The Knowledge Engineering Review, Vol 7:2, 1992, 101–113

Overview of current practice and research initiatives for the verification and validation of KBS

T. J. LYDIARD

Artificial Intelligence Applications Institute, University of Edinburgh, Edinburgh, UK

Abstract

Based on a survey of recent literature, this report aims to highlight the issues associated with the verification and validation of knowledge based systems. The confusion arising from the lack of clear terminology is considered, along with some of the characteristics of knowledge based systems that cause particular difficulties for verification and validation. The various approaches that can be adopted to address these difficulties are discussed, followed by a survey of recent research initiatives.

The author concludes that many of the difficulties associated with the verification and validation of knowledge based systems are a feature of the complexity of the system being built and the manner of its development rather than of the specific technology chosen to implement it.

1 Introduction

Knowledge based system reliability is currently one of the most important issues concerning knowledge based systems (KBS) technology. The widespread recognition of the importance of KBS reliability stems from both the need to move away from the development of small prototype systems towards larger real-world KBS applications, and from the increasing requirement to embed them in larger conventional systems, particularly those operating in a real-time context. Research into the field of verification and validation of KBS applications is intended to address issues associated with quality assurance of KBS applications, and to credit such applications with the same degree of reliability as conventional applications.

One measure of the quality of software is that the system performs its functions as intended by its specifiers (McGraw & Harbison-Briggs, 1989). This statement is based on the assumptions that the intentions are fully understood by the system specifiers, and that mistakes are not introduced during implementation. However, the more complex the problem is, the harder it is to understand the exact nature of the requirements of the clients, the more mistakes are likely to be introduced, and the more severe the nature of those mistakes. Verification and validation techniques provide a way of checking that the intentions of the clients are not misunderstood at the requirements specification, design or implementation phases.

The purpose of this report is to highlight the main issues associated with the verification and validation of knowledge based systems. These issues include discussion of the main terminology used and identification of the various components of verification and validation. The characteristics of KBS which pose special problems for verification and validation are discussed, along with some of the techniques currently used to overcome these problems together with details of tool support. Finally, some of the major research initiatives into verification and validation of knowledge based systems are outlined.

2 Terminology

Verification and validation form part of the overall quality assurance process (Born, 1988). However, there is much confusion about the difference between verification, validation and

evaluation, with the terms often being used interchangeably. As so often happens in relatively new fields of research, there is a lack of standardization of terminology, with each author re-defining the terms, and this makes comparison and classification of the various approaches difficult (Hoppe & Meseguer 1991). The most commonly accepted definitions of the various terms are presented below.

Verification can be summed up as 'Are we building the product right?' More formally, verification concerns the properties of the software development process. That means that verification is carried out on the process of software development. Essentially, verification is an activity which should ensure that the product of one phase of the life-cycle is consistent with itself and with the source from which it was derived. As such verification should be carried out at the end of each phase of development.

Validation can be summed up as 'Are we building the right product?' Validation is the process of evaluating software at the end of each development phase to ensure that it complies with the software requirements it was intended to satisfy. In other words, validation is carried out on the product.

Essentially, validation is an activity which should ensure that products of one phase conform with the user needs and requirements. In particular, validation should show that the final results conform to the stated requirements, and that the contents of the knowledge base conform to the current real-world expertise.

Validation is usually achieved through testing, which is a conformation technique which ensures that the behaviour is in accordance with pre-established criteria. Testing should be performed frequently during the development to ensure the quality of the results of the KBS. However, it is often carried out as an independent phase, and is frequently performed by the user.

Evaluation is a feature of both verification and validation, and concerns the assessment of the quantitative and qualitative characteristics of the KBS application through comparison with required standards. These characteristics may be the features of the inference engine, or the overall structure, utility and usability of the KBS. The main drawback to this type of quality assurance is that standards are difficult to formulate for KBS applications.

These definitions are close to those used within the software engineering community. However, the fundamental nature of KBS software requires slightly different components of the developing system to be considered and the traditional software engineering definitions of verification and validation may need to be tailored to accommodate them. These components are out-lined in the following section.

3 Components of verification and validation

The emphasis of verification and validation for KBS has been placed mainly on performance issues, such as the percentage of problems that the knowledge base can be used to solve correctly and the time taken to solve them. This emphasis is concerned principally with the functional requirements of the KBS, or the specification of the expected behaviour of the system. However, good performance may mask a host of other faults within the system such as a lack of clarity of the knowledge base itself which can create problems for modification and provision of explanation, etc. This may mean that the structure and form of the knowledge base does not reflect the manner in which the expert thinks about the domain, which may cause problems in the direct verification of the knowledge base by the expert (van Someren, 1991).

A KBS possesses a number of these non-functional requirements which include: the logical consistency of the knowledge; redundancy; efficiency; comprehensibility; its quality of discourse in user interactions and its general usefulness. These non-functional requirements should be specified and should be quantifiable in some way, so that they can be considered in verification, validation and evaluation.

Some of the major components to verification and validation are outlined below. Some of these components are solely concerned with verification of the knowledge or of the inference mechanism. Others concern aspects of validation, such as whether the requirements and expectations of the user have been met. The remainder are characteristics which need to be evaluated.

competency This characteristic deals with the quality of the source of the knowledge. It can be assessed by comparing the quality of the system's decisions and advice with that obtained from sources of knowledge other than the primary expert.

completeness It is important to check that the knowledge base is complete, that is that the system can deal with all reasonably expected situations within its domain. This involves ensuring that all the knowledge is referenced, that there is no attempt to access knowledge or data that does not exist and that, in the case of goal-driven reasoning, all the goals can be reached.

consistency The knowledge base must produce similar answers to similar questions. The knowledge contained within the knowledge base must not be contradictory in any way.

correctness The knowledge within the knowledge base should be 100% correct. However, even human experts make mistakes, or differ in their judgement of a particular case, so some measure of adequacy rather than correctness should be sought. This may include an estimate of the cost of each mistake and of how often mistakes are made.

efficiency The speed of system response is closely linked to the amount of search (and hence to the efficiency of the search algorithm) and to the volume and manner of representation of the data in the system. For each application it is important to ascertain the efficiency requirement of the system in terms of response time and memory usage. Consideration must be given to the type of search strategy which is most appropriate to the particular problem, for example should the system be goal-driven or data-driven, use depth-first or breadth-first search, and should it find any path, or find the optimal path? The manner in which the search strategy is implemented in the chosen software and the choice of a knowledge representation to facilitate this are also important issues.

maintainability This involves considering what will happen to the implemented system, for example who will be responsible for its enhancement and maintenance. This may affect decisions on the use of particular knowledge representations, and concerning the general readability of the knowledge base. The principal concern should be whether the knowledge base is understandable to people other than the original knowledge engineer and expert, in particular, whether another knowledge engineer could maintain it if the need arises.

performance Often used interchangeably with efficiency, performance is less concerned with speed of execution than with whether the system functions as expected. Performance also incorporates the idea of reliability–whether the system behaves in a predictable manner.

structure These types of errors can be detected through systematic analysis of the knowledge base. In rule-based systems checks for the following errors can be made relatively easily:

- redundant knowledge–where two condition statments are equivalent and one or more of their conclusions are the same;
- conflicting knowledge–where two condition statements are equivalent and one or more of their conclusions conflict;
- subsumed knowledge–where the conclusions of two pieces of knowledge are equivalent but in one the condition statements are fewer;
- circular rules–where the actions of one piece of knowledge lead back directly or indirectly to the condition statement of that same piece of knowledge.

testability The system should be designed in such a way as to permit a sensible testing plan to be carried out. The success rate of the KBS depends to some extent on the number and coverage of the

test cases used or the phase of development at which testing is started, and on the bias of the tester (O'Keefe *et al.*, 1987).

usability This concerns the utility of the system in a social and organizational context, that is the extent to which the KBS proves useful in practice in the environment in which it is supposed to be used. Important factors here are good ergonomic design, I/O design, flexible dialogue structure, help and explanation facilities, and robustness (Mengshoel, 1991).

4 Problems with the verification and validation of KBS

The boundary between conventional applications and KBS is becoming somewhat blurred as more and more complex conventional applications are built. The use of KBS software has facilitated the development of these complex advanced information systems. The complexity tends to be introduced through three areas, the characteristics of which have a pronounced effect on the verification and validation of the finished system. These are:

- input—Is there any uncertainty attached to the input data?
- the problem-solving strategy—Is the task that the system is intended to carry out entirely new or does it exist already? If it exists, does it exist in a readily observable form, or does it exist implicitly in the expertise of one or more people. If more than one expert is involved, what is the potential for conflict between them? How stable is the task? How complex is the task and to what degree must the system interact with other agents?
- the solutions—Can all the solutions be predicted or are they too numerous and/or too novel? How should solutions be graded in terms of correctness, optimality and accuracy?

If the solutions can be predicted and enumerated then verification and validation can be carried out using the performance of the system as the main criterion. However, if the solutions cannot be predicted or enumerated then extra attention must be paid to verification and validation of the input data and of the problem-solving processes.

KBS also pose special difficulties; these reflect the differences in the way in which they are built rather than fundamental differences in the types of applications being tackled.

The most commonly preferred conventional approach to software development follows the waterfall life-cycle method, so called because the design process moves in only one direction: forward, from one stage to the next. In the classical waterfall method, the various stages of development are seen as discrete phases in the life-cycle; the start of one phase is driven by the successful completion of the previous phase. Each phase should only be visited once during the life-cycle, although there is some provision for feedback between successive phases. Verification tends to occur at the end of each phase, and validation is added as a distinct phase at the end of the development period. This method works well when the nature of the system to be implemented is explicit, well-understood and, more importantly, when the original requirements specification and analysis are complete, accurate and stable.

However when developing KBS it is difficult to work in such a constrained way. The system to be implemented is typically not well-understood, may be very context-sensitive, and may contain significant amounts of complexity in its behaviour and uncertainty in its data. All these factors combine to make the production of complete and accurate analysis and design difficult. Such systems tend to be developed in an iterative or incremental manner. This approach creates a number of problems with respect to verification and validation. In particular, incremental development all too often results in the undisciplined production of undocumented, unstructured code which, because it lacks a suitable specification, is impossible to validate. Additionally, any assumptions and decisions made in the design and implementation of the system may not be recorded, and this makes subsequent maintenance and enhancement of the system difficult.

Recent attempts to bring more discipline to incremental development have resulted in the adoption of spiral life-cycle models. Spiral models are, like the waterfall model, based on

sequences of phases; in them, however, the same sequence may be repeated a number of times during the life-cycle. Each pass through the sequence of phases equates with one turn of a spiral. Spiral models tend to specify management activities rather than technical activities. The spiral model includes risk management activities to identify, assess and reduce the risk associated with the lack of a clear specification, with the use of new software, and with the complexity and uncertainty inherent within the application itself. Quality assurance is usually carried out as part of a review phase during each iteration of the spiral; it may involve verification and validation of all products affected by the developments made in that iteration.

While inadequate specifications are quite common within conventional systems development, they are even more common for KBS development. In general, the information on how to carry out significant portions of a task that a KBS is intended to automate or support is contained within the skills of current practitioners. The task may involve making judgements or interpretations which have never been explicitly recognized, on the basis of criteria which have never been characterized. It is often not even possible to determine reliably which information is potentially relevant, let alone how it is to be combined, this obviously makes it very much more difficult to produce a clear specification of how a KBS ought to behave. Lack of this clear specification is the cause of most of the problems relating specifically to verification and validation of KBS.

From this section it can be seen that there are two main areas presenting particular difficulties in the verification and validation of KBS, namely the lack of a clear specification and the problems in determining when to carry out verification and validation procedures.

5 Approaches to verification and validation of KBS

Opportunities to carry out verification and validation occur at various stages during the development of the KBS. If the waterfall approach to software development is used verification and validation can be carried out at the end of each phase of development. Verification and validation are frequently carried out on a module by module basis, and then on the system in its entirety.

Using an incremental approach to development may create difficulties with verification and validation. If new functionality is added to the existing system rather than developed in isolation, then the entire system must be verified and validated at each iteration of the development process. Unless care is taken as the system grows problems which could not be anticipated at earlier stages may be encountered; these include system inefficiency and incompatibility of software features. The problems encountered may be so severe as to require a redesign of the underlying structure of the whole system.

There has been some research effort into and development of a number of techniques to support verification and validation of KBS, and in identifying the most appropriate phases of software development to which they may be applied. Some of these techniques are outlined below.

5.1 Knowledge acquisition methods

The choice of knowledge acquisition methods should support the verification required (McGraw & Harbison-Briggs, 1989). However, inclusion of verification in the knowledge acquisition process itself can change its nature (Nazareth, 1989).

Attention should be given to the production of an adequate knowledge acquisition plan, to the knowledge elicitation sessions, and to the post-session analyses. A simple form of verification can be carried out by supplying the expert with transcribed and extracted information gathered from the knowledge elicitation sessions for review. As a result of identification of inaccuracies or omissions, the knowledge acquisition plan can be refined to reflect the appropriate sequence for the elicitation of specific information. An audit trail can also be maintained as to the source and context of the information going into the knowledge base.

Two approaches have been adopted to the verification process at this stage., The first approach includes verification of the domain knowledge itself, whereby the correctness of the expert's

knowledge in the target domain is verified (Enand *et al.*, 1990). Serious errors can occur if the application problem domain is not analyzed thoroughly, or if the expert has limited knowledge; these can lead to vague or incorrect solutions being delivered by the system. This risk can be dealt with to some extent by using multiple experts. If the domain can be divided into a number of specialized fields or sub-domains, then separate experts, each competent in a particular field can be used to complement each other. Alternatively, several experts within a single domain can be used. This can provide alternative opinions and can help to identify weaknesses in following a single line of reasoning.

The second approach to verification of acquired knowledge involves the explicit exclusion of domain knowledge. This view has attracted more interest as it allows the development of more generic principles concerning consistency and completeness issues (Nazareth, 1989).

5.2 Intermediate knowledge representations

In the early days of KBS development, coding used to begin as soon as the first knowledge elicitation session had finished. More recently, the trend has been to adopt a more systematic, controlled and methodical approach to building KBS. The real-world expertise is analysed and gradually transformed into a system implementation via a series of intermediate representations or models. This structured 'knowledge modelling' approach results in a series of documents from requirements specification to detailed designs, allowing more traditional verification to take place. A number of commercially available methodologies have been developed which support this approach (Inder & Filby, 1991), and research continues into the sequence of models to develop and the notations and formalisms used to express them.

5.3 Code checking

As a consequence of the early approach to KBS development which resulted in a tendency to produce executable code much earlier in the product development life-cycle, verification of KBS has centred around the development of tools for checking for the completeness and coherence of the system code. The majority of these code-checking tools have been developed primarily for rule-based systems, although as yet few are commercially available. These are outlined in section 6. The main function of these code-checkers is the evaluation of the internal properties of the artifact, i.e. they check for missing rules, simultaneously satisfied rules, and redundant rules. The more sophisticated checkers can also detect subsumed rules, unnecessary conditions, directly contradictory rules, unreferenced attribute values, unsatisfiable goals and unusable conclusions. Many of these systems transform the rules into an alternative format allowing more efficient detection of errors. The most commonly tried approaches are transformation into various types of graph (Nazareth and Kennedy, 1991), petri nets (Liu & Dillon, 1991), decision tables (Cragun and Steudel, 1987) or flow networks.

These code checking systems have a number of limitations. For example, their ability to detect contradictory rules is limited to situations where pairs of rules with similar premises assert conclusions which are direct negations of one another. They cannot handle what seems certain to be a much more common (and troublesome) situation, which occurs when rules assert things which allow a contradiction to be deduced. Also, these systems perform entirely static checks on the rule basis, and have nothing to say about the dynamic behaviour of the system. However, the development of these code-checking tools appears to be step in the right direction.

Tools for code-checking provide no measure of the quality of design in terms of its clarity, extensibility, likely stability or in the efficiency of design in system, These factors not only influence the development time required but also effect the usefulness and lifespan of the completed system. For less complex systems, these types of factors would be reviewed via the documentation produced at the end of each development phase. For KBS and complex systems these factors should be checked at each iteration through the development phase.

5.4 Static analysis tools

In addition to code-checkers there are a number of other tools which aid in performing static analysis. These include syntax checkers, cross-reference generators, structure-checkers and type analysers. Commonly, a subset of these tools is provided within the development environment of KBS tools. However, these tools are seen primarily as deugging tools for the developers, and little or no use is made of them for verification purposes.

5.5 Testing

Although many of the techniques used for testing conventional software can be used, KBSs pose a number of problems for validation. Due to the iterative nature of KBS development, validation procedures ought to be performed throughout the development process. However, the formation of test plans in the project planning stage can be difficult due to vagueness of the requirements specification and design.

The number of paths through a large KBS makes exhaustive testing impossible. If the system has been designed and implemented in a modular fashion then some form of unit testing might be possible. However, the modules are not usually designed to function in isolation so a test harness must be developed for each module, considerably increasing the development time. Testing in isolation also carries the risk of false errors produced from invalid combinations of data which might be detected at input stage or in a preceding module, and which do not appear when testing the complete system. As stated earlier, for large KBS applications exhaustive system testing is impractical. One way around this has been to test specific facets of the KBS such as the most frequently used paths, the particular elements of the knowledge base which are most critical in deriving conclusions, and all the integration points between knowledge bases, databases and external programs. Of these, testing the most sensitive path, that is the critical elements used to drive the conclusions, has been found to be the most effective (Ribar *et al.*, 1991).

5.6 Source of test data

Another problem is the source of the test data. In some cases, historical data can be obtained and the output of a test run of the KBS application can be compared to the output associated with the historical data. However, in most cases the test data has to be generated, either by the expert or using automatic test data generating programs. These programs would also have to be written specifically for each application, again adding to the overall development time. In addition the output of any test data would have to be analysed and deciding how to judge this can be difficult.

5.7 Adequacy of solutions

Specific to the validation of KBS is the idea of the adequacy of the solutions (Partridge, 1986). The kinds of information and responses involved with a KBS make it very difficult for the layman to assess the correctness of the solutions. This is best handled by the expert himself, with the solutions judged as 'ideal', 'acceptable', 'suboptimal' or 'unacceptable' rather than right or wrong. There is an additional complication here in that different experts may differ in their solutions to a particular problem, and even the same expert may reach different conclusions on different days. Therefore, validation should be carried out by the expert who has responsibility for the actions taken on the basis of the advice offered by the system.

The problems associated with judging the adequacy of the solutions highlights the need to establish suitable 'acceptable criteria' at the start of the system development. Assessment of solutions may vary according to the bias of the assessors. Developers and sponsors of the system generally assess it favourably,whereas other members of management who have not been involved in its development and may feel threatened by the technology, may assess the system unfavourably.

The best assessment is whether the introduction of the system makes any improvement to the business area it is supposed to support, or to the enviroment of the people who work there.

Despite the work that is being done in this area there are still a number of issues which are not being addressed by current verificiation and validation procedures. These issues are mostly concerned with monitoring and reviewing the quality of the knowledge acquired and include issues such as monitoring the effectiveness of the knowledge engineer, the relationship between the knowledge engineer and the domain expert, and the form of the knowledge acquisition sessions.

6 Tool support for verification and validation of KBS

Most of the tools developed to support verification and validation are code-checking tools, and the majority of them are in academic use only.

Within the UK work has been done on checking rule bases for missing rules, redundant rules and simultaneously satisfied rules. This is summarized by Johnson *et al.* (1987). In the system they developed, rules written for the expert system shells Sage and Xi are translated into a decision table format, in which form it is accessible to the checking techniques they developed in the late 1960s.

There are more sophisticated systems being developed, such as CHECK (Nguyon *et al.*, 1987; Nguyon, 1987) and EVA (Stachowitz & Combs, 1987; Stachowitz *et al.*, 1987). CHECK works primarily with rules for an expert system tool developed by Lockheed, although it has been extended to deal with ART rules. It can detect about a dozen inter-rule and rule to fact relation which are likely to indicate errors or omissions. These include subsumed rules, unnecessary conditions, directly contradictory rules, unreferenced attribute values and obviously unsatisfiable goals or unusable conclusions.

Like CHECK, EVA was also developed by Lockheed, and can detect obviously unsatisfiable goals and unusable conclusions, duplicated and subsumed rules, rule cycles and unnecesary patterns in rules. However, EVA is enhanced by the ability to handle programmer-specified meta-information, such as indications of class to sub-class relationships, synonymy, and ranges of legal values for attributes. Class to sub-class information is taken into account in checking for duplicated and subsumed rules, stated incompatibilities are used for checking the consistency of the patterns within a rule and relations can have restrictions imposed on the kinds of entity they can relate. Attributes can be given sets of legal values or numeric ranges, and this information will be used to detect cases which are not dealt with.

Another tool, Validator, developed by Jafar as part of his PhD work (Kang & Bahill, 1990), is designed to be used interactively at runtime to verify and validate rule-based KBSs. This tool consists of six modules: a pre-processor; syntax analyser; syntactic error checker; debugger; chaining thread tracer and a KB completeness module. Validator can detect nine categories of mistakes including unused rules, unused data and subsumed rules.

More recently, the Knowledge Integrity Checker (KIC) (Pearce, 1991) had been developed by the Turing Institute. KIC is designed to perform validation on a knowledge base represented in a logic programming language. KIC is claimed to be able to detect type conflicts and a number of syntactic, semantic and structural inconsistencies in the knowledge base such as unreachable rules, dead-end rules, cyclic rules, incompleteness and subsumption.

The C-based production rule language CLIPS, developed by NASA, has an integral basic rule-checker known as the Cross-Reference, Style and Verification utility (CRSV). This tool consists of three main components. The rule verification and relation cross-referencing utilities rely on the provision of user-defined standards. For rule verification, CRSV compares each pattern in each rule with the user-defined standards and where a discrepancy is found, an error message is generated. In the case of cross-referencing, relations used within the rules are compared to definitions and a list of how and where each relation is used within the knowledge base is generated. The style-checker looks for poor coding or improper technique. Although basic, the CRSV is provided as part of the CLIPS package to provide some aid in the process of verifying that the rules within the knowledge base are consistent with a user-defined set of standards.

Also available commercially is KnAcq, a knowledge acquisition toolkit produced by Common Knowledge Ltd. The primary aim of the tool is to drive the knowledge acquisition process via the production of a special diagram called an exception graph. This graph can be automatically transformed into rule-sets for a number of commercial KBS tools such as Nexpert object, Crystal and Xi-Plus, with the claim made that the rule-sets are guaranteed correct, as no intermediate modification has occurred; complete, due to the directed questioning initiated by the KnAcq tool; and consistent, due to the manner in which the rules are generated. KnAcq is designed to aid rapid construction of a KBS and to provide support for thorough and reliable maintenance of the KB.

7 Research initiatives

7.1 European perspective

7.1.1 UK JFIT projects

The projects listed below have taken place under the Information Engineering Advanced Technology Programme (IEATP) of the UK's Joint Framework for Information Technology (JFIT). This programme is jointly funded and managed by the UK's Department of Trade and Industry (DTI) and Science and Engineering Research Council (SERC). The majority of the projects fall within the Systems Engineering technical area.

Integrated Fault Management Environment Work at the Turing Institute has involved the automatic generation of shallow rule sets from more complex qualitative models. This is done for diagnostic systems by producing a complete table of cause and effects which is subsequently used for inductive rule generation. The inductive inference processes compress the data in the table producing an efficient rule set which directly associates simulated effects with failures. This shallow rule-set can then be passed to a tool such as the Knowledge Integrity Checker (see section 6) to identify errors (Pearce, 1991). This work is part of a larger Integrated Fault Management Enviroment (IFME) project involving a consortium of the Turing Institute, British Aerospace Space Systems Ltd., British Aerospace (Military Aircraft) Ltd., and AEA Technology. The project (project number IED4/11561), which is of 30 months' duration, was started in February 1990.

Development of a Methodology for Hybrid Knowledge-Based/Conventional Systems The aim of this project (project number IED4/1/1426) is to develop an abstract model of methods which allows conventional data processing methods and KBS methods to be integrated into a single framework and then tailored to the requirements of individual system development projects. Hybrid metrics are also being developed to assist in cost estimation and quality control. A number of tools are under development to support the selection and configuration of methods and to support the use of metrics. The project, which is of three years' duration, started in February 1990. The participants are BIS Information Systems Ltd., Expert Systems Ltd., and Aston University.

GATEWAY This was a research project into the metrication of KBS projects. Its principal aims were to develop metrics for the evaluation of KBS and KBS projects in terms of the possible benefits, costs, risks and quality of the resulting system. The project attempted to apply these metrics on KBS projects and to develop prototype KBS metrics support tools. The main participants in the project (project number IED4/11751) were Logica Cambridge Ltd., Integral Solutions Ltd., and Rutherford Appleton Laboratories. The project ran for two years from April 1990 to March 1992.

7.1.2 ESPRIT projects

The following projects have taken place under the European Strategic Programme for Research and Development in Information Technology (ESPRIT).

ACKNOWLEDGE This project (project number 2576) was concerned with efficient aquisition of knowledge. At its centre was the production of an integrated environment for knowledge acquisition, the knowledge engineering workbench (KEW). KEW incorporates a number of automated knowledge acquisition tools. Validation was considered in the context of knowledge acquisition methods and techniques, of knowledge transformation and integration, and with respect to generic problem-solving techniques. Validation tools included KVAT, which was developed by Sintef Delab (Norway). KVAT validates knowledge in the core knowledge base in frames expressed in Sintef's own frame language. The chain of reasoning used by the knowledge base and the solutions reached are compared against those produced by the expert, relieving the knowledge engineer of the need to perform manual walkthroughs.

The project ran from January 1989 to December 1991. It was co-ordinated by Cap Gemini Innovation (France) and there were seven partners, including Marconi Command and Control Systems Ltd (UK) and the University of Nottingham (UK).

COMPLEMENT COMPLEMENT is an ESPRIT project (project number 5409) entitled 'Comprehensive Large-Scale Engineering Methodologies and Training'. The objectives of the first of the first phase of this project are to study and assess the current practices, methods, techniques and tools applied by companies to the development and validation of complex systems. This will be followed by a second phase in which the definition, development and evaluation of new methodologies will be undertaken, with a view to fitting the largest set of industrial requirements such as real-time aspects.

The project, which is of three years' duration, started in October 1990. The project is co-ordinated by Syseca Temps Reel (France).

KADS-II The aim of KADS-II is to provide an advanced and comprehensive methodology for integrated KBS development. It is based on the work carried out on KADS-I (project number 1098), extending the current KADS methodology to cover the full development life cycle, from project definition through system development to the subsequent maintenance of the system. The consortium will also be producing a standard for KADS known as Common KADS. Greater emphasis will be given to quality control, hence techniques for validation and verification are expected to be developed.

The KADS-II project (project number 5248), started in October 1990 and will run for four years. There are over ten participants in the project, which is co-ordinated by Cap Gemini Innovation (France).

VALID The VALID project (project number 2148) is concerned with validation methods and tools for knowledge based systems. The objective of the project is to develop a tool-box based on a common representation within which validation concepts can be defined and analysed. There are several strands to the project, which provide the basis for several software products. They are outlined below:

- the definition of a KBS quality manual, with special attention to validation action during the different phases of KBS life-cycles;
- a general model in which validation issues can be defined;
- a common conceptual representation (CCR) in which most KBSs can be expressed;
- the definition of a validation metalanguage (VETA) to provide the abstract mechanisms necessary to build validation tools.

Three validation tools are currently under development as part of this project. WITNESS manages the comments from the validators of the knowledge base system, CURRICULUM CV is a query language for the use by the knowledge engineer when inspecting the knowledge base and CONTROLLER checks the integrity constraints associated with the input and output data, and of the input with respect to the output (Petitjean *et al.*, 1991).

The project co-ordinator is Cognitech SA (France), and there are three other partners. The project started in December 1988 and runs for three years.

VITAL The main aim of the VITAL project (project number 5365) is the construction of a workbench which should provide an integrated project support environment for all aspects of the development and maintenance of KBS and of the management of the KBS project. VITAL aims to provide methodological support for developing large, industrial, embedded KBS applications through the use of this workbench. The VITAL methodology and workbench will include support for verification and validation.

There are nine participants in the VITAL consortium which is co-ordinated by Syseca Temps Reel (france). The project which started in September 1990 is of four and half years' duration.

VIVA VIVA (Verification, Improvement and Validation of Knowledge-Based Systems) is the only project on KBS validation within the new ESPRIT III Programme. The VIVA project (project number 6125) will start in June 1991 and will run for three years.

The project aims to produce and deliver a comprehensive tool-set comprising both adaptations of existing validation tools selected from the tools previously developed by the members of the consortium (tools from the ESPRIT II project VALID—SACCO & SYCOJET (LIA), VORTEX (Logica), KRUST (University of Aberdeen)) and newly developed tools. This tool-set will include tools that allow off-line validation (static and dynamic checking), on-line validation (testing), as well as to remove interpretation (finding the causes of exhibited anomalies) and refinement (changing the KB to remove anomalous effects). The project will give prominence to a specific kind of knowledge, called the validation knowledge, which is needed by some of the tools for performing validation tasks (as in the tools SACCO and SYCOJET). The project will also produce and deliver a VIVA method which will support the user in selecting the right tool for the purpose throughout the KBS life-cycle. The tool-set will be developed and tested using applications drawn mainly from those developed by the European Space Agency (ESA) and Lloyd's Register.

The project co-ordinator is CRI-Computer Resources Intl. (Denmark), previously a member of the VALID project and there are six other partners: LIA—University de Chambery (France), University of Aberdeen (UK), Logica (UK), CISI (France), ESA and Lloyd's Register (UK).

7.2 American perspective

The majority of the research on verification and validation of KBS in the United States involves the development of support tools, several of which are described in section 6. NASA and its contractors have been involved in a number of verification and validation projects which were presented in the series of SOAR (Space Operations, Automation and Robotics) conferences. A relatively recent US research initiative is described below.

Guidelines for Verification and Validation of Expert Systems The overall goal of this project is to formulate and document guidelines for verifying and validating expert systems for use in nuclear utility applications. This work is sponsored jointly by the US Nuclear Regulatory Commission and the Electric Power Research Institute. Both are concerned with the quailty and reliability of expert systems used in nuclear utilities, the former from the point of view of regulatory safety concerns, and the latter from the point of view of providing utilities with effective means for developing expert systems which meet regulatory guidelines and standards. The Electric Power Research Institute in particular has since 1983 sponsored a broad programme for applying expert systems to utility needs, supporting tool development, practical applications, studies on verification and validation, and recently a training facility. The project intends to survey and evaluate conventional and KBS-specific practices for the verification and validation of KBS and on the basis of the results to develop and document a methodology for verification and validation.

This work is sponsored by the US Nuclear Regulatory Commission and Electric Power Research Institute, and carried out by the Defense Systems Group of Science Applications International

Corporation. The work will be carried out between October 1991 and October 1993 and the first three tasks of ten were complete as of April 1992.

8 Conclusions

This report has attempted to highlight some of the issues in the verification and validation of knowledge based systems. The recent EUROVAV-91 conference and a survey of the published literature indicate a lack of standardisation of the terminology used in this area, and that little distinction is made by authors between verification and validation. This makes comparison and contrast of work difficult. However, most authors agree that any re-definition of the terms for use in conjunction with KBS must be compatible with existing software engineering terms. After all, the ultimate purpose of verification and validation of conventional systems and KBS is the same.

Indeed, many of the components of verification and validation are the same for both KBS and conventional systems, although the emphasis placed on their verification and validation and in the ease with which it can be carried out seems to differ. The characteristics of KBS software which present particular difficulties for verification and validation are common to most advanced information processing systems. Problems may arise more from the complexity of the system being built and from the software development life cycle adopted than from the type of software used to implement the solution. This confirms the need for compatibility of the definitions of verification and validation between knowledge engineering and software engineering.

The current approaches taken to address the verification and validation needs of KBS fall into two broad categories. The first group address the requirement to produce a better specification and to reduce the number of misconceptions introduced prior to the start of coding. The techniques developed centre around the use of structured knowledge acquisition and knowledge transformation techniques. The second group concentrate on the provision of automated tools for code-checking and testing, as illustrated by the tool support section of this report. Until recently, the second group appears to have attracted most interest from researchers as this work is largely independent of domain knowledge and thus more easily generalized. However, there is a need to ascertain that the acquired domain knowledge is correct and coherent and, as can be seen from the discussion of current research initiatives, other work is addressing this issue. There does not appear to be much work concerning the identification of explicit stages in the development life-cycle when verification and validation procedures should occur.

Despite all the research which is currently underway, there seems to be no clear forum for the ideas emerging in the area of verification and validation. Papers concerning this issue are given at a variety AI/KBS, software engineering or domain-related conferences and published in the plethora of journals now available. This work must establish a small number of outlets through which to disseminate its results in order for researchers to keep up with current developments, to standardise on terminology and to prevent unnecessary duplication of effort.

9 Acknowledgements

Thanks are due to the Knowledge Engineering Methods group of AIAI for their contributions to the production of this report. In addition I would like to thank Dr. Robert Inder, whose previous work in verification and validation provided a basis for this work, and members of the VAVTALK international mailing list for their stimulating discussion on many of these issues. In particular thanks go to Dr. Lance Miller, Defense Systems Group, Science Applications International Corp. and Marc Ayel of LIA, Universite de Chambery, Savoie, France, for their contributions concerning current research projects.

References

Born, G, 1988, "Guidelines for quality assurance of expert systems" *CSA Working Group on QA and Expert Systems* (1.1), November.

Cragun, BJ and Steudel, HJ, 1987, "A decision-table-based processor for checking completeness and consistency in rule-based expert systems" *International Journal of Man-Machine Studies* **26** 633-648.

Enand, R, Kahn, GS and Mills, RA, 1990 "A methodology for validating large knowledge bases" *International Journal of Man-Machine Studies* **33** 361-371.

Hoppe, T and Meseguer, P, 1991, "On the terminology of VVT" In: M. Grisoni, editor, *Eurovav-91: Proceedings of the European Workshop on the Verification and Validation of Knowledge Based Systems* pp 103-108.

Inder, R and Filby, I, 1991, Survey of Knowledge Engineering Methods and Supporting Tools. Technical Report AIAI-TR-99, AIAI, December. (Also presented at the BCS SGES workshop on Knowledge-Based Systems Methodologies, December 1991.

Johnson, RG, Joly, GC and King, PJH, 1987, Survey of Techniques for the Checking of Rule-Based Expert Systems. Technical report, Department of Computer Science, Birkbeck College, London.

Kang, Y and Bahill, AT, 1990, "A tool for detecting expert system errors" *AI Expert* February, 46-51.

Liu, NK and Dillon, T, 1991, "An approach towards the verification of expert systems using numerical Petri nets" *International Journal of Intelligent Systems* **6** 255-276.

McGraw, KL and Harbison-Briggs, K, 1989, *Knowledge Acquisition: Principles and Guidelines*, pp 300-325, Prentice-Hall.

Mengshoel, OJ, 1991, "KVAT: a tool for incremental knowledge validation in a knowledge engineering workbench: In: M. Grisoni, editor, *Eurovav-91: Proceedings of the European Workshop on the Verification and Validation of Knowledge Based Systems* pp 133-146.

Nazareth, DL, 1989, "Issues in the verification of knowledge in rule-based systems" *International Journal of Man-Machine Studies* **30** 255-271.

Nazareth, DL and Kennedy, MH, 1991, "Verification of knowledge based systems using directed graphs" *Knowledge Acquisition* **3** 339-360.

Nguyen, TA, "CHECK applied to ART" In: *Third Conference on AI Applications* Washington, DC.

Nguyen, TA, Perkins, WA, Laffey, TJ and Pecora, D, 1987, "Knowledge base verification" *AI Magazine* **8**(2) 69-75, Summer.

O'Keefe, RM, Balci, O and Smith, EP, 1987, "Validating expert system performance" *IEEE Expert*, 81-89, Winter.

Partridge, D, 1986, *Artificial Intelligence: Applications in the Future of Software Development Engineering* Ellis Horwood.

Pearce, D, 1991, "A model based approach to validation" In: M. Grisoni, editor, *Eurovav-91: Proceedings of the European Workshop on the Verification and Validation of Knowledge Based Systems* pp 55-67.

Petitjean, S, Brunessaux, L and Vaudet, J-P, 1991, "Three pragmatic tools for the validation of knowledge based systems" In: M. Grisoni, editor, *Eurovav-91: Proceedings of the European Workshop on the Verification and Validation of Knowledge Based Systems* pp 111-123.

Ribar, G, Arcoleo, F and Hollo, D, 1991, "Loan probe: testing a big expert system" *AI Expert* 43-49, May.

Stachowitz, RA, Chang, CL, Stock, TS and Combs, JB, 1987, "Building validation tools for knowledge based systems." *Proceedings of the SOAR Conference*, Houston, TX NASA/JSC.

Stachowitz, RA, Combs, JB, 1987, "Validation of expert systems" In: *Proceedings of the 20th Annual Hawaii International Conference on System Sciences.*

van Someren, M, 1991, "Structural and formative validation of knowledge bases" In: M. Grisoni, editor, *Eurovav-91: Proceedings of the European Workshop on the Verification and Validation of Knowledge Based Systems.* pp 103-108