AN EXPERT SYSTEM DEVELOPMENT METHODOLOGY THAT SUPPORTS VERIFICATION AND VALIDATION

Chris Culbert Gary Riley
Robert T. Savely

Expert systems have demonstrated commercial viability in a wide range of applications but still face some obstacles to widespread use. A major stumbling block is the lack of well-defined verification and validation (V&V) techniques. The primary difficulty with expert system V&V is the use of development methodologies that do not support V&V. As with conventional code, the key to effective V&V is the development methodology based upon a panel review approach that allows input from all parties.

INTRODUCTION

Expert systems represent one important byproduct of artificial intelligence research efforts. They have been under development for many years and have reached commercial viability in the last three to four years. However, despite their apparent utility and the growing number of applications being developed, not all expert systems reach the point of operational use. One reason for this is the lack of well understood techniques for V&V of expert systems.

Developers of computer software for use in mission or safety-critical applications have always relied upon extensive V&V to ensure that safety and/or mission goals were not compromised by software problems. Expert system applications are computer programs and the same definitions for V&V apply to expert systems. Consequently, expert systems require the same assurance of correctness as conventional software.

Despite the clear need for V&V, considerable confusion exists over how to accomplish V&V of an expert system. There are even those who question whether or not it can be done. This confusion must be resolved if expert systems are to succeed. As with conventional software, the key to effective V&V is through the proper use of a development methodology that both supports and encourages the development of verifiable software.

THE COMMON EXPERT SYSTEM DEVELOPMENT METHODOLOGY

Most existing systems are based upon relatively new software techniques, which were developed to describe human heuristics and to provide a better model of complex systems. In expert system terminology, these techniques are called knowledge representation. Although numerous knowledge representation techniques are currently in use (rules, objects, frames, etc.), they all share some common characteristics, one of which is the ability to provide a very high level of abstraction. Another is the explicit separation of the knowledge, which describes how to solve problems, from the data, which describes the current state of the world.

Each of the available representations have strengths and weaknesses. With the current state of the art, it is not always obvious which representation is most appropriate for solving a problem. Therefore, most expert system development is commonly done by rapid prototyping. The primary purpose of the initial prototype is to demonstrate the feasibility of a particular knowledge representation. It is not unusual for

ISSN 0019-0578/89/01/0015/04/$2.50

entire prototypes to be discarded if the representation doesn't provide the proper reasoning flexibility.

Another common characteristic of expert system development is that relatively few requirements are initially specified. Typically, a rather vague, very general requirement is suggested, e.g., "We want a problem to do just what Charlie does." Development of the expert system starts with an interview during which the knowledge engineer tries to discover both what it is that Charlie does and how he does it. Often there are no requirements written down except the initial goal of "doing what Charlie does". All the remaining system requirements are formulated by the knowledge engineer during development. Sometimes, the eventual users of the system are neither consulted nor even specified until late in the development phase. As with conventional code, failure to consult the intended users early in the development phase results in significant additional costs later in the program.

So where does all this lead? The knowledge engineer is developing one or more prototypes that attempt to demonstrate the knowledge engineer's understanding of Charlie's expertise. However, solid requirements written down in a clear, understandable, *easy to test* manner generally don't exist. This is why most expert systems are difficult to verify and validate: not because they are implicitly different from other computer applications, but because they are commonly developed in a manner that makes them very difficult or impossible to test.

EXPERT SYSTEM ISSUES FOR A GOOD V&V METHODOLOGY

From the preceding section, it should be clear that a major problem with V&V of expert systems is the use of development methodologies that do not generate requirements that can be tested. The goal of any software development methodology is to produce reliable code that is both maintainable and verifiable. A software development methodology for expert systems must serve a similar purpose as one for conventional software. However, some differences between expert systems and conventional software will affect the development methodology.

Standard software development methodologies place emphasis on the capture of complete and rigid specifications to improve the ability of the system designers to find errors early in the design cycle. The specifications define both the intended purpose and the intended behavior of the program. These specifications are translated into requirements and serve as the primary comparison point for all V&V efforts.

Some expert systems can probably be developed by using conventional software engineering techniques to create software requirements and design specifications. [1] However, the type of knowledge used in expert systems doesn't always lend itself to this approach. It is best obtained through iterative refinement of a prototype that allows the expert to spot errors in the expert system reasoning before he can clearly specify the correct rules. Experts are not always able to articulate either the methods or the underlying reasons they use to solve a problem without applying the knowledge in context.

Another difficulty in writing complete specifications is that some expert systems deal with problems where there is no correct, absolute answer, only more or less adequate answers. [2] In this situation, it can be difficult for the expert to provide complete descriptions of what outputs the expert system should generate. It is more common to specify how the expert system should behave under certain circumstances. By properly modeling this behavior the expert system will presumably produce acceptable answers.

When defining a development methodology for expert systems, the broad range of applications to which they can be applied must also be considered. An expert system application can be anything from an entire program dedicated to complex reasoning to an embedded, integrated application where the reasoning portion is only one module in a larger conventional program. If the primary focus of the program is reasoning and the expert system is the dominant piece of code, certainly one would use a development methodology intended specifically for expert systems. If the reasoning module is only a small piece in an otherwise conventional program, the unique characteristics of expert systems still dictate the use of an development methodology specific to expert systems for that section of the program. However, the expert system development methodology must be consistent with the overall development methodology. Modular coding techniques that allow separate development of each module will greatly aid the integration.

A DEVELOPMENT METHODOLOGY FOR EXPERT SYSTEMS

In a previous paper, [3] the authors suggested a panels approach to verification of expert systems. A panel consisting of the expert system developers, the domain experts, the system users, and managers with system responsibility represents all applicable viewpoints. Verification of both design and purpose are provided by regular panel review. The following methodology incorporates this approach and is based upon a life cycle model proposed by Citrenbaum and Geissman. [4] The steps proposed below are most appropriate when developing an expert system that captures expertise from only one or two experts in a single domain area. It allows for the inclusion of conventional code portions, but the reasoning part of the system is the dominant piece. The modifications needed to adapt this methodology for large systems with multiple experts from multiple domains will be discussed in future paper.

The expert system life cycle can be split into four phases: the Problem Definition Phase, the Initial Prototype Phase, the Expanded Prototype Phase, and the Delivery/Maintenance Phase.

Problem Definition Phase

This phase is similar to the requirements definition stage

of a conventional program. The purpose of this work is to define what it is the expert system should do. The actual activities can be broken down as follows:

(1) Discuss the general domain area and focus on the types of problems the system managers would like to have solved. If needed, provide general background information on what expert systems are and what they are capable of.
(2) Identify who the experts are and arrange for access to them. Identify the users of the expert system.
(3) First panel meeting. Discuss the problem area with the experts, the users, and the managers to ensure consistent understanding of the purpose of the expert system. Identify and discuss the potential delivery environment.
(4) Interview experts at least two to three times and read any applicable documentation or background material. Focus on what it is the expert does and how he or she does. it.
(5) Interview potential users of the expert system. Focus on their skill levels with respect to the domain. Identify what their computer skills are.
(6) Select a representative subset of the full problem that can be rapidly developed in an initial prototype to demonstrate feasibility.
(7) Examine knowledge representation methodologies and choose a tool or shell for prototype development.

At the end of this phase, an initial requirements review (IRR) should be held by the full panel to present findings to date and to begin the documenting of requirements for future V&V efforts. The IRR report should include:

(1) an initial description of what it is the expert system should do. Identify both the long-term goals for the program and the short-term goals for the initial prototype;
(2) an initial estimate of the amount of time the expert will be needed for consultation during the next phase;
(3) a discussion of any limitation or assumptions about the users and how they will use the expert system. Also identify specific skills the users must have to work with the expert system;
(4) a discussion of potential limitations or trade-offs inherent in the delivery environment;
(5) a detailed listing of the requirements currently established; and
(6) a tentative schedule for demonstration of the initial prototype.

Initial Prototype Phase

This stage is similar to the design phase of a conventional program. The purpose of this work is to quickly demonstrate the feasibility of the project and establish the most appropriate knowledge representation techniques. The activities during this phase include the following:

(1) Design the system architecture. Select a knowledge representation method and an inference mechanism. Select an appropriate tool for development (Note that differences between the development environment and the delivery environment may influence tool choice).
(2) Identify and define the interfaces between expert system and any external information sources (databases, users, other programs, special hardware). Define what portions of the problem will use conventional code.
(3) Construct the initial prototype, focusing on the knowledge base and inferencing techniques. Generally, the interfaces are not fully developed and hooks are left in the knowledge base for future expansion. Develop the conventional code sections only as needed to support reasoning modules. Consult the experts as needed to clarify information.
(4) Demonstrate the partially completed knowledge base to the domain experts at least two or three times. These demonstrations will improve the experts' abilities to correct mistakes in understanding or approach at an early stage.

At the end of this phase, a prototype demonstration should be held to show the panel how the project is proceeding. Any requirements gathered during this phase should be documented. A preliminary design review (PDR) should be held with the full panel to discuss architecture issues and overall project feasibility.

Expanded Prototype Phase

This stage is similar to the coding phase of a conventional program. The purpose of this work is to expand the capabilities of the expert system to its full extent. The activities during this phase are iterative, and the following steps may be repeated many times until a satisfactory prototype is completed.

(1) Demonstrate the prototype to the full panel. Encourage the use of real problems and previous cases for comparison. Accept inputs from all portions of the panel about the overall system design and capabilities. This is the time when the requirements become more consistent and rigid.
(2) Critically examine the initial prototype. Further development can continue from this base, or it may be appropriate to throw away the initial prototype (keeping the knowledge base) and start with a new model.
(3) Discuss the user interface needs with the users. Expand the hooks from the initial prototype to include all features needed in the system. Develop and integrate all other interface areas (databases, external hardware, etc.). If the delivery system needs to be ported, consideration should be given to designing portable interfaces.
(4) Expand the knowledge in the knowledge base to cover all aspects of the full problem. Consult the experts as needed to clarify the reasoning. Demonstrate the system to experts regularly to get feedback on correctness of reasoning and validity of solutions.
(5) Fully develop and integrate any conventional code portions of the system.
(6) If long-term maintenance is to be provided by someone

other than the developers, the system maintenance should be involved in the final portions of extending the prototype.

At the end of this phase, the expert system should be fully functional. As with the previous stage, all requirements gathered during this stage should be documented. Depending on the project, the requirements can be documented at the end of this phase or throughout the development. In either case, all the requirements, including those gathered during the first two phases, should be documented in a formal System Requirements Document. A formal System Design Review should be held with the full panel. The panel as a whole should publish a test plan that describes how the completed expert system will be verified.

Delivery/Maintenance Phase

This phase potentially has as many as three steps. If the development environment is different from the delivery environment, then the expert system must first be ported to the delivery environment. Once the expert system is running in the target environment, final V&V testing can be done. Finally, after formal delivery of the expert system, a maintenance stage can be initiated.

The testing stage provides the final V&V efforts. There are many concerns during this stage that are common to both expert system and conventional programs as well as some concerns unique to expert systems including such issues as verifying the inference engine, correctness of reasoning, tracing of requirements, etc. These are discussed more fully in Reference 3. The method of testing should have been specified in the test plan generated at the end of the expanded prototype phase. Problems encountered during testing should be reviewed by the full panel and may send the project back into the extended prototype phase.

After the program has passed final testing, the expert system can be placed under standard configuration control. Depending on the type of program and its use, the review panel may become a configuration control board, or responsibility may be turned over to another group. Maintenance from this stage on can be handled in a fairly standard manner, although extensive changes may require further prototyping work and review by the original panel.

CONCLUSIONS

V&V of expert systems is necessary for the eventual use of this technology. The primary hindrance to effective V&V is the use of methodologies that do not produce traceable, testable requirements. In this paper we have presented a methodology based upon the panel approach, which can provide the needed requirements as well as consistent, continual verification through panel review.

REFERENCES

(1) Bochsler, D.C., and Goodwin, M.A., "Software engineering Techniques Used To Develop an Expert System for Automated Vehicle Rendezvous," Proceedings of the Second Annual Workshop on Robotics and Expert Systems, Instrument Society of America, Research Triangle Park, NC, June 1986.

(2) Partridge, D., "Engineering Artificial Intelligence Software," *Artificial Intelligence Review*, Vol. 1, No. 1, 1986.

(3) Culbert, C. J., Riley, G., and Savely, R. T., "Approaches to the Verification of Rule-based Expert Systems," Proceedings of SOAR'87:P Space Operations-Automation and Robotics Conference, Houston, TX, August 1987.

(4) Citrenbaum, R. L., and Geissman, J. R., "A Practical Cost-Conscious Expert System Development Methodology," Proceedings of AI-86: Artificial Intelligence and Advanced Computer Technology Conference, Long Beach, CA, April 1986.