Experiences in Improving the State of the Practice in Verification and Validation of Knowledge-Based Systems

Scott W. French
IBM Federal Systems Company
3700 Bay Area Blvd./MC 6402
Houston, TX 77058

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

David Hamilton
IBM Federal Systems Company
3700 Bay Area Blvd./MC 6402
Houston, TX 77058

Abstract

Knowledge-based systems (KBS) are in general use in a wide variety of domains. 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 (i.e., procedural) software and knowledge-based systems, both in construction and use. The identification of these differences affect the verification and validation process and the development of techniques to handle them 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 verification and validation of Knowledge-based systems. To date, the primary accomplishment has been the development and teaching of a four-day workshop on verification and validation of knowledge-based systems. With the hope of improving the impact of these workshops, we are now working directly with NASA KBS projects to employ concepts taught in the workshop. This paper describes two projects currently being assisted as part of this effort. In addition to describing each project, this paper describes problems encountered and solutions proposed in each case.

Background

Before attempting to improve the state of the practice in verification and validation of KBS, the state of the practice needed to be defined. Many conjectures had been made within the research community and despite their plausibilities, these remained primarily theoretical. Therefore, we conducted a survey of the state of the practice in KBS V&V (see [5] for a detailed presentation of the survey results).

Results from the survey were instrumental in providing direction for both near-term and long-term work in the V&V of KBS. The near-term direction we chose was 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 that they could actually do it. Therefore, 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 (see [6] and [7] for a discussion of the workshop contents).

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. To improve the impact of the workshop, then, we looked for (and found) ongoing KBS projects within NASA that would be willing to more directly 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 are working with to develop a KBS V&V approach. Each project fits within different development organizations within mission operations (shuttle and Space Station Freedom). The first group, called users, is composed primarily of flight controllers. Developing applications to automate and assist flight control activities within the mission control center is their objective. The facility developers group is developing the mission control complex itself. This development includes 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 are working with one project from each of these groups. For one of these projects we are working with a users group that is developing a monitoring application called the Bus Loss Smart System or BLSS. In the other project we are working with the facility developers for the space station mission control complex to develop criteria for assessing model-based user applications that are to be incorporated 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: understand the kinds of problems they are working on and teach them techniques that address those problems. Most of the problems they face relate directly to how these user applications are built. By this we mean that the basic development practices that support V&V are 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 are considered more of an enemy than a friend. For this reason, they rarely document the requirments that do 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 ever become certified. Testing emphasizes the functionality and user-interface aspects of their systems while ignoring 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)

The BLSS is a flight control application designed to monitor electrical equipment anomalies, loss or malfunction within key electrical systems onboard the orbiter. Since it is a prototype it only acts 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 is being developed using G2. Schematics of the electrical systems are created using G2 graphics capabilities. When anomalies are discovered in the electrical system the flight controller is notified by the BLSS via messages and highlighted items on the schematic. The flight controller then interacts with the BLSS by indicating whether the anomaly should be ignored or further analysis is needed. The BLSS then performs some deeper investigation into the anomaly.

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

The second method is also a simulation, but not a simulation that uses telemetry data from the mission control complex. Rather, special rules are inserted into the knowledge-base that cause certain events to happen at specific times while running in G2 simulation mode. A series of ten or so of these special cases have been developed to test the system. If the system passes all of these special cases, then testing is done.

Facility Developers Group Overview

The purpose of the "models assessment" effort is 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; they are prototype systems. These prototype systems were not intended to be used operationally[1]. However, rather than building operational tools by completely re-implementing these systems, it is hoped that the prototypes can be turned into operational tools through additional development and/or additional V&V.

Model Assessment

The purpose of the "models assessment" effort is to capitalize on existing Space Station Freedom (SSF) advanced automation projects by evaluating their usefulness and correctness. The usefulness of a prototype is judged by how well it meets the needs of its target flight controllers. This involves more than just the functionality of the prototype. Issues such as usability are also considered. Judging the correctness of a prototype depends on its current level of correctness and the additional effort required to make the prototype sufficiently correct. Factors that impact the assessment of correctness for a prototype are 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 have 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 is to develop a good set of requirements that support testing. The requirements that do exist lack sufficient detail (i.e., they are very ambiguous) to support testing and maintenance. They also fail to address other important aspects of requirements such as safety, resource consumption, user profiles, etc.. Fortunately, most of the information needed for their requirements does exist. Our approach is to collect these requirements into a complete document based on DoD Std 2167A that supports testing. Our objective is to demonstrate the value of following standards and teach them how to write good requirements.

[1] Being used operationally means they are directly used by an SSF flight controller during actual operations. That is, in flight and not just in a ground simulation.

To complement the DoD 2167A format we are providing 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 are advocating is to define the overall goal of the system, define the high-level tasks the system must perform (separated into competency and service tasks – see [24] and [25]), define who the users are and the expected use of the system (operational scenarios), build a state model for each task (see [4]), define specific task requirements, integrate the tasks through the definition of pre/post conditions and task invariants.

Another urgent need for the BLSS is a good design specification that supports verification. The BLSS developers are currently defining this specification using an outline based on the DoD Std 2167A. We are helping 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 are also planning to help them use inspections as a way to increase the quality of these specifications.

The last area where we are helping with the BLSS is during user acceptance testing. This is different from the certification testing we described previously. This is primarily a "black-box" test activity performed by the BLSS users to convince themselves that the system works. We have convinced them to use a statistical testing approach based on their simulation scripts. Simulation scripts are going to be created that include at least one failure for each bus being monitored. The tester keeps track of the number of BLSS errors (based on severity - failing to identify a bus failure is the most severe error) versus how long the BLSS is in operation. Using these statistics we will apply a reliability model to quantify the quality of the BLSS. As of yet we have not defined what this number should be. We also have not applied the reliability model yet since many of the simulations still need to be defined and executed.

Model Assessment

We are working with the model assessment team to construct a general V&V approach. The first step in this approach is to develop requirements for each prototype. The requirements format developed for the BLSS project will be used as a base for a models assessment requirements format. Requirements will be divided into requirements for evaluating prototype usefulness and requirements for evaluating prototype correctness. Initially only the requirements supporting usefulness evaluation need to be written. Then, if they are deemed useful, additional requirements will 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 is 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 need only be high level at this point, since the user will have the opportunity to interact with the tool and judge, firsthand, the usefulness of the interface.

Once the prototype has been deemed useful, the more difficult task of assessing correctness begins. 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 want to show that the failure of any application will not interfere with other control center applications. For minimal functionality we want 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 are all requirements that are not competency requirements. These include, but are not 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 is 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 will 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 are considering measuring MTBFs for safety, minimal service and minimal competency requirements. Minimal MTBFs will be decided upon (based on the functional area being automated) and if the predicted MTBFs exceed the desired MTBFs, the system will be accepted for operational use[00].

Failure Environment Analysis Tool (FEAT)

FEAT is a baselined control center tool that processes directed graph (digraph) representations of cause-effect relationships in a system. The cause-effect relationships may be functional or logical. FEAT can simulate the effects of user-input events (i.e., predict failures from faults) and diagnose the fault responsible for a failure. Each digraph is mapped to a schematic of the physical system being modelled and will be incorporated into system requirements. As with anything generated by hand, there are many potential types of errors that could occur. These include cause-effect relationships that are missing in the digraph, cause-effect relationships that are incorrectly stated in the digraph, the mapping from the digraph to the schematic is incorrect and the digraph is structurally inconsistent (i.e., there is a logical inconsistency or contradiction in the digraph).

We are beginning to investigate V&V approaches that could be used to detect and/or prevent these and other types of errors (the full paper will have more information on this subject).

Summary

The initial response from the projects described is very favorable. Using the results from these early projects, it is hoped that a more permanent working relationship between KBS developers and the V&V team can be established. We are currently in the early stages of pursuing additional work of this type at all of the NASA centers.

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

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

11. 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.

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

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

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

15. 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.

16. Lockheed Engineering and Sciences Company. *FEAT – Failure Environment Analysis Tool: Software User's Guide.* NASA Contract # NAS 9-17900, February 1993.

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.