EXPERIENCES IN IMPROVING THE STATE OF THE PRACTICE IN VERIFICATION AND VALIDATION OF KNOWLEDGE-BASED SYSTEMS

Scott W. French
IBM FSC
Rockville, MD

Chris Culbert
NASA JSC
Software Technology Branch
Houston, TX

David Hamilton
IBM FSC
Houston, TX

ABSTRACT

Knowledge-based systems (KBSs) are in general use in a wide variety of domains, both commercial and government. 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 development life-cycle. However, there are some essential differences between conventional software and KBSs, both in construction and use. The identification of these differences affect the verification and validation (V&V) process and the development of techniques to handle them. The recognition of these differences is the basis of considerable on-going research in this field. For the past three years IBM (Federal Systems Company - Houston) and the Software Technology Branch (STB) of NASA/Johnson Space Center have been working to improve the "state of the practice" in V&V of Knowledge-based systems. This work was motivated by the need to maintain NASA's ability to produce high quality software while taking advantage of new KBS technology. To date, the primary accomplishment has been the development and teaching of a four-day workshop on KBS V&V. With the hope of improving the impact of these workshops, we also worked directly with NASA KBS projects to employ concepts taught in the workshop. This paper describes two projects that were part of this effort. In addition to describing each project, this paper describes problems encountered and solutions proposed in each case, with particular emphasis on implications for transferring KBS V&V technology beyond the NASA domain.

BACKGROUND

Before this project began, KBS V&V was the subject of several ongoing research projects and articles in popular trade journals. Much of this discussion and research was based on several conjectures about differences between KBS V&V and V&V of conventional software. These conjectures were based on theoretical arguments and limited personal experiences. However plausible these conjectures were, no systematic effort had been made to determine the extent to which they impacted industry's ability to deploy reliable KBSs. That is, there was no evidence that the state of the practice in KBS V&V needed any improvement. After all, many successful KBS systems had been developed and, presumably, verified and validated in some fashion.

To understand the state of the practice in KBS V&V, we performed an extensive survey of several KBS V&V projects within NASA, other government agencies, and commercial companies ([5]). This survey validated the conjectures to some extent. It showed that deployed KBSs were generally less accurate or reliable than users and developers expected. We could not determine whether this was because the KBSs were of low quality or because expectations were unrealistically high. In either case, it pointed to a need to

better define and meet accuracy and reliability requirements. More importantly, the survey indicated that the development of KBSs differed significantly from generally advocated practices. For example, less than half of the projects had any form of documented requirements. There were also some indications that KBS V&V was of some difficulty. For example, over sixty percent of the projects indicated that test coverage determination was a particular problem, but through follow-up interviews we learned that many were unfamiliar with existing test coverage techniques. The interviews also showed that many developers were domain experts with little programming experience; this may account for the unfamiliarity with traditional V&V approaches.

Results from the survey were instrumental in providing direction for both long-term and near-term work in the V&V of KBS. Though the survey appeared to justify the need to research new methods for KBS V&V, it also pointed out the need to inform KBS developers about V&V methods that already existed. The near-term direction we chose was to train KBS developers both in known KBS V&V techniques and in conventional V&V techniques (many of which had been shown to be useful in KBS V&V, despite the differences between KBS and conventional software). We sought to educate KBS developers in a way that both convinced them of the importance of V&V and gave them some confidence through hands-on experience with techniques so they could effectively use them. This appeared to be the most immediate way to make a significant impact in the state of the practice in KBS V&V.

We developed a four day workshop, teaching the underlying theory supporting V&V (i.e., why V&V is important), a wide-range of testing techniques, and a set of guidelines for applying these techniques. The material was based on a broad survey of V&V methods and was reviewed by several leading KBS V&V researchers. This material included:*

- a review of basic V&V concepts
- an explanation of the key differences between KBSs and conventional software
- a summary of over fifty V&V techniques
- examples, worksheets and guidelines for the techniques that were considered most useful
- an extensive set of references, cross referenced to each technique and concept

The workshop was taught several times to many NASA KBS developers. Although the results of the workshop have been very favorable (see [8]), the responsibility for applying the material taught lay entirely in the hands of the students. We contacted a small sampling of students several weeks after each course in an attempt to find out how well they were able to apply the material. By far, the most frequent answer was that they had not yet had an opportunity to apply them. (Perhaps the reason that they were able to attend the class was that they were "in between" projects.) To improve the impact of the workshop, we looked for (and found) ongoing KBS projects within NASA that would be willing to apply concepts taught in the workshop. The remainder of this paper describes this work.

PROJECT DESCRIPTIONS

This section describes two projects within NASA that we worked with to develop a KBS V&V approach. Each project fit within different development organizations within mission operations (Space Shuttle and Space Station Freedom). The first group, called users, was composed primarily of flight controllers who develop applications to automate and assist flight control activities within the mission control center. The second group, facility developers, developed the mission control complex itself. This development included both development of the key parts of the mission control center and incorporation of users group applications as part of the control center baseline. We worked with one project from each of these groups. We worked with a users group that developed a monitoring application called the Bus Loss Smart System or BLSS. We also worked briefly with facility developers for the space station mission

---

* See [6] and [7] for a discussion of the workshop contents.

control complex to develop criteria for assessing model-based user applications for inclusion into the control center baseline. This section gives insight into these projects by describing their environment, procedures and problems.

OVERVIEW OF THE USERS GROUP

In preparing to work with these groups, we taught a condensed (one day) version of the workshop to both flight controllers and application developers. There was a two-fold objective in teaching this workshop: (1) understand the kinds of problems they are working on and (2) teach them techniques that address those problems. Most of the problems they faced related directly to how these user applications are built. By this we mean that the basic development practices that support V&V were not practiced. We found that in most cases, the flight controller is the expert, developer and user. Inspections are viewed as being too expensive (both in dollars and time) and are, therefore, not done. Requirements were considered more of an enemy than a friend. For this reason, they rarely document the requirements that did exist. Their systems are viewed as prototypes, not as flight certified mission control applications. Becoming certified means that the application is added to the baselined set of mission control center applications. Few user applications had become certified. Testing emphasizes the functionality and user-interface aspects of their systems and not other important kinds of correctness such as safety and resource consumption.

Based on this insight into their development environment, several techniques were presented to the group. Most of these focused on helping them specify what the system should do and how it should do it. The following list of techniques was presented:

- Inspections ([18] and [2])
- Cause-Effect Graphing ([18], [19], [20] and [21])
- State Diagrams ([23])
- Decision Tables ([17])
- Assertion Analysis ([10])
- Object-oriented Analysis ([23], [11] and [26])
- Connectivity Graphs ([12] and [22])
- Petri Nets ([22], [15] and [1])
- Minimum Competency ([24] and [25])
- Pre/Post Conditions ([4], [3], [14], [9] and [13])

Bus Loss Smart System (BLSS)

BLSS is a flight control application designed to monitor electrical equipment anomalies, loss or malfunction within key electrical systems onboard the orbiter. Since it was a prototype it only acted as an assistant to the flight controller in analyzing telemetry data sent from the orbiter to the ground.

Like most of the other flight control applications it was developed using G2. Schematics of the electrical systems were created using G2 graphics capabilities. When anomalies were discovered in the electrical system, the flight controller was notified by BLSS via messages and highlighted items on the schematic. The flight controller then interacted with BLSS by indicating whether the anomaly should be ignored or further analysis was needed. BLSS then performed some deeper investigation into the anomaly.

Two primary methods were being used for testing BLSS. Both were system or "black-box" methods. With the first method, the flight controller supplied the programmer with simulation "scripts" (very much like operational scenarios). A simulation was then run based on this script to see that required outputs (as stated on the script) were generated. These simulations used actual telemetry data as supplied by the mission control complex.

The second method was also a simulation, but not a simulation that uses telemetry data from the mission control complex. Instead, special rules were inserted into the knowledge-base that caused certain events to happen at specific times while running in G2 simulation mode. A series of ten or so of these special cases had been developed to test the system. If the system passed all of these special cases, then testing was considered to be done.

FACILITY DEVELOPERS GROUP OVERVIEW

The purpose of the "models assessment" effort was to capitalize on existing Space Station Freedom (SSF) advanced automation projects. In these advanced automation projects, prototype systems were built in order to prove or demonstrate the ability to automate SSF operations using advanced technology. These prototype systems were not intended to be used operationally (i.e., the were not to be used directly by an SSF flight controller during actual flight operations). However, rather than building operational tools by completely re-implementing these systems, it was hoped that the prototypes could be turned into operational tools through additional development and/or additional V&V.

Models Assessment

The models were evaluated according to their usefulness and correctness. The usefulness of a prototype was judged by how well it met the needs of its target flight controllers. This involved more than just the functionality of the prototype. Issues such as usability were also considered. Judging the correctness of a prototype depended on its current level of correctness and the additional effort required to make the prototype sufficiently correct. Factors that impacted the assessment of correctness for a prototype were their lack of good requirements, their need to be stand-alone applications (i.e., the failure of one application should not affect another), their required role and function (e.g., advisor fault detection, diagnosis, etc.) and the role of their experts (users/experts may or may not be the developer).

APPROACH

Both projects had been studied in sufficient detail to define a V&V approach. In this section we describe our approach for each of these projects and the specific activities implementing that approach.

Bus Loss Smart System.

The most urgent need for the BLSS seemed to be to develop a good set of requirements that supported testing. The requirements that did exist lacked sufficient detail (i.e., they were very ambiguous) to support testing and maintenance. They also failed to address other important aspects of requirements such as safety, resource consumption, user profiles, etc.. Fortunately, most of the information needed for their requirements did exist. Our approach was to collect these requirements into a complete document based on DOD Std 2167A that would support testing. Our objective was to demonstrate the value of following standards and teach them how to write good requirements.

To complement the DOD 2167A format we provided the flight control group with a requirements handbook that describes the format of the document, the characteristics of good requirements, a step-by-step requirements definition method and a verification method for requirements. The approach we advocated was to define the overall goal of the system, the high-level tasks the system must perform (separated into competency and service tasks as discussed in [24] and [25]), user profiles, operational scenarios, and a state model for each task (see [4]). The tasks were then integrated through the definition of pre/post conditions and task invariants. Another urgent need for the BLSS was to have a good design specification that supported verification. The BLSS developers began defining this specification using an outline based on the DOD Std 2167A. We helped them incorporate a data dictionary based on the state

models described in both the requirements and the design along with pre/post conditions for each procedure in the implementation. We had also planned to help them use inspections as a way to increase the quality of these specifications.

The last area where we are helped with BLSS was during user acceptance testing. This was different from the certification testing we described previously. This is primarily a "black-box" test activity performed by BLSS users to convince themselves that the system works. We had convinced them to use a statistical testing approach based on their simulation scripts. Simulation scripts were to be created that include at least one failure for each bus being monitored. The tester would keep track of the number of BLSS errors (based on severity - failing to identify a bus failure would be the most severe error) versus how long BLSS is in operation. Using these statistics we would apply a reliability model to quantify the quality of BLSS, in order to "certify" it.

Results

The BLSS development project had not yet been completed at the end of our consulting engagement. A draft requirements document and a draft design document had been developed but formal testing had not yet begun. Though they expressed great interest in the V&V approach that we had defined for them, it does not appear that they have continued following it as well as we had hoped.

Models Assessment

The general V&V approach defined for the Models Assessment was as follows. The first step was to develop requirements for each prototype. The requirements format developed for the BLSS project was to used as a base for a models assessment requirements format. Requirements were to be divided into requirements for evaluating prototype usefulness and requirements for evaluating prototype correctness. Initially, only the requirements supporting usefulness evaluation needed to be written. Then, if they were deemed useful, additional requirements would need to be documented to support evaluation of the correctness of the prototype. The initial requirements should include a description of the current operation of the system with emphasis on the problem(s) that the prototype was intended to address, the goals of the prototype (e.g., rapid diagnosis of faults or comprehensive identification of every possible failure condition) and a high-level description of the user interface to the system. This needed only be high level at this point, since the user would have the opportunity to interact with the tool and judge, firsthand, the usefulness of the interface.

Once the prototype had been deemed useful, the more difficult task of assessing correctness would begin. At this point, the tool should no longer be considered a prototype because it is being "certified" for operational use. There are two major types of correctness to be considered: safety and minimal functionality. With regard to safety, we wanted to show that the failure of any application would not interfere with other control center applications. For minimal functionality we wanted to demonstrate that both minimal service and minimal competency requirements are satisfied. Competency requirements (see [24] and [25]) define the "knowledge" or "intelligent ability" of the system. Service requirements would be all requirements that were not competency requirements. These would include, but are not be limited to, input and output formats, response time, processor the tool should run on, etc..

The general approach for this phase of V&V of the tool would be to validate the requirements by inspection, require the developer to verify the tool against the requirements, and then perform final validation via statistical testing. Statistical testing would involve running the tool in an operational environment for some period of time, recording any failures that might occur. This failure information will be used to predict an expected mean time in between failures (MTBF) of the system in operational use. We considered measuring MTBFs for safety, minimal service and minimal competency requirements.

Results

Unfortunately, we did not have an opportunity to apply the approach because no prototype made it to the point of being assessed for correctness. The primary reason for this was that much of the SSF architecture had changed by the time the prototype was ready for evaluation. So the only assessment that could be made was whether the prototype was a useful automation demonstration.

SUMMARY AND IMPLICATIONS FOR FURTHER TECHNOLOGY TRANSFER

Although it is felt that the material and approaches developed in this project have great potential to improve the state of the practice in KBS V&V (and in all software V&V), the results to date have only been moderately successful at best. Our initial concern and risk was that projects would be unwilling to try a sophisticated V&V approach because of the perceived cost. This is because we were targeting KBS projects which tend to be small and follow a rapid development (i.e., prototyping) process. To mitigate this risk, we strove to develop a streamlined V&V approach that involved a small number of techniques that had the best cost/benefit ratio (i.e., requirements and inspections) and/or directly addressed the problem of certification (i.e., statistical testing). The initial interest we received from the projects led us to believe that we were successful in this aspect.

Another problem that we did not fully appreciate was the length of time that would be required for a project to become self sufficient in following an established V&V approach. Our consulting engagement needed to be much longer than the six to eight months that we had, so that we could have followed each project to successful conclusion of at least a first release of the system.

A final problem in transferring technology to the target projects was the lack of a defined and enforced process. KBS projects have historically been small and involve rapid, highly iterative, development. This is true of KBS projects inside and outside of NASA ([5]). (And it may be true for most software development projects outside of NASA.) Because of this, there was no way for our suggested V&V approach to be officially adopted and enforced beyond our consulting engagement.

This project has important implications for the transfer of software engineering technology outside of NASA. NASA's software engineering methods and technology are among the best and NASA has a good reputation for building high quality software; therefore, NASA has much that could benefit others who do software development. However, many commercial projects are unlike the typical large, well-defined and safety critical NASA projects. KBS projects have many similarities with the typical commercial projects in that they are usually small, ill-defined applications that must be developed quickly. This does not necessarily imply that NASA's software technology is unsuitable for commercial projects. Because there is also a growing realization that more discipline and rigor is needed in many industries where software and KBSs are becoming key parts of safety critical systems, such as in medical devices.

Just as traditional and well-accepted V&V methods had to be adapted to fit the KBS projects discussed in this paper, NASA's software engineering methods will need to be adapted to fit the commercial software development environment. But, as evidenced by the survey discussed in this paper, such methods do appear to be needed. Also, based on the experiences discussed in this paper, transferring these adapted methods will require a systematic concerted effort. Simply making the techniques available to interested commercial software developers, as we tried to do with our KBS V&V workshop, will likely have minimal impact. These conclusions are consistent with the experiences of other attempts to transfer software engineering technology, such as those reported in [9].

REFERENCES

1. Becker, S.A. and Medsker, L., "The Application of Cleanroom Software Engineering to the Development of Expert Systems." Heuristics: The Journal of Knowledge Engineering. Quarterly Journal of the International Association of Knowledge Engineers (IAKE) Volume 4 Number 3, pp. 31-40. Fall 1991.

2. Fagan, M.E., "Design and Code Inspections to Reduce Errors in Program Development." IBM Systems Journal Volume 15 No. 3 pp. 182-211, 1976

3. Gries, D., The Science of Programming. Springer- Verlag New York, Inc. 1981.

4. Hamilton, D. and French, S., "A Design Language for Testable Expert Systems." Workshop Notes from the Ninth National Conference on Artificial Intelligence - Knowledge Based Systems Verification, Validation and Testing. July 17, 1991.

5. Hamilton, D., Kelley, K. and Culbert, C., "KBE V&V - State-of-the-Practice and Implications for V&V Standards." Workshop Notes from the Ninth National Conference on Artificial Intelligence - Knowledge Based Systems Verification, Validation and Testing. July 17, 1991.

6. Hamilton, D. and French, S.W., Workshop on Verification and Validation of Expert Systems. University of Houston/Clear Lake RICIS Contract #69 Deliverable #2, February 1992.

7. Hamilton, D., French, S.W. and Culbert, C., "An Approach to Improving the State-of-the-Practice in Verification and Validation of Expert Systems." Workshop Notes for the AAAI-92 Workshop on Verification and Validation of Expert Systems. July 16, 1992.

8. Hamilton, D. and French, S.W., Workshop on Verification and Validation of Expert Systems - Final Report. University of Houston/Clear Lake RICIS Contract #69 Deliverable #5, August, 1992.

9. Hamilton, D. and French, S.W., "Advancing the State of the Practice in Formal Verification", proceedings of AIAA Computing in Aerospace 9, October 19-21, 1993

10. Hoare, C.A.R., "Introduction to Proving the Correctness of Programs." ACM Computing Surveys. pp. 331-353, September 1976.

11. Howden, W.E., "Comments Analysis and Programming Errors." IEEE Transactions on Software Engineering. Volume 16 Number 1 pp. 72-81, January 1990.

12. Korson, T. and McGregor, J.D., "Understanding Object-oriented: A Unifying Paradigm." Communications of the ACM. Volume 33 No. 9 pp. 40-60 September 1990.

13. Landauer, C.A., "Correctness Principles for Rule- Based Expert Systems." Expert Systems with Applications. Pergamon Press. Volume 1 Number 3 pp. 291-316, 1990.

14. Linger, R.C., Mills H.D. and Witt, E.I., Structured Programming: Theory and Practice. Addison-Wesley Publishing Company 1979.

15. Liskov, B. and Guttag, J., Abstraction and Specification in Program Development. McGraw-Hill Book Company 1986.

16. Liu, N.K. and Dillon, T.. "An Approach Toward the Verification of Expert Systems Using Numerical Petri Nets." International Journal of Intelligent Systems. Volume 6 Number 3, pp. 255-276. June 1991.

17. Montalbano, Decision Tables. Science Research Associates, 1974 18. Myers, G.J.. The Art of Software Testing. John Wiley & Sons, Publishing 1979.

19. Myers, G.J.. Software Reliability Principles and Practices. John Wiley & Sons, Publishing 1976.

20. Myers, G.J.. Reliable Software Through Composite Design. Mason/Charter Publishers 1975.

21. Myers, G.J.. Composite/Structured Design. Litton Educational Publishing 1978.

22. Nazareth, D.L.. An Analysis of Techniques for Verification of Logical Correctness in Rule-Based Systems. pp. 80-136, Catalog #8811167-05150. UMI Dissertation Service, Ann Arbor, MI 48106, 1988.

23. Rumbaugh, J.. Object-Oriented Modeling and Design. Prentice-Hall, Inc. 1991.

24. Rushby, J. and Crow, J.. Evaluation of an Expert System for Fault Detection, Isolation and Recovery in the Manned Manuevering Unit. Final Report for NASA contract NAS1-182226 (NASA/Langley)

25. Rushby, J.. Quality Measures and Assurance for AI Software. NASA contractor report #4187, NASA Langley.

26. Yourdon, E. and Coad, P.. Object-Oriented Analysis . Prentice Hall, Inc. Englewood Cliffs, NJ 1990.