https://ntrs.nasa.gov/search.jsp?R=19930002757 2020-03-17T09:49:27+00:00Z

N93-11945

KNOWLEDGE-BASED SYSTEM V&V IN THE SPACE STATION FREEDOM PROGRAM

Keith Kelley & David Hamilton
IBM Federal Sector Division
MC 6402
3700 Bay Area Boulevard
Houston, Texas 77058

Chris Culbert
NASA/Johnson Space Center
Software Technology Branch/PT4
Houston, Texas 77058

ABSTRACT

Knowledge Based Systems (KBSs) are expected to be heavily used in the Space Station Freedom Program (SSFP). Although SSFP Verification and Validation (V&V) requirements are based on the latest state-of-the-practice in software engineering technology, they may be insufficient for Knowledge Based Systems (KBSs); it is widely stated that there are differences in both approach and execution between KBS V&V and conventional software V&V. In order to better understand this issue, we have surveyed and/or interviewed developers from sixty expert system projects in order to understand the differences and difficulties in KBS V&V. We have used this survey results to analyze the SSFP V&V requirements for conventional software in order to determine which specific requirements are inappropriate for KBS V&V and why they are inappropriate. Further work will result in a set of recommendations that can be used either as guidelines for applying conventional software V&V requirements to KBSs or as modifications to extend the existing SSFP conventional software V&V requirements to include KBS requirements. The results of this work are significant to many projects, in addition to SSFP, which will involve KBSs.

INTRODUCTION

Knowledge-based systems, or expert systems, are in general use in a wide variety of domains. (Although there is a growing acceptance of different definitions for knowledge-based systems and expert systems, we will use the terms interchangeably in this paper. The differences between KBS and expert systems do not significantly affect the V&V process.) As reliance on these types of systems grows, the need to assess their quality and validity reaches critical importance. As with any software, the reliability of a KBS can be directly attributed to the application of disciplined programming and testing practices throughout the life-cycle. However, there are essential differences between conventional software and knowledge-based systems, both in construction and use. The identification of how these differences affect the verification and validation (V&V) process and the development of techniques to handle them is the basis of work in this field.

Much of the work in KBS V&V has focused on developing conceptual approaches and postulating different techniques for performing some or all aspects of V&V on various types of KBSs or expert systems (ESs) [5]. Very little work in this field has demonstrated the usefulness of proposed techniques on operational KBS. Even more importantly, since effective V&V must be applied throughout the life-cycle, there has been almost no case study work in applying disciplined software V&V principles throughout the development of an operational KBS. The long term goal of our work is to develop guidelines, standards, tools, and techniques for V&V of all KBS applications which many be used in the Space Station Freedom Program (SSFP). As a precursor to determining the applicability or usefulness of many of the proposed KBS V&V techniques, it is important to develop an understanding of what V&V practices are commonly in use today and how proposed techniques can improve upon those practices.

It has been widely claimed that few expert systems are subjected to the same level of V&V that conventional

software routinely undergoes [4]. However, this practice has not been well documented. More important for our purposes, little documentation exists (an exception is documented in [8]) which describe the problems associated with KBS V&V from the developer or user's point of view. The specific purpose of our survey was to begin documenting the experiences and problems KBS developers have encountered in performing V&V on their systems and relate those problems to the kinds of issues KBS V&V researchers consider important. The overall strategy for determining the state-of-the-practice was to determine how well each of the potential expert system V&V issues are being addressed and to what extent they have impacted the development of expert systems. Our approach was to develop a set of survey questions for both KBS developers and users and then to follow that survey with selected interviews.

Because our ultimate goal is to develop guidelines, etc. for SSFP, we compared the results of our survey to the existing SSFP V&V requirements. We also analyzed all the SSFP V&V requirements to determine their general applicability to KBS V&V.

In this paper, we first summarize the results of this survey (a more complete discussion of the survey results appears in [9]) and then we summarize the results of analyzing SSFP V&V requirements.

SURVEY RESULTS

A total of 70 people, 93% of which were developers, responded to the survey concerning a variety of knowledge-based systems. Seventy percent of these systems were operational and the remainder were considered prototypes (although some of these "prototypes" had users). These systems covered a range of criticalities and sizes, requiring as little as one person-month of development effort to as much as two hundred person-months of development. Most (75%) of the systems were concerned with diagnosis, primarily in the aerospace field (73%).

Questionnaire Results

Much of the results can be derived by simply calculating the fraction of respondents that answered a question in a certain way. The following is a short summary of each type of information gathered. Unless otherwise noted, the percentages shown are the percentage for all the responses, both developer and user combined.

Performance Criteria

Thirty-nine percent estimated that the expert system performed with an actual accuracy of less than 90% and 54% estimated an accuracy of less than 95%. Most (50%) estimated the problem space coverage between 60% and 95%. In comparing the accuracy of the expert and the expert system, most (79%) expected the expert system to at least as accurate as the expert. Yet, the actual systems were often (75%) estimated to be less accurate than expected and also (62%) less accurate than the expert. Users, more often than developers, estimated the expert system as being less accurate than expected and less accurate than the expert.

Requirements Definition

Seventy-five percent indicated that expert consultation was a basis for determining the behavior of the system. More revealing is that for 52% of the systems surveyed, there were no documented requirements. Forty-three percent indicated that prototypes or similar tools were used for requirements. Forty percent had medium difficulty in generating requirements, 35% said the requirements were hard to develop, 25% said the requirements were easy to develop. Fifty-eight percent of developers had a high level of contact with experts during development.

Development Information

The most frequent (40%) life-cycle model used is the Cyclic Model (repetition of Requirements, Design, Rule Generation, and Prototyping until done). However, 22% of the respondents stated that no model was followed. Most development was done with an expert system shell (CLIPS and others), and the predominant Interface Code was C and LISP. Applications were reasonably large, requiring an average of 23 person-months to develop. Developed systems were not reported to be particularly sensitive to change (77% said changes only occasionally caused an unexpected behavior).

V&V Activities Performed

Most V&V activities relied on comparison with expected results and checking by the expert. Sixty-six percent used functional testing and 44% used structural testing. Fifty-nine percent had the domain expert check the knowledge base. On average, 24% of the development was spent on V&V. While all (100%) of the users rated V&V of expert systems as hard, the response from developers varied. Thirty-four percent of the developers said the V&V effort was of medium difficulty while 27% said it was hard and 33% said it was easy, 5% said it was impossible. Significantly, each V&V technique was used as the sole V&V technique in at least one project. Also, in general, there were wide ranging uses of V&V techniques; each technique was used by many projects.

V&V Issues Encountered

The known issues most often cited as problems were: test coverage determination (63%), knowledge validation (60%), real-time performance analysis (33%), and

problem complexity (40%). Other problems cited were: modularity (27%), configuration management (20%), certification (11%), and understandability (10%). The least cited problem was analysis of certainty factors (only seven respondents indicated that certainty factors were used). Every known issue was cited by at least one respondent. The expected system use varied widely (3-2000), while actual system use was relatively good. However, less than half of the respondents provided information, suggesting that actual use was much lower than reported. Of those who responded with an opinion, 96% felt that their expert system was at least as reliable as a typical conventional software system, and 51% felt it was more reliable.

Interview Results

In addition to acquiring written responses to the survey questions, interviews were performed to gather additional data and to clarify questions concerning the written responses. Additional information from these interviews are summarized in this section.

Structural Testing

Based on the survey results, a commonly used evaluation approach was the use of structural testing. This was surprising because the common perception among KBS researchers is that many common forms of structural testing are relatively difficult to apply to expert systems. From the interviews, we learned that although some projects did attempt to measure the actual test coverage (i.e., percentage of rules executed during testing) many others did not actually measure the coverage. Instead, they attempted to develop test cases that would cover all of the knowledge base (or at least the important parts) but made no attempt to measure how well the knowledge base was actually covered. Also, there appeared to be no attempt to cover interactions between knowledge base elements (e.g., rule interactions). Generally, each element was tested as if it were an independent piece of the knowledge base. Some knowledge base developers felt that more formal structural testing would be too much effort and would hinder the development process too much. The interview results suggest that although structural testing was used, it was a very weak form of structural testing (at least compared to, say, branch coverage in procedural software testing).

Experts Developing Expert Systems

It appeared that the expert was heavily relied upon to aid in evaluation of the knowledge base; this subject was probed more deeply during the interviews. The developers felt that a close interaction between the expert and the knowledge base developer was mandatory to successfully develop an expert system. This is not a surprising result and it has been discussed at length in the literature [1]. Many KBS developers feel this interaction is so important that they think the best approach is simply to have the expert develop the system. Though it is important for a knowledge engineer to understand the problem domain and to thoroughly represent that domain [6], it is generally accepted that the domain expert should not be the sole developer of an expert system: this is described in more detail in [7], p.154 as the Knowledge Engineering Paradox: "The more competent domain experts become, the less able they are to describe the knowledge the use to solve problems." There are many problems associated with the development of an expert system by a domain expert. Experts often use knowledge that is so highly compiled and implicit that they have difficulty defining that knowledge explicitly (so a machine can use it). Furthermore, collection of domain knowledge from "introspection" is generally held in doubt by psychologists [3]; that is, experts often don't solve a problem the way that they think they do. Finally, building expert systems often involves building highly complex software systems, systems that require skills and training that domain experts seldom have. Some of these issues were recognized by at least one interviewee who felt that when his group begins to tackle more sophisticated problems, they would need developers with better-developed software and knowledge engineering skills.

Requirements Writing and the Conventional Software Life-Cycle

We anticipated that expert systems were being developed using a much more iterative and less structured life-cycle than the conventional waterfall model. Although the subject of life-cycle models was not intentionally addressed during the interviews, it often came up when discussing requirements. It seems that several respondents associated "requirements" with the conventional waterfall model. They felt very strongly that the conventional approaches to software development, such as the waterfall model, were much too formal and structured for expert systems development. Some even suggested it would be disastrous to apply them to expert systems. For many, this feeling extended to documenting requirements, others simply used a different approach to requirements. For example, in some cases, requirements were not written because it was felt that a requirements document was a formally written paper document that needed to be "approved" before development could proceed. In other cases, an iterative prototyping development effort took place and was followed by documenting system requirements. These requirements were then used to test the system to ensure that it worked as everyone thought it should.

Prototypes vs. Operational Systems

Although we asked respondents to state that their system was either "a prototype" or "operational," we received

indications that this distinction was often difficult to make. For example, responses included "it is both a prototype and operational," or "it is an operational prototype," or "it is just a prototype but we have many users." It seems that some systems are originally intended to be a prototype but are used operationally. Some intentionally approach the development of an operational system by first developing a "prototype" and once the prototype is "certified," it is considered "operational." Others acknowledge there is a danger that a prototype will be used as if it were operational. They have taken steps to ensure that a prototype system that is not accidentally relied upon in an operational setting.

Real-Time Performance Analysis

In our survey, we intended "real-time performance analysis" to refer to the ability to predict the response time for an expert system. That is, the ability to analyze the time performance of the system. However, from the interviews we learned that many interpreted "real-time performance analysis" to mean the ability to get the system to run as fast as desired/necessary. While this is important, it is unclear from the survey and the interviews just how many (if any) of the respondents had quantifiable, rigid needs for expert systems which could generate a response in a guaranteed time frame. Certainly few of the system developers had formally analyzed or documented any "hard" real-time constraints.

Issues Independent of A System Being an Expert System

An important, but difficult, aspect of analyzing expert system development methodology is distinguishing properties of expert systems that are significantly different from properties of conventional software [2]. This is also an important aspect of the analysis of this survey of V&V issues. Several comments appeared to be due more to factors other than the fact that the system being developed was an "expert system." The interviews helped clarify this issue, and the important ones are discussed in this section.

Extensive Use of Prototyping and Rapid Development

The conventional waterfall life-cycle model has proven to be ineffective for conventional software development. Therefore, it is no surprise that developers do not want to use it for expert system development. A more iterative model (e.g., the spiral model) that includes the use of rapid prototyping is being perceived as a better alternative to the waterfall model. "Conventional" software development projects often include the use of prototyping for activities like developing better user interfaces and having developers better understand the problem domain. These kind of issues are not unique to expert system development, but did come up often in the survey, particularly during the interviews.

Small/Simple vs. Large/Complex Systems

Although some of the systems surveyed are fairly large (e.g., 200 person-months), they are generally much smaller than dedicated software development projects (e.g., Shuttle mission control center (MCC), Shuttle flight software, etc.). The systems surveyed seem to be isolated efforts to develop off-line applications for niches for which expert system technology was felt to be very suitable. They were generally systems that were not part of a larger software system, though they are often used in conjunction with a large data processing system (e.g., they receive real-time data from a large data processing system). This allowed the expert system developers to work without many of the constraints imposed on larger systems (e.g., tightly controlled configuration management).

Addressing a Knowledge Engineer Instead of a Programmer

Although we did not intend to gather information on the experience and background of individual expert system developers, we did learn that several respondents involved in developing expert systems are experts in a problem domain without significant programming experience. This fact was important when formulating the detailed recommendations (discussed in [9]).

Issue Summary

It may be the case that the above issues are indeed typical of expert system development projects and that they should be addressed when addressing V&V of expert system problems. However, it should be recognized that they are somewhat different than the other issues that are true of all expert systems regardless of their size and who is developing them. This may point to a need to tailor suggestions for V&V of expert systems to considerations such as the size of the expert system, the experience of the developer, whether the system is embedded in a much larger software system, etc.

Recommendations Based on the Survey

The major goal of this survey was to discover and document the current state of the practice in V&V of expert systems. Based on the survey results, it appears that much can be done to improve the practice. As a starting point, recommendations for improving KBS V&V were drawn from the survey and interview results. These recommendations are separated into two categories: direct recommendations which are directly supported by the survey results and inferred recommendations which can be inferred from the survey results by analyzing relationships among the responses.

Direct recommendations include:

- Develop requirements for expert system verification and validation
- Address most often encountered issues
- Recommend a life-cycle for expert systems development

Inferred recommendations include:

- Address readability and modularity issues
- Address configuration management issue
- Develop criteria to classify expert systems by intended use
- Investigate applicability of analysis tools

Survey Conclusions

The original goal of our survey was to gather data and document the current state-of-the-practice in KBS V&V. The survey and follow-up interviews have given us considerable insight into the kinds of problems that developers have really encountered in developing and verifying expert systems. Many of these problems will require additional work before solutions will be readily available. The analysis of the survey and interviews and the subsequent recommendations can serve as valuable reference for directing future KBS V&V research into those areas which are of the most value to KBS developers and users. In addition, managers of KBS development projects can learn from these results to structure life-cycle approaches for KBS development which are more likely to lead to high quality application software.

SPACE STATION FREEDOM PROGRAM V&V REQUIREMENTS ANALYSIS

There are several software V&V requirements for the Space Station Freedom Program (SSFP) that are contained in SSFP documents. KBS V&V issues were not considered when these requirements were defined so it was felt that they might not be appropriate for the V&V of KBSs. To understand the scope of this problem and how it might be resolved, we defined a task to:

- Identify all SSFP V&V requirements
- Analyze the applicability of the requirements to KBSs
- Make recommendations so that all V&V requirements would apply to KBSs. A recommendation could be to change an existing V&V requirement or to develop a KBS V&V technique that could be used to satisfy a requirement.

(A more detailed discussion of this work is discussed in [10].)

Analysis

From several SSFP documents, we initially identified 93 SSFP V&V requirements which were specific to the technical work of software V&V. That is, we did not consider hardware requirements, general documentation requirements, or logistical requirements such as reporting procedures. Grouping similar requirements together and eliminating some minor duplication resulted in 50 distinct requirements.

We analyzed each of the 50 requirements to answer the following questions:

- What is the intent of this requirement ?
- Does this requirement make sense for a KBS ?
- Is this requirement currently satisfied in the current state-of-the-practice ?
- If it is not in the current state-of-the-practice, is there any inherent reason it could not be satisfied ?
- If there is no inherent reason it can not be satisfied, what is it about KBS development that makes this requirement difficult to satisfy ?

Results

Twenty-seven of the requirements are defined either at a level of generality or at a point in the life-cycle where specific software attributes are indistinguishable and can be applied equally to both KB and conventional software systems. Seven of these requirements can be applied to KBSs using existing processes. Thus, 16 requirements remained that were uniquely difficult or impossible to satisfy for KBSs.

We learned that many requirements that would be difficult to satisfy for KBSs were due to two major factors: "life-cycle model" (four requirements) and "system requirements" (five requirements). The "life-cycle model" factor existed because a general waterfall-type of life-cycle model was assumed to be used for system development. For example, the SSFP configuration management requirements would be difficult to apply to an highly iterative life-cycle by having a high overhead to document and release changes to the system. The "system requirements" issue existed because many of the requirements relied on the existence of a detailed set of requirements that identified many considerations; the general state-of-the-practice definitely does not include the generation of such detailed requirements. For example, there is an SSFP requirement to verify quality requirements yet there is no well-understood way of measuring the quality of a KBS.

The remaining V&V requirements that would be a problem for KBSs are:

- Identification of modules (There is no clear way of identifying "chunks" of knowledge as a module, e.g., a rule grouping.)

- Verifying maintainability (It is not clear what makes an expert system maintainable.)
- Requirements to code mapping (Can not be mapped to modules unless modules can be identified; mapping to individual rules/frames is too difficult.)
- Performance analysis (It is difficult to analyze the response time of non-procedural programs.)
- Path coverage (Paths in the conventional sense do not apply to non-procedural programs, paths in a broader sense are much more difficult to identify in non-procedural programs.)
- IV&V (Because of the heavy reliance on experts to aid in verification, independent verification [without the expert or using a different expert] may not be feasible.)
- Verifying off-the-shelf-components (There are not standards in KBS languages as there is in the standard procedural language, Ada.)

Implication to Other Programs

Most existing programs have V&V standards and guidelines that are similar to the SSFP V&V requirements and were generated with conventional procedural software in mind. An analysis similar to the one summarized here would be necessary to adapt the existing program standards and guidelines so they could be applied to KBSs. This approach would be preferable to generating a separate set of standards and guidelines for KBSs. As with SSFP, it is likely that the majority of standards and guidelines could be applied to KBSs without any difficulty so there would not be much duplication. Also, in practice, it may not be clear where in the system a KBS ends and conventional software begins. It may even be the case that a system that starts out being a KBS might end up being implemented as conventional software or visa versa. So having separate KBS and conventional software V&V standards and guidelines would create many difficulties.

SUMMARY

From the survey that we have performed, we have determined that there are some issues with respect to the state-of-the-practice in V&V of KBSs. We have also learned about common practice as well as problems. From the analysis of SSFP V&V requirements, we have learned that conventional V&V standards and guidelines are not completely applicable to V&V of KBSs. We have also learned that the state-of-the practice in conventional software V&V (as represented by standards and guidelines) is significantly different than the state-of-the-practice in KBS V&V.

REFERENCES

1. Bell, M.Z., "Why Expert Systems Fail," JOURNAL OF THE OPERATIONS RESEARCH SOCIETY, Vol. 36, No. 7, 1985, pp. 613-619.
2. Culbert, C., Riley, G., & Savely, R.T., "An Expert System Development Methodology Which Supports Verification and Validation," In PROCEEDINGS OF ISA 88, Instrument Society of America, Houston TX, 1987.
3. Ericson, K.A., & Simon, H.A., PROTOCOL ANALYSIS, MIT Press, Cambridge MA, 1984.
4. O'Keefe, R.M., & Lee, S., "An Integrative Model of Expert System Verification and Validation," EXPERT SYSTEMS WITH APPLICATIONS: AN INTERNATIONAL JOURNAL, Vol. 1, No. 3, 1990, pp. 231-236.
5. Rushby, J., "Quality Measures and Assurance for AI Software," NASA Contractor Report No. 4187, Houston TX, 1988.
6. Slagle, J.R., & Gardiner, D.A., "Knowledge Specification of an Expert System," IEEE EXPERT, Vol. 5, No. 5, 1990, pp. 29-33.
7. Waterman, D.A., A GUIDE TO EXPERT SYSTEMS, Addison-Wesley, Reading, MA, 1986.
8. Constantine, M.M, & Ulvila. J.W., "Testing Knowledge-Based Systems: The State of the Practice and Suggestions for Improvement," EXPERT SYSTEMS WITH APPLICATIONS: AN INTERNATIONAL JOURNAL, Vol. 1, No. 3, 1990, pp. 237-248.
9. Hamilton, D., Kelley, K., & Culbert, C., "State-of-the-Practice in Knowledge-Based System Verification and Validation," To appear in EXPERT SYSTEMS WITH APPLICATIONS: AN INTERNATIONAL JOURNAL, 1991.
10. "Expert System Verification and Validation Study," RICIS Contract #069, Phase 2 - Requirements Identification, Delivery 2 - Current Requirements Applicability, University of Houston / Clear Lake, 1991.