VALIDATION OF HIGHLY RELIABLE, REAL-TIME KNOWLEDGE-BASED SYSTEMS

Sally C. Johnson
System Validation Methods Branch, MS 130
NASA Langley Research Center
Hampton, VA 23665-5225

ABSTRACT

Knowledge-based systems have the potential to greatly increase the capabilities of future aircraft and spacecraft and to significantly reduce support manpower needed for the space station and other space missions. However, a credible validation methodology must be developed before knowledge-based systems can be used for life- or mission-critical applications.

Experience with conventional software has shown that the use of good software engineering techniques and static analysis tools can greatly reduce the time needed for testing and simulation of a system. Since exhaustive testing is infeasible, reliability must be built into the software during the design and implementation phases. Unfortunately, many of the software engineering techniques and tools used for conventional software are of little use in the development of knowledge-based systems. Therefore, research at Langley is focused on developing a set of guidelines, methods, and prototype validation tools for building highly reliable, knowledge-based systems.

The use of a comprehensive methodology for building highly reliable, knowledge-based systems should significantly decrease the time needed for testing and simulation. A proven record of delivering reliable systems at the beginning of the highly visible testing and simulation phases is crucial to the acceptance of knowledge-based systems in critical applications.

INTRODUCTION

Highly reliable, real-time knowledge-based systems (KBSs) have been proposed for many aerospace applications, including space station, manned and unmanned spacecraft, as well as civilian and military aircraft and other life-critical applications. For example, continuous operation of a space station will require extensive, around-the-clock monitoring by large numbers of expert ground control personnel unless some degree of system autonomy is obtained through the use of knowledge-based expert systems. Many of the systems proposed for the space station would result in loss of life if they were to fail during operation. Even when personnel are not involved, the loss of equipment and/or experiments can be prohibitively expensive. Therefore, these on-board systems must be reliable and validatable. Similarly, a pilot's associate or other advisory system, even if not in direct control of the craft, could only be used if the pilot were confident of its outputs. In many emergency situations, a pilot does not have the time to consider how the system arrived at its conclusion, but must quickly and confidently follow the directions he is given. If this were not the case, then the advisory system would never have been needed in the first place.

A credible validation methodology for highly reliable KBSs does not exist today. Most current research efforts in verification and validation of KBSs focus on a rapid-prototyping life cycle, review panels, testing, and development of limited static analysis tools for checking consistency and completeness of a rule base [1]. These techniques are necessary, but alone are not comprehensive enough to validate a system to be used in a life-critical application. Consistency and completeness checking only tests for a limited number of prespecified types of errors. The complexity of the knowledge base in a realistic system makes exhaustive testing impossible. More rigorous validation techniques must be developed.

This paper documents the ongoing research at NASA Langley to develop concepts, guidelines, and methodologies for the validation of KBSs. The scope of the effort and how Langley's research plan was developed are discussed. The state of the art in validation of conventional software is presented. Characteristics of KBSs affecting validation are discussed, and how validation of KBSs differs from conventional software is characterized. The research approach being followed at Langley is then presented, followed by details of the methods, guidelines, and prototype tools being developed. Finally, the expected results from this research project are discussed.

BACKGROUND

The research plan presented in this paper is the culmination of a research effort that began at NASA Langley in 1986 [1 - 4]. The first step was to characterize the potential needs for, and identify current research in, validation of KBSs

through workshops, classes, and industrial contacts. A research team with varied backgrounds from artificial intelligence, software engineering, and validation was then established. The differences between validation of a conventional software system and validation of a KBS were characterized, and applicability of conventional techniques to KBSs was assessed. The major issues and requirements particular to KBS validation were identified. A number of deficiencies in methods available for KBS validation became apparent, and a preliminary set of tools and methods to be developed were then identified to address those deficiencies.

Concepts, guidelines, methodologies, and supporting tools for the validation and verification of KBSs are to be developed. Because of the lack of available validation methods and the proliferation of KBS development projects, the methods and tools developed will be made available to near-term and mid-term KBS development efforts as soon as practicable. Feedback from these development efforts will provide valuable insight as to the effectiveness and comprehensiveness of the tools and methodologies developed.

The target applications are life-critical KBSs for NASA or military aircraft or spacecraft applications with any one or a combination of rule, frame, or object knowledge representations. Most of the tools and techniques developed will also be useful and cost effective for developing high quality KBSs for applications with less stringent reliability requirements. To keep the development effort feasible and within the bounds of realistic funding expectations, a number of issues will not be addressed, including the following topics: automatic programming, validation of learning, cost/reliability tradeoffs, and validation of advanced hardware architectures. These aspects can only be realistically addressed after significant advances are made in other KBS validation and verification areas.

VALIDATION OF CONVENTIONAL SOFTWARE

The development and validation of reliable conventional software is a major concern within NASA, the Department of Defense, and industry. After many years of research and the development of a new engineering discipline -- Software Engineering -- to address this problem, a number of techniques have been developed. Yet, the discovery of software "bugs" in operational life-critical software is not uncommon [5]. The FAA has not yet certified any civil air transports with flight-critical digital avionics. Thus, the techniques used today for conventional software may actually be inadequate for life-critical applications.

When conventional software is developed for life-critical military or space applications, validation is an ongoing process throughout the life cycle [6]. Limited design tools are available to aid in dividing the problem into a hierarchical set of modules. These modules are developed and tested separately and then integrated. The programmers adhere to strict coding standards and other techniques such as information hiding that have been found to lead to more reliable code. The developed code is subjected to extensive code walkthroughs and inspections in addition to the static checking provided by sophisticated compilers and other static code analysis tools. Each independently developed module is subjected to extensive testing. The interactions between modules are carefully tested during system integration. The system is then subjected to functional testing and finally simulation.

A system developed for a space application is reviewed periodically throughout the development life cycle by a safety assessment team from NASA to ensure that the procedures discussed above are closely followed [7]. Likewise, the developers of a system for a NASA experimental aircraft must convince a NASA safety team that the system is reliable before flight testing can begin. Similar procedures are followed by the Air Force to ensure adherence to MIL-STD-2167 and by the FAA for civil aircraft systems. The assessment teams typically base their estimates of the reliability of a given system on evidence of rigorous adherence to good software engineering techniques and documentation of traceability to specifications as well as on the absence of serious errors uncovered during testing.

Experience with conventional software has shown that the use of good software engineering techniques and static analysis tools can greatly reduce the time needed for testing and simulation of a system. Errors caught early in the implementation phase are significantly easier and less expensive to correct than those uncovered during the testing phase. The focus of the testing phase should be tuning system performance and promoting confidence about the inherent reliability of the program being tested. Errors caught during this phase should represent the occasional translation and coding errors, not major oversights or misunderstandings of the specifications.

Reliability is a characteristic that must be built into the program from the beginning. Making a poorly written program into a reliable one simply by extensive testing is at least extremely difficult and expensive, if not impossible.

CHARACTERISTICS OF KNOWLEDGE-BASED SYSTEMS

There are two major differences between KBSs and conventional software that affect the validation process -- structure and functionality.

A KBS is divided into some form of knowledge base, which may be rules, frames, procedures, or some other structure or combination of structures, and a reasoning algorithm, such as an inference engine, which operates on the knowledge. This separation of the system into algorithm and data, plus the inherent structuring of the knowledge base may actually aid in the validation process.

Unfortunately, many of the techniques and tools used for conventional software are of little use in the development of KBSs. Researchers are just beginning to develop guidelines for implementing software engineering concepts such as

modularization, information hiding, and structured coding. Development shells and preferred KBS languages such as LISP and Prolog do not support strong typing and other features used in static code analysis, and the compilers do little static checking for errors. Code walkthroughs are less effective for KBSs because each piece of the knowledge is viewed individually and interactions are difficult to conceptualize. The use of new symbolic or parallel architectures significantly compounds the validation problem.

In addition to the above differences attributable to the KBS implementation method, there are further differences caused by the fact that KBSs are often used to implement "expert systems." A KBS is usually expected to have considerably more functionality than would be expected for a conventional software system, especially in the case of an expert system. Much of how the system is to operate is not explicitly known at the start of the project and is to be determined by the knowledge engineer during system development. Expert system applications are typically characterized by the absence of a well-understood algorithm or even well-known performance requirements. The knowledge is often poorly understood and may come from different and even conflicting sources. Access to the expert sources may be limited. A rapid-prototyping development life cycle is used, making traceability of requirements to the code more difficult to ensure. The rapid-prototyping life cycle is not unique to KBSs and is beginning to be studied extensively as an acceptable method for developing conventional software. However, it is still generally recommended that the prototype be discarded or used as a working specification for the development of the real system. Without a well-understood algorithm to follow and with often limited access to the "expert," compiling test cases to assess whether the system is operating "correctly" is usually expensive and difficult. These characteristics have given KBSs a well-deserved reputation for ad hoc, trial-and-error development. Therefore, very rigorous verification of safety will be necessary before a KBS can be certified for use in a life-critical application.

A complete validation methodology must necessarily include guidelines for system development throughout the software life cycle. The rapid prototype scheme of software development, which is very favorable for the development of KBSs, must be accompanied by a specification of the system. The rules used in the prototype represent the knowledge that has been collected about how the system should perform. However, there may be unanticipated interactions between these rules. The system specification should include information about the contents of the knowledge base and deductions that should be possible from it. This "metaknowledge" becomes the basis for the validation effort and should include both "do's"--a specification of what the system should do--as well as "don'ts"--what the system explicitly should not do. Each of these assertions about the system must be classified as to level of criticality--whether failure of the assertion could cause loss of life or property or simply inefficiency or passenger discomfort. Most applications will contain a mix of assertions of various criticality levels.

The most critical assertions of what the system should and should not do, such as crash the plane, must be verified using rigorous techniques, such as formal verification. The search algorithms employed and their implementations and interactions must also be rigorously verified. This includes verification that the search algorithms will complete within real-time deadlines.

APPROACH

The emphasis on KBS validation research at NASA Langley has been placed on aiding the KBS developer in building a quality product and assessing it before the final phases of testing and simulation are reached. Testing and simulation are then used to assess and tune how well the KBS performs the desired functionality requirements, rather than to try to verify safety properties.

There are several reasons for concentrating research efforts on the design and implementation phases. Two reasons come from experience with conventional software. First, errors are much easier and less expensive to correct if uncovered early in the development life cycle. Also, since exhaustive testing of a nontrivial system is impossible, testing cannot be expected to catch enough errors to change an inherently unreliable program into a reliable one. Most importantly, the largest impediment to deployment of KBS for a life- or mission-critical application is a categoric lack of confidence in all KBSs on the part of those who ultimately make such decisions. This is true of any methods or technologies that are viewed as being radically new and different. The only way to change this image is to arrive at the highly visible testing phase with reliable software and use testing merely to tune system performance. Seeing the uncovering of a number of serious errors during the testing phase of any piece of software alarms safety review teams.

Thus, NASA Langley's efforts in KBS validation research will consist of developing and assessing a number of guidelines and methods for building high reliability into KBSs before they reach the testing phase. The research topics being pursued by NASA Langley and its contractors and grantees are discussed in the following section. Some of the projects discussed below have not even begun yet, and few have progressed past an initial feasibility study phase.

THE PRELIMINARY SET OF TOOLS AND METHODS

A preliminary set of guidelines, methods, and tools have been identified as promising for the development and validation of highly reliable, real-time KBSs. Prototypes of the tools will be developed and integrated with a development environment. The methods and guidelines will be developed, documented, and demonstrated on KBS applications. The preliminary research projects to be pursued include:

- guidelines for scoping the application
- requirements documentation tool
- guidelines for knowledge acquisition
- a development environment supporting software engineering techniques
- consistency and completeness checking tool
- sensitivity analysis tool and guidelines
- methods and tools for formal verification of safety properties
- a base of reasoning algorithms formally characterized to support formal verification
- methods for real-time performance analysis
- methods for implementing a KBS on a fault-tolerant parallel processor

The tools and methods will be applied to several applications, such as the Systems Autonomy Demonstration Project (SADP) demonstration systems, to assess their effectiveness.

Scoping the Application

Before development begins, it is essential to determine a feasible application, or to "scope" the application. This is especially difficult and important for a KBS because of the overzealous selling of AI leading to statements such as "we don't have to know how to do it, we can program it using AI." A set of periodically updated guidelines will be developed for choosing and scoping applications for development. As more KBS development and validation tools and software engineering methods become available, these guidelines will be modified to reflect the current state of the art in KBS development.

Requirements Definition

Validation must be in mind from the beginning of system development. To be useful later in the validation phase, the requirements for the system are divided into the following categories [1]:

1. Desired Competency Requirements -- How well the system is expected to perform. Since the functionality desired from the system is often poorly understood before the system is built, these may of necessity be vague and incomplete.

2. Minimum Competency Requirements -- What the system explicitly must do and must not do to ensure safe operation. These must be precise and comprehensive to support validation and should be rated as to level of criticality.

The requirements developed during this phase and the metaknowledge collected during the knowledge acquisition phase will be documented using a requirements documentation tool. This tool will support traceability between the requirements and the implementation. Also, the consistency and completeness checker and safety property verification tool will directly access this information during the validation phase. Guidelines for developing specifications and guidelines for specification of safety properties will also be developed.

Knowledge Acquisition

A set of guidelines for knowledge acquisition to support validation will be developed. Three types of information should be collected from the experts during the knowledge acquisition phase:

1. Knowledge -- Procedural information about how the system should perform its operation.

2. Metaknowledge -- Metaknowledge, or knowledge about knowledge, describes constraints on the knowledge that can later be used for consistency and completeness checking. The metaknowledge should be documented using the Requirements Documentation Tool.

3. Test cases -- Examples of what proper outputs would be for given inputs to the system.

System Development

The knowledge base is developed from the above information using rapid prototyping on a system development environment, similar to an expert system shell. The system development environment will form the core of the integrated toolset. The development environment must be able to support the development of a KBS composed of a combination of knowledge representations of rules, frames, and objects. The validation and verification tools must be able to directly access the KBS as it is developed. The reasoning algorithm will be chosen from a suite of algorithms or separately developed and formally characterized.

A basic development environment will be chosen from the available environments. The chosen environment will then be enhanced to extend its capabilities, provide support for frames and objects as well as rules, and target it to support probable future NASA applications. Much research will also be done in assessing the software engineering techniques being developed for KBS and in developing new methods such as those used for conventional software, including coding standards for modularization, information hiding, and strong typing. An example of the application of software engineering techniques to KBSs may be found in [8]. Support for these software engineering techniques will be added to the development environments.

Consistency and Completeness Checking

A static analysis tool, including a completeness and consistency checker will be integrated with the toolset to automatically check that the knowledge in the system meets the conditions described by the metaknowledge collected during knowledge acquisition. Quite a number of static analysis tools with various capabilities are currently being developed in industry [9-12]. The Lockheed AI Center has been identified as the source for research and development of a static analysis tool because of their extensive background and sizeable accomplishments in the development of the EVA system.

Current tools are still limited in what they can check for; however, checking of more complex forms

of metaknowledge should be possible in the future. Research will be conducted to assess the usefulness of various types of static KBS analysis. The tool will then be enhanced to provide the types of checking found to be most useful. The static analysis tool will be very useful for finding some types of errors in a knowledge base, but it can only find errors that specifically violate the metaknowledge given. Some verification that the system meets its minimum competency requirements could be done by this tool.

Sensitivity Analysis

Because of the trial-and-error methods often employed in KBS development, KBSs frequently exhibit "instability" or "fragility" properties. These include sensitivities to:

1. Sequence dependencies -- Depending on the order of rule firings, the same input can produce wildly different outputs.

2. Input values -- Slight changes in input values produce extreme changes in output values.

3. Constants -- Slight changes of numerical values contained within the knowledge base, such as constants encoded within the rules or certainty factors, produce extreme changes in output values.

These sensitivities do not necessarily mean that an error is present, but point to likely errors and to values which must be very accurate because the system computation is extremely sensitive to them.

The sensitivity analysis research and tool development is being conducted under a grant to Worcester Polytechnic Institute. A sensitivity analysis tool will be developed to automatically perform specified sensitivity analyses. The tool development is based on the use of Evidence Flow Graphs, which are independent of the knowledge representation of the KBS [3]. A rules-to-graph translator is already being developed to automatically translate a knowledge base of rules from the system development tool into a graphical structure. Other translators to perform translation of other knowledge representations will also be developed. Research will be conducted to extend the types of sensitivity analyses performed and assess their usefulness in finding errors in the knowledge base.

If the system is to be used in a control application, its stability must be validated [13]. Each input value is known only within certain error tolerances. This value is manipulated mathematically using other input values and parameters that also have degrees of error. It must be shown that the result computed by the system is within the tolerance needed by the system. The maximum error of input values and parameters as well as the error tolerances allowable on the outputs must be included in the specification.

Verification of Safety Properties

The KBS must be mathematically verified to meet the minimum competency requirements for safe operation. This is an expensive step, but one that is necessary for life-critical applications. Research into specification of safety properties and mathematical verification of them is being conducted by SRI International. These procedures will be applied to an example application to demonstrate the feasibility of formal verification of safety properties of a realistically complex system. A safety verification tool will be developed to aid the user in this process. The actual mathematical verification will be performed by a theorem prover being developed by SRI for conventional software and hardware [14]. The safety verification tool is basically an interface between the development environment and the theorem prover and will directly access the knowledge base and reasoning control information stored in the development environment.

Reasoning Algorithms

Although many KBSs are written in rules that look like sentences in formal logic, reasoning algorithms typically perform operations that bear no resemblance to first-order logic, such as Prolog's treatment of negation and "cuts." For formal verification of safety properties to be possible, the formal semantics of these features must be defined and adherence of the algorithm used to the defined semantics must be verified. For most applications, one or a combination of several reasoning (inference) algorithms will be chosen from an established base of algorithms. If a new reasoning algorithm must be developed for the application, the semantics of the new algorithm must be defined and verified. Research to develop techniques for semantic characterization of reasoning algorithms verification of those characterizations, as well as establishment of a base of characterized reasoning algorithms will be performed by SRI International.

Real-time Performance Analysis

Large KBSs are notorious for their very slow performance. Any performance gains expected from the development of faster symbolic processors and more efficient implementations will probably be offset by the growing size and complexity of systems. Because of interactions between knowledge, addition of knowledge to the system can result in exponential increases in search times. Verification that the search algorithms will complete within real-time deadlines will be very important in applications such as aircraft control and advisory systems that have deadlines on the order of a few milliseconds.

A worst case analysis will probably be too conservative to be useful for many systems. However, it may be possible to show analytically that the probability of missing a real-time deadline is within the reliability requirements of the system.

Real-time performance is a function of the hardware architecture, the reasoning algorithm,

how that algorithm is implemented on the hardware architecture, plus the structure and contents of the knowledge base. Techniques for performing real-time performance analysis based on measurable parameters of the system will be developed by the Charles Stark Draper Laboratory.

Parallel Architectures

As KBSs become larger and more complex, the use of parallel architectures will be necessary to obtain acceptable performance. The Charles Stark Draper Laboratory is developing a functional programming model for implementation of a KBS on a fault-tolerant parallel processor. The programming model will provide for graceful degradation, deadlock detection and recovery from excessive generation of parallelism, and distributed checkpointing and error recovery as well as load balancing to increase system performance. Once the programming model is implemented, the system will be used to study optimal KBS parallelization schemes for maximizing performance on a parallel processor and to study real-time performance analysis.

EXPECTED RESULTS

Although none of the tools and methods will be completed in the near term, many of the basic concepts behind those tools and methods are already being developed. Much of the development and validation methodology will be useful to system builders in the near term even before details are worked out and tools are developed. This includes guidelines for what types of information should be collected during the requirements specification and knowledge acquisition phases, how this knowledge can support the validation effort, and various sensitivity analyses to be performed. Guidelines for choosing and scoping a feasible application will have been documented, and a description of software engineering practices that are useful for KBS will have been developed. The first flight test of a simple KBS application, the Mode Control Logic Panel developed by Langley's Aircraft Guidance and Controls Branch, will be conducted in Summer 1988 on the Advanced Transport Operating Systems (ATOPS) aircraft at Langley. This system was developed as a rule-based system then coded in the C programming language.

An integrated prototype toolset with limited validation capabilities should be available for system builders to use by the mid 1990's. The tools and methodologies will be made available to interested KBS developers as beta-test sites, and documentation and consultation on the use of the tools will be made available. Feedback as to the usability and effectiveness of the tools and techniques will be a crucial part of future planning.

CONCLUDING REMARKS

The aim of research at NASA Langley in validation of KBSs is to develop a set of guidelines, methods, and tools to aid a KBS developer in building a highly reliable KBS. An integrated toolset of prototype tools will be developed to demonstrate the methods and how to implement them. The integrated toolset will in no way be comprehensive enough to support the development of all or even most future NASA KBS applications. The development of a user-friendly toolset with an advanced, comprehensive development environment will be left to industry, but will hopefully be supported by the core research of this project.

The methods and tools being developed purposefully end at the beginning of the testing phase. Exhaustive testing of a realistically complex KBS is impossible. Many KBSs are expected to have considerably more functionality than conventional software and to operate correctly in unanticipated environments. Testing over various expected scenarios will typically uncover only very obvious errors and will not significantly add to the robustness of the KBS or its ability to operate correctly in other unanticipated scenarios. Thus, the system should be relatively reliable before it reaches the testing phase, and testing and simulation should be concentrated on tuning system performance.

REFERENCES

1. Rushby, John, "Quality Measures and Assurance for AI Software," Contract NAS1-17067 Task 5 Final Report, SRI International, Menlo Park, CA., 1988.

2. Morell, Larry, "Use of Metaknowledge in Verification of Knowledge-Based Systems," First International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Tullahoma, Tennessee, June 1-3, 1988.

3. Green, Peter, and Becker, Lee, "Evidence Flow Graph Methods for Validation and Verification of Expert Systems," NASA CR to be published.

4. Beaton, Robert, "Reliability and Performance Evaluation of Systems Containing Embedded Rule-Based Expert Systems," NASA CR to be published.

5. Neumann, Peter G., "Some Computer-Related Disasters and Other Egregious Horrors," ACM Software Engineering Notes, Vol. 10, No. 1, January 1985.

6. Spector, Alfred, and Gifford, David, "The Space Shuttle Primary Computer System," Communications of the ACM, Vol. 27, No. 9, September 1984.

7. Culbert, Chris, Riley, Gary, and Savely, Robert, "Approaches to the Verification of Rule-Based Expert Systems," First Annual Workshop on Space Operations, Automation, and Robotics (SOAR87), NASA CP 2491, August 1987.

8. Jacob, Robert J. K., and Froscher, Judith N., "Developing a Software Engineering Methodology for Knowledge-Based Systems," Naval Research Laboratory, Arlington, Va., NRL Report 9019, December 1986.

9. Stachowitz, R. A., Combs, J. B., and Chang, C. L., "Validation of Knowledge-Based Systems," Second AIAA/NASA/USAF Symposium on Automation,

Robotics and Advanced Computing for the National Space Program, Arlington, Va., March 1987.

10. Bellman, Kirstie L. and Walter, Donald O., "Testing Rule-Based Expert Systems," November 1987, submitted for publication.

11. Suwa, Motoi, Scott, A. Carlisle, and Shortliffe, Edward H., "An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System," AI Magazine, Vol. 3, No. 4, Fall 1982.

12. Nguyen, Tin A., Perkins, Walton A., Laffey, Thomas J., and Pecora, Deanne, "Knowledge Base Verification," AI Magazine, Vol. 8, No. 2, Summer 1987.

13. Castore, Glen, "A Formal Approach to Validation and Verification for Knowledge-Based Control Systems," First Annual Workshop on Space Operations, Automation, and Robotics (SOAR87), NASA CP 2491, August 1987.

14. Moser, Louise, Melliar-Smith, Michael, and Schwartz, Richard, "Design Verification of SIFT," NASA CR 4097, September 1987.