miller-IS Model (1990), AISE (ANSI, 1992) and CommonKADS (ESPRIT 5248). These models can be assessed against the TRILLIUM$_K$ evaluation scale, a scale that assess the rigor and formality of a methodology on a scale of one to three (Preece, 1995). The assessment of

TABLE 1
$TRILLIUM_K$ quality assessment matrix

	Methods and scores							
Phase	Buchan	Davi	Grover	Alexander	Weitze	Miller	AIS	CKA
Problem	2	2	2	3	2	3	3	3
Conceptual	2	1	2	3	2	2	3	3
Design model	1	1	2	—	2	3	3	3
Implemented	1	1	2	—	2	3	3	3
Verification	1	1	1	—	1	2	1	2
Validation	1	1	2	—	2	2	2	2

the methodologies against this scale has been studied (Plant, 1994) and the results are summarized in Table 1.

The development methods for knowledge-based systems are focused upon the use of increasingly formal aspects of system development, such that the proof obligation for systems can be ultimately determined and met (Baughman & Gamble, 1996; Gamble & Shaft, 1996). These development mechanisms are at or approaching the $TRILLIUM_k$ Level 2 capability† with certain aspects moving towards Level 3 capability (Preece, 1993). The use of the TRILLIUM evaluation scale is useful in determining formality; however, the most telling aspect of a systems rigor is the degree to which it can be verified and validated.

From Table 1, we can see that even though methodologies exist that formalize their development process, the evaluation surrounding these processes through their validation and verification has not yet reached the same level of formality.‡ Thus, this paper attempts to show that through the use of meta-knowledge and a methodology based upon refinement of that meta-knowledge, a higher degree of verification and validation is ultimately achievable.

Section 2 outlines both the rationale for using meta-knowledge in system creation and a four-stage development methodology. The section also introduces the concept of a composite specification, a mechanism for combining the partial specifications of individual aspects of the system. The composite specification is an approximation of the formal specification of the whole system.

Section 3 details the use of meta-knowledge in a complete life-cycle model, in which four levels of abstraction characterize the system development, each connected to the other through the meta-knowledge model. This approach draws upon the philosophy of both representational refinement and Newell's knowledge level.

† $TRILLIUM_k$: a framework of assessment developed by Preece for Bell Canada (Preece, 1993) that increases from Level 1 where informal development is used to Level 4 where a completely formal approach is used; see Appendix.

‡ Validation can be defined as "are we building the right system" and verification as "are we building the system right". The area validation and verification has an extensive literature (Preece, 1993; Plant, 1994; Gamble & Landuaer, 1995).

Section 4 examines the intermediate level in more detail and considers the relationship that the meta-knowledge plays in refining the elicited knowledge through to the formal level. This is achieved in part through the use of a knowledge-filter concept and the use of formal methods to create independent domain, cognitive engineering and representation specifications.

The final section discusses the applicability of the approach in practice and the contribution the meta-knowledge approach makes to improved validation and verification.

2. The specification of knowledge-based systems

The natural point from which to develop any software system is the creation of a specification. The specification should ideally detail every aspect of the system in unambiguous terms that all interested parties can consider. The task of creating such a specification for knowledge-based systems is, however, far from easy for any but the most trivial of systems. In the light of this problem, knowledge engineers have been often forced to proceed with only a minimal specification or no specification at all. This is a less than ideal situation and a source from which many subsequent developmental problems emanate.

2.1. A MULTILEVEL REFINEMENT PHILOSOPHY

In order to overcome the problem of weak specifications in knowledge-based system development we combine several techniques and approaches. First, the lifecycle is based upon *multiple levels of refinement* as illustrated in Figure 1.

Through these levels the problem is taken from its abstract outline form, starting with a definition of that problem at the *conceptual level* in the form of an operational concept (Miller, 1990). This baselines the problem for the rest of the development. Elicitation of knowledge is then performed upon the problem space. The *intermediate level* captures, consolidates and distills the knowledge from the multiple elicitation techniques utilized.

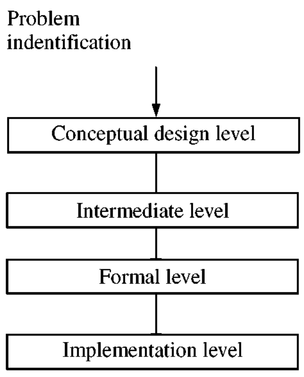

FIGURE 1. Development levels in KBS development.

The intermediate level attempts to capture the knowledge in a form suitable for further refinement. The refinement process to the third level, the *formal level*, is the normalization of the elicited knowledge held in the intermediate representation. This process creates three formal specifications that capture the domain knowledge, the cognitive aspects of the system and the representation. The final level, the *implementation level*, is the consolidation of all the formal specifications into an implementation.

2.2. A COMPOSITE SPECIFICATION PHILOSOPHY

As mentioned in Section 2.1, the creation of an operational concept or initial specification provides the knowledge engineer with a problem description from which to develop the system. This development is based upon the creation of a *composite-specification*, a set of specifications, each of which focuses upon a particular aspect of the development process such as the domain-knowledge or the representation. The composite attempts to overcome the lack of a total formal specification from which to derive the system.

We illustrate the four areas where specifications can be created in relation to a knowledge-based system, with varying degrees of formality.

In Figure 2, two distinct types of specifications are present: the *dynamic* specifications and the *static* specifications. Dynamic specifications refer to aspects of the system that are under constant change or for which the interaction of the components are undetermined due to factors such as: combinatorial complexity, incompleteness in the knowledge base or use of heuristic information. Static specifications refer to those aspects that do not change, but rather remain consistent over a period of time. Thus, there are four static specifications: the *initial specification*, the *domain specification*, the *representation specification* and the *cognitive engineering specification*. The dynamic specification is that information captured in the *meta-knowledge model*. The meta-knowledge model is dynamic because it holds information about the behavior and inter-relationships among the other static specifications. Since the behavior and relationships are defined during refinement, the meta-knowledge model is changing as development proceeds. The

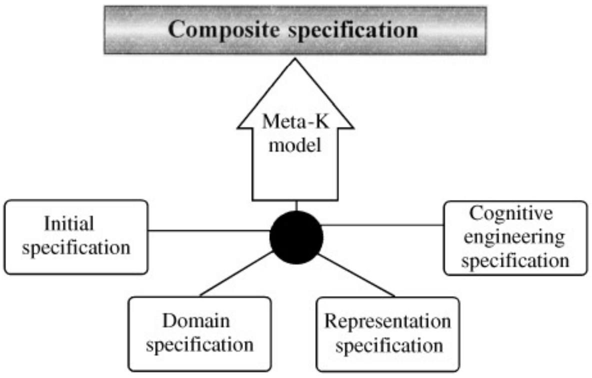

FIGURE 2. A composite specification.

meta-knowledge model is the facilitator of refinement between the levels associated with Figure 1.

We will now briefly consider the relationship between meta-knowledge and the philosophy of design as utilized in the methodology of this paper. A more detailed discussion of their inter-relationships is presented later.

2.3. META-KNOWLEDGE

In order to see, therefore, the relationship between the creation process of knowledge-based systems and their subsequent verification and validation, two perspectives of the system need to be considered: the *static* perspective and the *dynamic* perspective. The static aspect refers to the structure of the knowledge, while the dynamic aspect refers to the behavior of that knowledge during execution. These perspectives are analogous to the static and dynamic specifications presented in Section 2.2. Thus, static knowledge supports the *verification* process, i.e. showing the system was built right and moves toward formal functionality in system design. The dynamic knowledge supports the *validation* research, i.e. showing the right system was built.

There is an established and growing literature on the verification of systems, much of which has been performed on the static structure of the rules (Preece, 1993; Plant, 1994; Gamble & Landuaer, 1995; Schmolze & Vermesan, 1996). This research includes verification tools which identify inconsistencies and incompleteness in the knowledge base (Plant, 1997). A system is *inconsistent* if it asserts something that is not true of the modeled domain. A system is *incomplete* if it lacks deductive capability (Morell, 1989). Thus, the question arises as to what we are to verify our knowledge-based systems against, given the difficulty of obtaining a complete/total specification for the domain.

The approach advocated in this paper is the creation of a composite specification of the system functions that are known and which can be defined formally as discussed in Section 2.2. However, in order to support a composite or partial specification, additional information about the knowledge base needs to be obtained, such as the history of development, expert conflicts, etc. This dynamic support knowledge, termed *meta-knowledge*, is vital for verification (Morrell, 1989) and is derived using the knowledge-elicitation process.

The concept of meta-knowledge is most commonly used in relation to knowledge representation and can be defined as "structures that describe other structures" (Barr, Bennett & Clancy, 1979). The use of meta-knowledge in this way was extensively covered by Davis (1980), who divided knowledge into two types: (1) object-level, which comprises information about a task domain and (2) meta-level, which comprises information (*meta-knowledge*) about the object-level. Davis advocates the attachment of meta-knowledge to rules, such that the meta-knowledge can be used to control rule firing for performance enhancement. However, from the aspect of knowledge acquisition for cognitive modeling, meta-knowledge captures intrinsic, commonplace properties in human cognition that are central to an understanding of knowledge and intelligence (Barr *et al.*, 1979).

An example set of categories into which a knowledge engineer may use meta-knowledge in knowledge-based system development is accuracy, applicability, source and reliability (Morell, 1989). Meta-knowledge may also be found in a structural engineering knowledge base, where the framework allows for the structural engineers to annotate a design by explaining why changes were made to the design, the history of those changes and any interrelationships among the information that had also changed. Other meta-knowledge may be included, such as the life span of the domain knowledge of the structure, sources of knowledge and history of knowledge-base maintenance. From this it can be seen that meta-knowledge can be a useful mechanism to assist in validating the process of system development, as well as supporting system verification.

The need to incorporate meta-knowledge into a system design is therefore essential; however, this aspect of the design process has tended in the past to be overlooked. This was perhaps overlooked due to the simplicity of the methodologies being deployed and the nature of the systems being created.

The value of meta-knowledge to the entire development process is reported by other researchers to be extremely high (Hayes, 1973; Sandewall, 1975; McCarthy, 1979; Doyle, 1980; Weyhrauch, 1980; Warren, 1981; Brown, 1982; Gallaire & Lasserre, 1982; Genesereth & Smith, 1982). However, to derive maximal benefit from the information, a model of the meta-knowledge needs to be placed within the context of a rigorous development methodology such that the meta-knowledge can be obtained and applied appropriately during the development. In this paper, we incorporate into each level (defined by its representative specification) a component that models the meta-knowledge at that level. The meta-knowledge model aids in specification refinement within a level and between levels.

2.4. NEWELL'S KNOWLEDGE LEVEL

We have so far outlined two design philosophies upon which the methodology is based: that of multilevel refinement utilizing a composite specification paradigm and that of meta-knowledge in which the levels and specifications of the methodology are held together. In order to justify this approach further, we will take a third philosophical stance and relate the philosophy of design utilized here to that of Newell's Knowledge Level (Newell, 1982).

In order to understand the basis upon which Newell postulated his Knowledge-Level, it is necessary to consider the following two questions: "What is the nature of knowledge? How is it related to representation?" (Newell, 1982). The answers to these questions are complex and have many different perspectives. We present an overview of these issues in order to appreciate the use of the knowledge-level paradigm in the context of this paper.

Newell viewed computer systems as being at a series of levels, as illustrated in Figure 3. These levels start at the bottom with the device level, through the logic level, to the program level or symbol level, finally to the configuration level (processor memory switch level). A level was defined as follows: "A level consists of a *medium* that is to be processed, *components* that provide primitive processing, *laws of composition* that permit components to be assembled into *systems* and *laws of behavior* that determine how system behavior depends on the component behavior and the structure of the system" (Newell, 1982). The levels were also defined in two more ways: (i) "a level can be defined autonomously, without reference to any other level" and (ii) "each level can be reduced to the one below it, such that each aspect of a level—medium, components, laws of composition and behavior can be defined in terms of systems at the level below" (Newell,

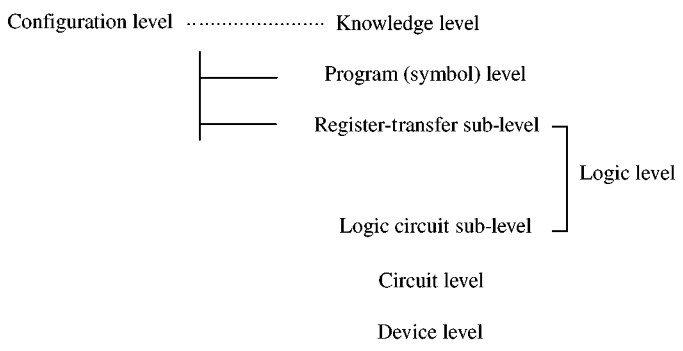

FIGURE 3. Computer system levels (Newell, 1982).

1982). Finally, the levels are constrained by further two issues: "Each computer system level is a specialization of the class of systems being described at the next lower level" and "computer system levels are approximations" (Newell, 1982).

Having defined the notion of levels, Newell then placed a further layer onto the model and proposed a knowledge level that resides above and is separate from the symbol level. This level is characterized by *the knowledge level hypothesis*: "There exists a distinct computer systems level, lying immediately above the symbol level, which is characterized by knowledge as the medium and the principle of rationality as the law of behavior" (Newell, 1982).

The philosophy of levels and the assertions that denote their formulation are a natural mechanism for the methodology we describe in this paper. It can be seen that the operational concept of the initial specification defines the very highest abstract level a system can take, and that this system, refined through intermediate levels, can be transformed into the data structures and control mechanisms of the symbolic level. Thus, to quote Newell once again in an attempt to answer the two postulates at the beginning of this section, we aim at drawing the three philosophies described in Section 2 together and, hence, provide the basis for the methodology to follow: "The theory of the knowledge level provides a definition of representation, namely, a symbol system that encodes a body of knowledge. It does not provide a theory of representation, which properly exists only at the symbol level and which tells how to create representations with particular properties, how to analyse their efficiency, etc. It does not suggest a useful way of thinking about representation is according to the slogan equation Representation = Knowledge + Access" (Newell, 1982).

3. Defining a complete life-cycle model

In this section, we present a four-phase life-cycle model describing the overall development process for a knowledge-based system. The intention is to show, at a high-level, how specification refinement can occur *within* each level and *between* levels, provided a meta-knowledge model is included in each level and refined between levels. There is no exact prescription for the internal development of each level due to the variances between systems being created. However, each level does require particular components that are application or user-defined through a suitable or familiar representation.

This section describes the components and representational medium for successful level refinement and transition in each level. However, we first present an overview of the whole life cycle, followed by an overview of the nature and role of the meta-knowledge model.

The four-phase (level) model of knowledge-based system development is depicted in Figure 4. The approach to the description of each level is similar to the principles that Newell described in the levels of a computer system (Newell, 1982). Each level has a representation medium, components, laws of composition and laws of behavior that are defined abstractly. Refinement must take place within each level, with the meta-knowledge accumulating sufficiently to refine the specification to the next level. Thus, there are no feedback lines between levels. If it is necessary to return to a more abstract

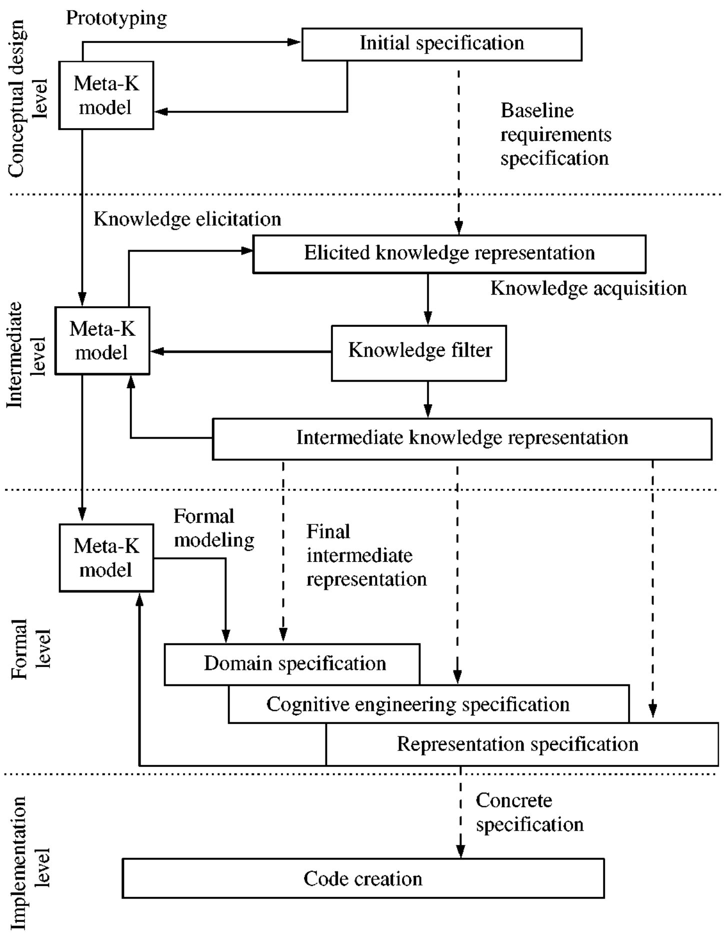

FIGURE 4. A development methodology for knowledge-based systems.

level of specification, the entire life cycle should be repeated with the new knowledge entering as meta-knowledge at the top level. This allows for more complete verification and validation at each level and between levels because of the constancy of the information.

The methodology can be seen as composed of four levels: the conceptual design level, the intermediate level, the formal level and the implementation level. These levels of refinement encompass the five specifications that define the composite specification: the initial specification, the domain, the cognitive engineering and representation specifications, together with the meta-knowledge model. The meta-knowledge model is transformed into subsequent versions of itself in three different levels of the systems development.

3.1. THE META-KNOWLEDGE MODEL

In the development of a knowledge-based system, the knowledge engineer is obligated to not only elicit the staticdomain-specific knowledge, but also has the task of eliciting the dynamic meta-knowledge associated with the domain knowledge, and maintain a separate specification for this knowledge; this is the role of the meta-knowledge model.

The meta-knowledge model plays an important role in a support capacity to the static specifications that are developed during system development. The models aim to provide information about the static knowledge that will enable the static knowledge to be refined at each level in a consistent manner. The meta-knowledge model will act as an oracle to the knowledge engineer detailing the history of data and knowledge items, as well as information that is not represented directly in the static nature of the specification. An example of this may be the life span of the information contained within the specification and the source of that information. This may be important if, for example, the domain is that of a medical database and the drugs in that database have expiration dates or change in reaction after a certain period (potency). The meta-knowledge model may also contain past versions of the information contained in the specification, thus assisting the knowledge engineers in any potential analysis of the system, whether this is a system upgrade, a verification process or retrospective analysis.

The meta-knowledge model needs to be as robust as possible because its current information is refined and new information is accumulated throughout its use in the three layers. As the model moves into the intermediate level, not only will the information pertaining to the refinement of the initial specification be captured in the meta-knowledge model but also, for example, design decisions on the role, nature and choice of elicitation techniques, who performed them, for what duration and side issues such as reliability of expert testimony, focus of the testimony and information on why some information became part of the encoded knowledge while other information was left out. Hence, the use of formality is encouraged through the use of a formal notation such as Z or VDM (Jones, 1980). Formal modeling will enable the relationships between the static domain knowledge and the meta-knowledge be correctly specified and identified. However, it is clear from the variety of information and data formats that not all information can be encoded in a formal notation and, hence, the meta-knowledge model will itself be composed of a variety of data/information representations.

Having identified the significance of the composite specifications in the creation of knowledge-based systems, we will now discuss how these specifications can be integrated together through the use of a rigorous development methodology, and show how these specifications may be equated to knowledge levels.

3.2. CONCEPTUAL DESIGN LEVEL

The aim of this level is to formalize an operational concept of the system to be developed. This operational concept or initial specification will act as the baseline document for the rest of the development process. Following Miller (1990), this phase utilizes iterative prototyping on the initial "problem concept" towards a specification. This phase does not end until all parties {customer, developer, user} agree that they finally understand what the system is intended to do, and in particular how it is supposed to do it; what Miller terms "*the Operational Concept*" (Miller, 1990).

The prototype process is primarily intended to establish the boundaries of the solution space. It is very important that prototyping is used only to this end, as it is extremely detrimental to consider the more complex development issues at this stage, e.g. representation, interface, etc. These decisions would be made on incomplete knowledge of the domain and environment.

To help establish the boundaries of the solution space, the research in cognitive engineering (Woods & Roth, 1988; Mancini, Woods & Hollnagel, 1988; Johnson & Westwater, 1996), which aims to provide design principles in the creation of person–machine systems is made use of (Norman, 1981; Wise, Hopkins, David & Sager, 1993) the use of these cognitive engineering principles will ultimately, later in the development process, produce a specification of all the human–computer interactions involved within the system. However, at this early stage in the life cycle the aim is to influence the creation of the initial specification, such that it accommodates the difficulties that will occur during system creation, as well as alert and arm the knowledge engineer during this process. This aspect of system creation is often overlooked as Roth and Woods note:

> "One of the main reasons that (intelligent) systems failed is that the designers took a narrow view of the knowledge acquisition task. They focused on mimicking how experts solved specific problems rather than attempting to develop a formal specification of the range of problems that arise in the domain and the factors that contribute to problem difficulty" (Roth & Woods 1989).

They describe four pitfalls in building intelligent systems.

- Failing to appreciate the demands of the task.
- Failing to support the human problem solver.
- Assigning user responsibility but not the control of the system.
- Mimicking sub-optimal coping strategy.

In order to overcome these problems, they must first be recognized as actual problems by the knowledge engineer. Once this understanding has been reached, the cognitive engineering techniques can be utilized in this developmental level as well as in subsequent levels. For example, five techniques that assist the knowledge engineer understand the design process and the interaction of these processes, both in terms of the processes themselves and with the user, are as follows.

- Specification of the human–computer interface.
- Cognitive task analysis.
- Knowledge encoding.
- Competence modeling.
- Performance modeling (Roth & Woods, 1989).

The resultant of these will ultimately comprise the specification of the cognitive engineering aspects of the system design. However, it is the understanding that these factors are strong constituents of the development as a whole, which is their main contribution at this level, as they help to shape and influence the initial specification.

The information gathered and created around the cognitive engineering aspect of the initial specification is held in the meta-knowledge model along with the other conceptual-level design-decision information. This information is invaluable in the creation process as a whole and the subsequent maintenance aspects. The meta-knowledge model holds information regarding the process by which the ultimate initial specification was derived, and other information that is not normally held in a specification.

The analogy of this level to Newell's knowledge level is as follows. The medium is data, information and knowledge, the primitive processing occurs over the medium in the form of prototyping, while the laws of composition permit components to be assembled into systems. At this level, the components are a variety of representational forms that compose the initial specification, e.g. math functions defining a potential input set and sets of relations identifying sub-systems within this input set. The laws of behavior may simply be a decision table defining potential condition–action relations within the initial specification, as well as the meta-knowledge which defines the domain in terms of its boundaries and complexity.

3.3. INTERMEDIATE LEVEL

The second of the four levels is the intermediate level. This level is covered extensively in Section 4 and illustrates the use of the knowledge-level paradigm in detail; however, for completeness, an overview of the major concepts in that section will be introduced here.

The intermediate level fulfils a very important function of the life-cycle model, in that it acts as a transformational level, taking the operational concept of the initial specification and through knowledge elicitation and knowledge acquisition, produces a representation of the domain knowledge from which the system will ultimately be constructed. This representation is termed the intermediate representation.

The process can be outlined as in Figure 5.

Having obtained a specification of the systems requirements in Level 1, the task at Level 2 is therefore to extract from the domain expert: knowledge, information and data, upon which the system can be constructed. This is the function of the *knowledge-elicitation process*: the resultant of which is the creation of a set of knowledge representations, each corresponding to the outcome of different elicitation sessions and process. Information pertaining to the elicitation task itself being captured separately in the

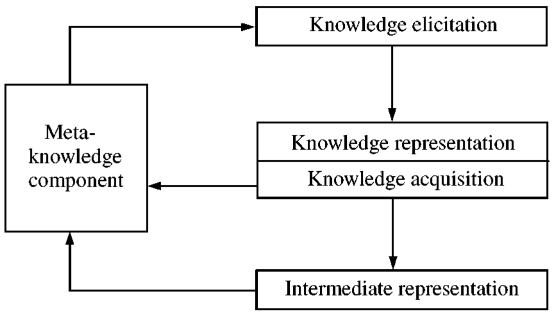

FIGURE 5. Outline of the intermediate level sub-processes.

meta-knowledge model, e.g. its success, nature of the elicitation, sector of the domain that was under consideration, reason for choosing the technique, etc. The elicited representation is then subjected to a *knowledge-acquisition process* in which the elicited form is analysed and knowledge is extracted, becoming transformed into an intermediate representation. The aim of the intermediate representation is to add structure, definition and formality to the knowledge and enable verification to be performed upon it more easily than could be performed on the raw elicited form. The transformation process again feeds back into the meta-knowledge model information regarding, for example, the match between the elicited knowledge and the representations used, the nature and applicability of the information and verification status metrics. The meta-knowledge model then feeding back into the elicitation process itself, thus acting as a control mechanism.

The final outcome is an intermediate representation that will itself be capable of transformation into the next level: the formal level. The intermediate representation acting as a consolidator of the knowledge and medium through which the knowledge can be adequately judged for verification properties.

3.4. FORMAL LEVEL

The second level of the development methodology resulted in a stable and well-defined specification of the systems knowledge: the intermediate specification. From this the knowledge engineer can move forward to the creation of the remaining three rigorous specifications that together with the third refinement of the meta-knowledge model, form the composite specification.

(1) ***Domain specification.*** The aim of the *domain specification* is to capture the information, data and knowledge of the domain in a formal model. This formal model is based upon the final intermediate representation upon which transformational processes are performed. The rationale for the use of a formal domain specification is that the notation used at this level can be more rigorous than at the previous levels. The methodology does not prevent formal notations being used at higher levels of the life cycle; however, the premise of representational refinement underlying the approach advocates that when a domain knowledge is not suitable for immediate mapping to a formal notation then

intermediate steps should be used. This preserves the integrity of the information and knowledge.

A formal notation that has been utilized in practice is the "Z" notation (Spivey, 1990; Gold & Plant, 1994; Murrell, Plant & Gamble, 1996). A specification in a language such as this produces several advantages. First, the Z notation in which it is written is clear, concise, unambiguous and allows for both a technical and non-technical readership. Second, the use of a formal notation has significant maintenance benefits, such as allowing knowledge engineers to keep a correct document of the domain information in an implementation-independent form; this, for example, facilitates maintenance, changes in language or representational form.

The use of formal methods for the representation of the domain knowledge will be significantly beneficial from a verification perspective. The details of the system verification at this level can be captured in the meta-knowledge model, facilitating future system revisions and maintenance.

(2) ***Cognitive engineering specification.*** As briefly mentioned earlier, embedded within the initial specification development process is an aspect of *cognitive engineering* known as *cognitive task analysis* (Roth & Woods, 1989). This process describes the cognitive demands imposed by a particular task, along with the sources of good and poor performance with respect to that task (Woods & Hollnagel, 1987), where the union of these tasks represents the problem space the KBS covers. The aim of cognitive task analysis is to allow the knowledge engineer to determine why a problem is hard, what the typical errors domain practitioners make and how a KBS can reduce error and improve performance (Roth & Woods, 1989). The results of this task analysis are therefore a part of the meta-knowledge model for the conceptual design level and utilized in the intermediate level to perform successful elicitation. Again the task analysis data are captured, held and refined by the meta-knowledge model. This approach to task analysis has also been used effectively in the CommonKADS methodology by Wielinga, Breuker and others in their model of expertise (Breuker & Weilinga, 1987; Hesketh & Barett, 1990; ESPRIT.5248).

The cognitive engineering specification, as we have already noted, is composed of many aspects, including: specification of the human–computer interface, cognitive task analysis, knowledge-encoding, competence and performance modeling. The combined effect of utilizing these cognitive components is very powerful, and can be considered as a significant factor in maintaining the semantic correctness of the system throughout the development process. The cognitive analysis at the formal level processes the intermediate knowledge representation and the meta-knowledge model from that representation. The outcomes can similarly be partitioned in two ways: to the cognitive engineering specification itself and to the meta-knowledge model at the formal level.

The primary artifact that is within the cognitive engineering specification is the specification of the human factors associated with the system, primarily its human–computer interface. The specification of this can be subjected to formal techniques, as demonstrated by Sufrin and He (1990), who use the "Z" notation to specify an interface, and Jacob (1983), who formally specifies a human–computer interface.

The meta-knowledge model at the formal level, in addition to the artifacts surrounding the domain specification, contains several artifacts that surround the cognitive engineering process itself, the first of which, the task analysis model, has already been described. Two other models may also be included.

- *The competence model*: it provides a model of the required competence expected from the model in the domain (Roth & Woods, 1989).
- *The performance model*: it describes the knowledge and strategies that characterize good and poor performance in the domain (Roth & Woods, 1989).

The adoption of these two models, building upon the understanding gained in the first two levels and within the formal level itself, allows the designer to understand the system boundaries more accurately at which performance may degrade. A study by Roth "revealed the brittle performance that can result when systems do not support the human in adapting to problem-solving to different sources of unanticipated variability" (Roth & Woods, 1989). Hollnagle and others, in the field of human factors, have also considered the reliability of the human in terms of cognition and human–system operation (Hollnagle, 1992).

The utilization of these differing cognitive aspects of the systems design allows the knowledge engineer to construct a human–computer interface that maps to the cognitive model used by the problem solver, allowing for more accurate and flexible explanation capabilities. It increases the understanding of the user's needs as an operator and, by defining the interface formally, the variance in the interpretation of the system or its behavior is reduced.

(3) ***Representation specification.*** The third of the formal specifications is the representation specification. The aim of the representation specification 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. In order to find the most suitable form, several influencing factors must be taken into account.

- Information obtained from performing knowledge acquisition upon the intermediate representation.
- Information pertaining to representation selection that can be obtained from considering the composition of the domain specification.
- Information resulting from the cognitive engineering processes.
- Information from the meta-knowledge model.

Each of these information sources provide valuable insights on the selection of a representation scheme and control architecture for the domain under consideration.

The meta-knowledge model will already contain results of the analysis performed during the creation of the intermediate representation detailing the rationale and metrics behind the form chosen. For example, an intermediate representation that is a complete decision table will indicate a production system may be applicable, while a decision tree may indicate inheritance and the use of a frame-based representation. This can also be refined by considering the composition of the meta-knowledge model, the framework in which the meta-knowledge is represented and the data typing indicated by the formal notation. For example, an intermediate representation of a decision table may have an associated meta-knowledge model framework of hooks or slots containing meta-knowledge that would facilitate meta-rules to be attached to a production system architecture, and this can then be accommodated in the semantics and syntax of the representation.

Further refinements to the specification will also result from the consideration of the formal domain specification and cognitive engineering specifications; the former indicating specific data typing and structural requirements as well as issues pertaining to the nature of the solution space. For example, will certainty factors, probability analysis or statistical analysis be required to overcome incompleteness in the domain knowledge or data requirements? Similarly, cues from the analysis of the cognitive engineering specification may lead to changes in the representation from the norm to accommodate domain-specific or human factor requirements. The information surrounding these design issues and constraints are captured and represented in the meta-knowledge model at the formal level.

Once the knowledge engineer has determined the representational needs for the system, it can then be formally specified in terms of its semantics (Lassez & Maher, 1983; Murrell & Plant, 1995) and syntax (Gold & Plant, 1994), which forms the representation specification (Craig, 1991). The advantages of having an independent formal specification of the representation in conjunction with the meta-knowledge model are that any changes to the system can be done in light of an understanding of the consequences that change may have on any other aspect of the system. Furthermore, there may be a need to change the representation at some point, and that can be reasoned over too.

The specification of the representation allows the knowledge engineer an opportunity to consider and identify those aspects of the knowledge-representation language to be used and specify them in a formal manner, in terms of its denotational semantics and syntax. For without these, it is extremely dificult to reason about a domain description/representation with any certainty. The selection of a representation involves difficult considerations and on which there is an extensive literature (Brachman & Levesque, 1985; Doyle & Patil, 1989; Davis, Shrobe & Szolovits, 1993).

3.5. IMPLEMENTATION LEVEL

Having created all aspects of the partial specification, we are now at a point at which these specifications can be combined into a form that will allow us to move towards implementation. This stage is known as the *concrete specification.*

The creation of the concrete specification is in stages: first, the domain knowledge is transformed from its "Z" specification into the form advocated by the representation specification and second a formal specification of the control architecture that is associated with the representation is created.

It should be noted that this is not the implementation, since the representation is a hybrid between a high-level version of what is to be implemented and a formal specification in the style suggested by the syntax and semantics of the representation specification (e.g. pseudo-code). The aim is to produce an implementation-independent representation of the system. This will allow the knowledge engineer to have a simplified version (minus the complex syntax) with which to reason out the implementation later in the life cycle, e.g. maintenance.

Having created the concrete specification, it is then used as the basis of the system implementation; the interface issues being resolved by the referral to the meta-knowledge model at the formal level, the cognitive engineering specification and man–machine interface specifications.

It should be noted that the implementation level does not have a meta-knowledge model of its own, but utilizes the meta-knowledge contained in the formal level. This is due to the directness of the mapping between the formal level and the implementation. A meta-level could be created if desired to hold the selection criteria for the implementation language. All other maintenance decisions emanate from the higher levels of the specification and, thus, the top three meta-knowledge models would all capture the decision-making surrounding these changes.

The implementation of the system should be the most straight-forward of all the stages, due to the high degree of structuring and refinement that has been performed upon the system in the previous phases. The mechanism through which the system is implemented is left open to the knowledge engineer.

4. Detailing a specific level of abstraction

In this section we discuss in more detail the level of specification abstraction called the intermediate level.

4.1. THE KNOWLEDGE ELICITATION PROCESS

The unique nature of knowledge-based systems is that they utilize domain-specific information that is "*expert*" in nature, in that the information may in itself unique, scarce or uncommon. However, it is the way that the expert employs that information which makes the information valuable. Thus, one of the most important tasks befalling the knowledge engineer is to ensure that as much useful structural control and relational knowledge is elicited from the expert source as possible. This process forms the basis of the knowledge elicitation task upon which the knowledge-based system development is performed. In the elicitation process, the knowledge engineer can extract both static domain knowledge, e.g. facts, rules, heuristics, etc., and select a representation (the elicited knowledge representation) in which to manipulate this knowledge. Thus, the knowledge engineer's task in elicitation can be seen as two-fold: the elicitation of static-domain-specific knowledge and the elicitation of dynamic meta-knowledge.

The choice of elicitation technique, for which there is an extensive literature (Welbank, 1983; Diaper, 1989; Scott, 1991; Madan, 1995) will depend heavily upon the domain under consideration, the type of knowledge to be extracted and the point the elicitation has reached; the meta-knowledge model being used to control and direct the elicitation process. 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. Several frameworks for the analysis of these techniques have been proposed (Burton, Shadbolt, Hedgecock & Rugg, 1987; Dhalival & Benbasat, 1990), including *cognitive* *mapping* and *knowledge encoding*, two aspects of Woods' cognitive engineering paradigm (Woods & Roth, 1988; Roth & Woods, 1989).

The resultant of the elicitation process, depending upon the technique employed, will be, what we have termed the elicited representation. This will, for example, be a transcript in the case of an interview or an on-line report. The aim of this stage in the life cycle is to provide a permanent record of the knowledge, in the form in which it was extracted. This will enable the knowledge engineer to follow an "elicitation trail" later if necessary in conjunction with the meta-knowledge model (e.g. maintenance phase).

The process of eliciting the different knowledge types, often from different sources, with differing knowledge levels, using different techniques, at different periods of time means that there will be a set of elicited representations which together form a historical database of elicited knowledge. It is the integration of these differing forms of knowledge and representations that is the focus of the knowledge acquisition process, in which we utilize the concept of a *knowledge filter* to isolate different aspects of knowledge, the type of that knowledge, the inter-relationships, the heuristics and meta-knowledge associated with each item of the domain knowledge. The knowledge filter attempts by using different techniques upon different elicited forms to extract the maximum information from the knowledge, which then gets integrated together to form the intermediate representation; we will discuss this further in the next section.

4.2. THE KNOWLEDGE FILTER

To handle distinct elicited forms in practice, a knowledge filter is needed. In order to do this, we introduce the software development process called the Knowledge Filter. The filter comprised of a series of sub-processes, each of which processes the elicited representation in order to distill a resultant output that reflects the sub-process function. For example, if the elicited representation were a transcript, one of the sub-processes utilized would be conversational coherence in order to obtain an understanding of the *alignment* within the dialogue (Ragan, 1983). The sub-processes can then be utilized in the subsequent system development.

The knowledge filter depicted in Figure 6 is an example of where the application of the knowledge level principle assists in understanding, and correctly structuring a set of processes. An examination of these processes in relation to Newell's paradigm reveals that the elicited representation acts as the *medium* to be processed. The sub-processes of the knowledge filter act as *components* that provide the primitive processing. The analysis of the information resulting from the primitive processing results in the meta-knowledge model and the intermediate representation, both of which can be defined formally. The formal processes themselves act as the *laws of composition* which permit the components to be assembled into systems. The meta-knowledge model and the intermediate representation are also bound through formal descriptions of their behavior, e.g. what can and cannot be added to them, and thus these form Newell's *laws of behavior*. Further, each of the representations can be considered in isolation or rigorously transformed into the next representation, thus obeying Newell's constraints. We can see, therefore, that these form one level of a methodology which, when added to the other levels, provide a description of a complete knowledge-based system development environment.

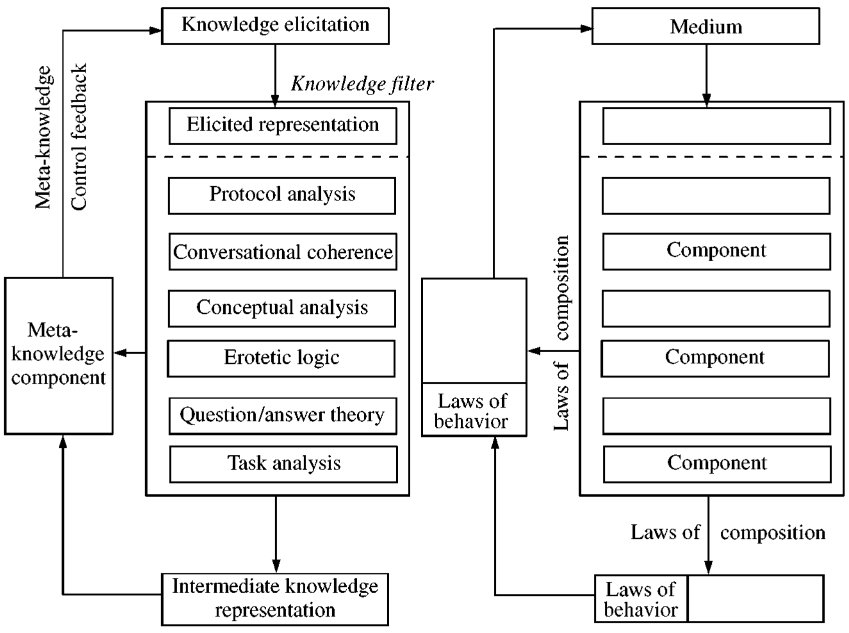

FIGURE 6. The intermediate level of development.

4.3. THE META-KNOWLEDGE MODEL

As we have discussed in the previous section, the knowledge filter acts to isolate the different aspects of information contained within the elicited representation. The resultants of this are the refined meta-knowledge model and the intermediate representation.

The meta-knowledge identified during the knowledge filtration process becomes an aspect of the meta-knowledge model, where a formal notation is again advocated to represent its information. The meta-knowledge is then fed back into the elicitation process to enable the knowledge engineer to both control the elicitation process and obtain the grain size and scope of knowledge required with meta-knowledge also being generated further. The process is continued until a steady state in the elicited representation has been reached. This is similar in concept to the use made of explanations by Davis (1980) in TEIRESIAS. However, the model presented here is abstracted from implementation constraints by virtue of containing its own formal model of the domains' meta-knowledge.

In order to achieve a steady state, a *meta-knowledge controller* is used in conjunction with a meta-system model. The meta-knowledge controller represents a feedback loop that the knowledge engineer uses to identify desirable changes to the elicitation process and, consequently, the elicited representation. This feedback utilizes the meta-knowledge model to contrast the status of the elicited knowledge with the operational concept, by which the system is bounded and the status of the intermediate representation. The creation of the meta-knowledge level framework assists the knowledge engineer verify the knowledge in the intermediate representation in terms of completeness, correctness and consistency. Thus, the knowledge engineer can assess more accurately the progress of the elicitation process towards a steady state as defined by the operational concept.

The relationship between the elicitation of meta-knowledge and the system boundary is a factor that needs careful consideration, as we need to ensure that the controller acts to elicit only knowledge within what Checkland (1981) has termed the "relevant" system. The concept of the "relevant" system is based upon Anaxagorus's Theorem that "in everything there is everything" and that "in all problems are to be found all problems", such that we can, if we are not careful, elicit knowledge that although related to our problem domain, is outside of the boundary that was defined in the operational concept. This is important from a validation perspective; otherwise, we would have no limit to the system domain. Gigch puts a systemic perspective on this dilemma:

> "if we push the system boundaries too far, we face the problem of having to consider too many systems, the situation becomes too complex and unsolvable … if we do not push the boundary far enough, we face the 'environmental fallacy' (Churchman, 1979) when not all the relevant systems are taken into account" (Gigch, 1984).

Thus, the meta-knowledge controller has an important role in determining that the grain size and scope of the knowledge is within the system boundary, and further, that the boundaries be respecified if examination of the domain meta-knowledge determines this to be necessary in order to achieve a validated and verified system. The meta-knowledge controller can be defined in terms of Gigch's three-part meta-system paradigm.

(i) A hierarchy of problem-solving levels in which higher system levels can judge and rate solutions at the lower level.
(ii) A framework to provide evaluation criteria in meta-language terms, i.e. a language appropriate to judge lower systems solutions.
(iii) A guarantee of truth at each systems level, except the very last (or highest).

We can utilize this paradigm as follows.

(i) The hierarchy of problem-solving levels can be equated to Newell's knowledge level such that the knowledge elicited is fed into the elicited representation at a lower level, and it in turn is fed into the intermediate representation at the next lower level, and so on recursively. Davis (1980) recognizes this in his discussion of meta-rules at the application level, where "rules at one level in the hierarchy, for instance, are used to control the invocation of rules at the next lower level, while at the same time they are data to the rules above which examine and reason about them". The meta-knowledge contained in the meta-knowledge controller is used to judge and rate the validity of the elicited representation.
(ii) The meta-knowledge model provides the framework for evaluation of the knowledge elicitation and the elicited representation.
(iii) The framework is formal, and as such facilitates the evaluation of the meta-knowledge as well as the mapping of that knowledge through increasing levels of rigor as the development moves from the conceptual level to the implementation level. Rigor is possible at all levels except at the highest level where a formal specification of the system cannot be created; thus, the correctness of the system cannot be guaranteed at this level.

The meta-knowledge feedback loop can also be used to address a relationship between the object-level and meta-level knowledge that Davis (1980) identified.

"The forms of knowledge captured well at the object-level will similarly be easy to express at the meta-level; those difficult to represent at the object-level will prove equally difficult at the meta-level" (Davis, 1980).

If the elicitation of meta-knowledge proves difficult, this is an indication that the elicitation and representation of the object-level is also weak and not yet at the steady state and, thus, the elicitation needs to continue with perhaps the application of an alternative elicitation technique. This is where the formalized representation of the meta-knowledge proves valuable as it allows a more accurate determination of the status of the representation to be made, in terms of completeness, correctness, consistency, grain size, etc., without which only informal estimates can be made.

The aim of the meta-knowledge model, therefore, is to achieve a steady state, whereupon the meta-knowledge model in conjunction with the intermediate representation feeds into the next level, such that the formal domain, cognitive engineering and representation specifications can be constructed. The advantage of using the meta-knowledge model is that the knowledge engineer is no longer creating a system based only upon static domain information, but also utilizing and controlled by meta-knowledge through the paradigm of Newell's knowledge level.

4.4. THE INTERMEDIATE REPRESENTATION

The second output of the knowledge filtration process is the production of the intermediate representation (Plant, 1991; Scott, 1991), a representation more structured than the elicited representation in which to hold the elicited knowledge. The knowledge engineer uses this intermediate representation as a focus point for the knowledge filter and assesses its status in conjunction with the meta-knowledge model, through the control mechanism described earlier. Intermediate representations are of the form: decision tables, AND/OR graphs, decision trees, each of which encourage completeness, correctness and consistency, allow for refinement and reduction while having clean yet concise structures.

The creation of the dynamic meta-knowledge model and the static intermediate representation allows the knowledge engineer to have a dual perspective in the remainder of the system development. The aim of the intermediate representation is to provide a more rigorous form than the elicited representation with which to reason over the domain, while the meta-knowledge representation contributes by allowing the knowledge engineer to understand and be more sensitive to the dynamic aspects of the systems development, e.g. the use of meta-knowledge allows us to enhance our understanding of the systems explanation capability, as defined in our cognitive engineering specification.

The meta-knowledge model and the intermediate representation are used in the creation of three specifications at the next knowledge level: the domain specification, the cognitive engineering specification and the representation specification as well as a continued development of the meta-knowledge model itself.

5. Summary and conclusions

This paper has attempted to illustrate several points. First, when a development life cycle of representation refinement is utilized, which follows the principles of Newell's knowledge level, then the system development will become self-verifying.

The second issue addressed was the use of a meta-knowledge model. This model allows for a dynamic perspective of the system to be obtained in conjunction with the static aspects found in the other representations at each level.

The use of the knowledge filter to produce a better meta-knowledge model and intermediate representation was introduced along with the meta-knowledge feedback loop to induce the elicitation of further meta-knowledge and, hence, ultimately act as self-verifying mechanism for the intermediate representation.

The paper intends to show that the ability to verify and validate knowledge-based systems is not just a matter of static testing of the knowledge base, but must emanate from a holistic knowledge-level development method incorporating meta-knowledge and a knowledge-level philosophy.

In conclusion, the paper has demonstrated a methodology for the creation of knowledge-based systems that attempts to utilize the rigor of software engineering with the knowledge-level principles to achieve a higher degree of control over the knowledge engineering process. The use of this approach to development should enable better degrees of software quality in knowledge-based systems to be obtained in the future.

I wish to thank all those who have contributed ideas and thoughts to this paper at The Programming Research Group, Oxford University, The Department of Computer Science at Liverpool University, and particularly Dave Woods, Dan O'Leary, Alun Preece for their insights on the Knowledge Level and the Meta-knowledge Debate.

Dedication

This paper is dedicated to the memory of Seymore Cray who died on 5 October 1996.

References

Alexander, J. H., Freiling, M. J., Shulman, S. J., Staley, J. L., Rehfuss, S., & Messick, S. L. (1986). Knowledge level engineering: ontological analysis. *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86)*, pp. 963–968, Philadelphia, PA.

Ansi (1992). *Life Cycle Development of Knowledge Based Systems Using DoD-Std 2167A*, ANSI/AIAA G-031-1992.

Barr, A., Bennett, J. & Clancy, W. (1979). *Transfer of expertise: a theme for AI research*. Report No. HPP-79-11, Heuristic Programming Project, Stanford University, Stanford, USA.

Baughman, D. & Gamble, R. (1996). A methodology to incorporate formal methods in hybrid KBS verification. *International Journal of Human-Computer Studies*, **44,** 213–244.

Brachman, R. J. & Levesque. H. J. (1985). *Readings in Knowledge Representation.* Los Altos, CA: Morgan Kaufman.

Breuker, J. & Wielinga, B. (1987). Use of models in the interpretation of verbal data. In A. L. Kidd, Ed. *Knowledge Acquisition for Expert Systems: A Practical Handbook*. New York: Plenum.

Brown, D. (1982). *MINIMA*. Teknowledge. Inc. Internal Report.

Buchanan, B. G., Barstow, D., Bechtal, R., Bennett, J., Clancy, C., Kulikowski, C., Mitchell, T. & Waterman, D. A. (1983). Constructing an expert system. In F. Hayes-Roth, D. A. Waterman & D. G. Lenart, Eds. *Building Expert Systems.* Reading, MA: Addison-Wesley.

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

Checkland, P. B. (1981). *Systems Thinking, Systems Practice*. Chichester: Wiley.

Churchman, C. W. (1979). *The Systems Approaches and its Enemies*. New York: Basic Books.

Craig, I. D. (1991). *Formal Specification of Advanced AI Architectures*. Chichester: Ellis Horwood.

Davis, R. & Lenat, D. (1982). *Knowledge-based Systems in AI*. New York: McGraw-Hill.

Davis, R. (1980). Meta-rules: reasoning about control, *Artificial Intelligence*, **15,** 179–222.

Davis, R., Shrobe, H. & Szolovits, P. (1993). What is knowledge representation. *AI Magazine,* **14,** 17–33.

Dhaliwal, J. S. & Benbasat, I. (1990). A framework for the comparative evaluation of knowledge acquisition tools and techniques. *Knowledge Acquisition*, **2,** 145–166.

Diaper, D; Ed. (1989). *Knowledge Elicitation: Principles, Techniques and Applications*. Chichester: Ellis Horwood.

Doyle, J. (1980). *A model for deliberation, action and introspection*. Technical Report 581, M.I.T. AI Lab.

Doyle, J. & Patil, R. (1989). Theses of knowledge representation: language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, **48,** 261–297.

ESPRIT Project P5248 KADS-II/DM1.1b/UvA/018/6.0/FINAL.

Gallaire, H. & Lasserre, C. (1982). Metalevel control for logic programs. In K. Clark & S. Tarnlund, Eds. *Logic Programming*, pp. 173–185. New York: Academic Press.

Gamble, R., Roman, G.-C., Ball, W. E. & Cunningham, H. C. (1994). Applying formal verification methods to rule-based programs. *International Journal of Expert Systems*, **7,** 203–239.

Gamble, R. & Landuaer, C. (1995). Validation and verification of KBS, AAAI-95, Workshop Notes, Montreal, Quebec, Canada.

Gamble, R. & Shaft, T. (1996). Eliminating concerns for redundancy, conflict and completeness in knowledge based systems. *International Journal of Software Engineering and Knowledge Engineering*, **6**.

Genesereth, M. R. & Smith, D. E. (1982). *Meta-level architecture*. HPP-81-6, Stanford University Heuristic Programming Project, December.

Gigch, J. P. van (1984). Epistemological questions raised by the meta system paradigm. *International Journal of Man–Machine Studies*, **20,** 501–509.

Gold, D. I. & Plant, R. T. (1994). Towards the formal specification of an expert system. *International Journal of Intelligent Systems*, **9,** 739–768.

Grover, M. D. (1983). A pragmatic knowledge acquisition methodology. *Proceedings of the* 8th *International Joint Conference on Artificial Intelligence (IJCAI-83)*, pp. 436–438.

Hayes, P. (1973). Computation and deductions. *Proceedings of the Symposium on Mathematical Foundations of Computer Science*, pp. 105–117, Czechosloviakian Academy of Sciences.

Hesketh, P. & Barett, T. (1990). *An introduction to the KADS methodology*. Esprit Project P1098, Deliverable M1, EEC ESPRIT Project, Brussels, Belgium.

Hollnagel, E. (1992). *Reliability of cognition: foundations of human reliability analysis*. CRI A/S, Denmark.

Jacob, R. J. K. (1983). Using formal specifications in the design of a human–computer interface. *Communications of the ACM*, **26**.

Johnson, G. I. & Westwater, M. G. (1996). Useability and selfservice information technology—cognitive engineering in product design and evaluation. *AT&T Technical Journal*, **75,** 64–73.

Jones, C. (1980). *Software Development a Rigorous Approach*. Englewood Cliffs. NJ: Prentice-Hall.

JONES, K. (1995). *Focusing software requirements through rapid prototyping*. MEng. Dissertation, Royal Military College of Canada, Canada.

LASSEZ, J. L. & MAHER, M. (1983). The denotational semantics of Horn clauses as a production system. *Proceedings of the 2nd Natinal Conference on Artificial Intelligence (AAAI-83)*, pp. 229–231, Washington, DC, USA.

MADAN, M. (1995). *Evaluating alternative approaches to knowledge acquisition for expert systems*. Ph.D. Thesis, University of Calgary. Canada.

MANCINI, G., WOODS, D. D. & HOLLNAGEL, E. (Eds). (1988). *Cognitive Engineering in Dynamic Worlds*. New York: Academic Press.

MCCARTHY, J. (1979). First order theories of individual concepts and propositions. In J. HAYES, D. MITCHIE & I. MIKULICH, Eds. *Machine Intelligence 9*, pp. 120–147. Chichester: Ellis Horwood.

MILLER, L. (1990). *A realistic industrial strength life cycle model for knowledge-based system development and testing*. AAAI Workshop Notes: Validation and Verification, August, Boston, MA.

MORELL, L. J. (1989). *Use of metaknowledge in the verification of knowledge-based systems*. NASA Contractor Report 181821, Langley Research Center, Virginia, USA.

MURRELL, S., PLANT, R. T. & GAMBLE, R. (1996). Formal specification of rule-based systems through styles. *AAAI Workshop Notes: 9th Workshop on Validation & Verification of Knowledge-Based Systems*. Portland, Oregon, August.

MURRELL, S. & PLANT, R. T. (1995). Formal semantics for rule-based systems. *Journal of Systems and Software*, **29,** 251.

NEWELL, A. (1982). The knowledge-level. *Artificial Intelligence*, **18,** 87–127.

NORMAN, D. A. (1981). *Steps towards a cognitive engineering*. Technical Report, University of California at San Diego, Program in Cognitive Sciences, CA, USA.

O'LEARY, D. E., Ed. (1994). Special issue: verification and validation of intelligent systems: five years of AAAI Workshops. *International Journal of Intelligent Systems*, **9**.

PLANT, R. T. & PREECE, A. D. (1996). Editorial: special issue on verification and validation of KBS. *International Journal of Human-Computer Studies*, **44,** 123–125.

PLANT, R. T. (1994). *Validation and verification of KBS*. AAA-94 Workshop Notes, Seattle, WA, USA.

PLANT, R. T. (1994). *Methodologies for the development of knowledge-based systems*. Working Paper, Department of CIS, University of Miami, Coral Gables, FL33124, USA.

PLANT, R. T. (1991). A rigorous approach to the development of knowledge-based systems. *Knowledge-based Systems*, **4,** 186–197.

PLANT, R. T. & MORENO, H. R. (1994). *A survey of knowledge acquisition techniques*. Working paper, Department of CIS, University of Miami, Coral Gables, FL, USA.

PREECE, A. D. (1995). Towards a quality assessment framework for knowledge-based systems. *Journal of Systems and Software*, **29,** 219–234.

PREECE, A. D. (1993). *Validation and verification of KBS*. AAA-93 Workshop Notes, Washington, DC, USA.

PREECE, A. D., GROGONO, P. & SHINGHAL, R. (1993). Assessing the capability of knowledge-based system developers. *IEEE CAIA Workshop on Validation and Verification*, Orlando, FL, USA.

RAGAN, S. L. (1983). Alignment and conversational coherence. In R.T. CRAIG & K. TRACY, Eds. *Conversational Coherence: Form, Structure, and Strategy*. Beverley Hills, CA: Sage.

ROTH, E. M. & WOODS, D. D. (1989). Cognitive task analysis: an approach to knowledge acquisition for intelligent system design. In G. GUIDA & C. TASSO, Eds. *Topics in Expert System Design*, pp. 233–264. Amsterdam: Elsevier.

SANDEWALL, E. (1975). *Ideas about management of LISP databases*. Working Paper 86, M.I.T. AI Lab.

SCOTT, C. (1991). *A Practical Guide to Knowledge Acquisition*. Reading, MA: Addison-Wesley.

SCHMOLZE, J. & VERMESAN, A. (1996). *Validation and verification of KBS*. AAAI-96 Workshop Notes.

SPIVEY, J. M. (1990). *The Z Notation*. Englewood Cliffs, NJ: Prentice-Hall.

SUFRIN, B. A. & HE, J. (1990). Specification, analysis and refinement of interactive processes. In M. HARRISON & H. THIMBLEBY, Eds. *Formal Methods in Human–Computer Interaction*. pp. 153–200. Cambridge: Cambridge University Press.

WARREN, D. (1981). Efficient processing of interactive relational database queries expressed in logic. *Proceedings of the 7th VLDB Conferences.*

WATERMAN, D. A. (1986). *A Guide to Expert Systems*. Reading, MA: Addison-Wesley.

WELBANK, M. (1983). *A review of knowledge acquisition techniques for expert systems*. British Telecom Research Labs. Martlesham Consultancy Services.

WEYHRAUCH, R. W. (1980). Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence*, **13,** 133–170

WISE, J., HOPKINS, D. V. & SAGER, P. (1993). Verification and validation of complex man–machine systems. NATO Advanced Study Institute, Vimero, Portugal. Berlin: Springer.

WOODS, D. D. & HOLLNAGEL, E. (1987). Mapping cognitive demands in complex problem solving worlds. *International Journal of Man–Machine Studies*, **26,** 257–275.

WOODS, D. D. & ROTH, E. M. (1988). Cognitive systems engineering. In M. HELANDER, Ed. *Handbook of Human–Computer Interaction*. New York: North-Holland.

Paper accepted for publication by Editor, B. R. Gaines.

Appendix

TRILLIUM$_K$ level 1 capability	
Problem specification	No explicit statement of requirements. No test plan. No acceptance criteria
Conceptual model	No documented conceptual model
Design model	No documented design model for knowledge base. Typically, a commercial shell is used; the reasons for choosing this shell should be documented
hline implemented model	Implemented knowledge base is the only complete description of knowledge. Inference engine is typically that of an existing expert system shell
Verification analyses	Verification performed by informal proofreading—no formal verification analysis conducted
Validation analyses	Validation performed by *ad hoc* testing and informal evaluations. No permanent recording to test suite

TRILLIUM$_K$ level 2 capability	
Problem specification	Informal statement of requirements, test plan, and acceptance criteria
Conceptual model	"Paper model" stated semiformal. Separation of concerns achieved by isolating domain, task, and cooperative knowledge components

Design model	Architectural design for system components. Semiformal or formal designs for procedural parts of knowledge base, and for inference engine
Implemented system	Implemented knowledge base and inference engine is traceable, where appropriate, to conceptual and design models. When a third party shell is used, ensure that its behavior conforms to that required
Verification analyses	Knowledge base integrity and expression logic is checked automatically, with all detected anomalies fully documented and resolved
Validation analyses	Testing using documented test suite is performed according to problem specification. Semiformal evaluations of system useability are performed and documented
	TRILLIUM$_K$ level 3 capability
Problem specification	Semiformal statement of requirements, including minimum and desired functionality. Formal constraints should be associated with all possible minimum requirements. Test plan and acceptance criteria associated with each functional requirement that cannot be verified formally
Conceptual model	Formal knowledge-level model (that is, with well-defined syntax and semantics). Appropriate representation languages used for domain, task and cooperative knowledge base components
Design model	Formal architectural design, module-interface specifications, and internal module designs for all system components, including interface engine, domain knowledge modules, task knowledge modules, meta-level control knowledge modules and external interface components
Implemented system	Implemented system is fully traceable to conceptual and design models or is derived automatically from them using a correctness-preserving transformation procedure. A third-party tool may be used only if rigorous assurances are available of its correctness and reliability
Verification analyses	Full inference logic is checked and all anomalies documented and resolved. System compliance with all minimum constraints is verified and documented if possible
Validation analyses	Rigorous structural and functional testing is performed according to problem specification. Test suite is executed and maintained using support tools. Approved empirical methods are used for usability and utility evaluations, and results are fully documented