NUREG/CR-6316
SAIC-95/1028
Vol. 3

# Guidelines for the Verification and Validation of Expert System Software and Conventional Software

Survey and Documentation of Expert System Verification and Validation Methodologies

Prepared by
E. H. Groundwater, L. A. Miller, S. M. Mirsky

Science Applications International Corporation

Prepared for
U.S. Nuclear Regulatory Commission

and

Electric Power Research Institute

NUREG/CR-6316
SAIC-95/1028
Vol. 3

# Guidelines for the Verification and Validation of Expert System Software and Conventional Software

Survey and Documentation of Expert System Verification and Validation Methodologies

Manuscript Completed: February 1995
Date Published: March 1995

Prepared by
E. H. Groundwater, L. A. Miller, S. M. Mirsky

Science Applications International Corporation
1710 Goodridge Drive
McLean, VA 22102

**Prepared for**
**Division of Systems Technology**
**Office of Nuclear Regulatory Research**
**U.S. Nuclear Regulatory Commission**
**Washington, DC 20555-0001**
**NRC Job Code L1530**

**and**

**Nuclear Power Division**
**Electric Power Research Institute**
**3412 Hillview Avenue**
**Palo Alto, CA 94303**

## ABSTRACT

This report is the third volume in the final report for the Expert System Verification and Validation (V&V) project which was jointly sponsored by the Nuclear Regulatory Commission and the Electric Power Research Institute. The ultimate objective is the formulation of guidelines for the V&V of expert systems for use in nuclear power applications. The purpose of this activity was to survey and document techniques presently in use for expert system V&V.

The survey effort included an extensive telephone interviewing program, site visits, and a thorough bibliographic search and compilation. The major finding was that V&V of expert systems is not nearly as established or prevalent as V&V of conventional software systems. When V&V was used for expert systems, it was almost always at the system validation stage after full implementation and integration usually employing the non-systematic dynamic method of "ad hoc testing." There were few examples of employing V&V in the early phases of development and only weak sporadic mention of the possibilities in the literature. There is, however, a very active research area concerning the development of methods and tools to detect problems with, particularly, rule-based expert systems. Four such static-testing methods were identified which were not discovered in a comprehensive review of conventional V&V methods in an earlier task.

## TABLE OF CONTENTS

ABSTRACT ..... iii

EXECUTIVE SUMMARY ..... ix

1.0 INTRODUCTION ..... 1
- 1.1 Purpose and Scope of Activity 2 ..... 1
- 1.2 Report Organization ..... 1

2.0 TECHNICAL APPROACH ..... 3
- 2.1 Overall Approach ..... 3
- 2.2 Telephone Interviews and Data Collection ..... 3
- 2.3 Site Selection and Visits ..... 12
- 2.4 Technique Characterization and Analysis ..... 17

3.0 REFERENCE LIFECYCLE FOR THIS ACTIVITY ..... 23

4.0 EXPERT SYSTEM V&V TECHNIQUE DESCRIPTIONS ..... 25
- 4.1 Types of Techniques ..... 25
- 4.2 Requirements and Design Testing ..... 25
- 4.3 Static Testing ..... 29
- 4.4 Dynamic Testing ..... 40
- 4.5 The State-of-the-Art ..... 53

5.0 AUTOMATED TOOLS ..... 55
- 5.1 Syntax Checking Tools ..... 55
- 5.2 Semantic Checking Tools ..... 63
- 5.3 Knowledge Acquisition/Refinement Tools ..... 64
- 5.4 Intelligent Compilers ..... 65
- 5.5 Other Dynamic Testing Tools ..... 65
- 5.6 Summary on Automated Tools ..... 66

6.0. CATEGORIZATION AND ANALYSIS OF TECHNIQUES AND TOOLS ..... 67
- 6.1 Components of Expert Systems ..... 67
- 6.2 Expert System Faults: Anomalies or Invalidities ..... 70

7.0. CONCLUSIONS ..... 75

APPENDIX A. BIBLIOGRAPHY ..... 77

# List of Figures

Figure ES-1 Classes of V&V Techniques Which Have Been Applied to Conventional Systems . . . x

Figure 2.1-1 USNRC/EPRI Expert System V&V Project Detailed Activity 2 Work Plan . . . 4

Figure 2.2-1 USNRC/EPRI V&V Interview Questionnaire . . . 9

Figure 2.2-2 Survey Form: Nuclear Industry Expert Systems that Have Been Tested . . . 13

Figure 2.2-3 Activity 2 Telephone Interviews . . . 14

Figure 2.4-1 Tools/Techniques Worksheet . . . 20

Figure 3.0-1 Relationship of V&V Activities to Generic Project Activities . . . 24

Figure 4.1-1 Classification of V&V/Testing Techniques . . . 26

Figure 6.2-1 Comprehensive Expert System V&V Matrix . . . 71

## List of Tables

Table 2.2-1 Persons Contacted for Telephone Interviews . . . . . 5

Table 2.3-1 Preliminary Site List for Visit Evaluation . . . . . 15

Table 2.3-2 Site Selection Criteria . . . . . 16

Table 2.3-3 Relative Site Rankings Using Site Selection Criteria . . . . . 18

Table 2.3-4 Recommended Site Visits Based on Site Selection Criteria Ranking . . . . . 19

Table 4.2-1 Requirements and Design V&V Techniques Applied to Expert Systems . . . . . 27

Table 4.3-1 Static Testing V&V Techniques Applied to Expert Systems . . . . . 30

Table 4.3-2 Types of Syntactic Errors Found by Automatic Rule Base Syntax Checkers . . . . . 39

Table 4.4-1 Dynamic Testing V&V Techniques Applied to Expert Systems . . . . . 41

Table 5.0-1 Description of Automated Tools for Expert System V&V . . . . . 56

Table 6.1-1 Components and General Features of Knowledge Based Systems . . . . . 68

Table 6.1-2 Components and Typical Testing-Related Features of Knowledge-Based Systems, with Testing Recommendations . . . . . 69

## EXECUTIVE SUMMARY

This report is the third volume in the final report for the Expert System Verification and Validation (V&V) project, which was jointly sponsored by the Nuclear Regulatory Commission (USNRC) and the Electric Power Research Institute (EPRI). The ultimate objective is the formulation of guidelines for the V&V of expert systems for use in nuclear power applications. The purpose of Activity 2 was to survey and document techniques for expert system V&V. The survey used the results of Activity 1, a survey of techniques for conventional software V&V, to determine which of these techniques are being applied to expert systems, and what new techniques have been developed solely for expert system V&V.

The survey effort included: 1) an extensive telephone interviewing campaign to over 130 points of contact, 2) site visits to nine institutions conducting research in or applying expert system V&V, and 3) the collection of an extensive library of well over 300 bibliographic references. The survey encompassed work done both within the nuclear power industry and in other industries as well. Contacts included corporations, universities, government agencies, and utilities. Within the last four to five years, there has been an explosive growth of interest and work in the field. It has now reached a level of maturity where expert system V&V techniques are being implemented in automated tools and being applied to operational expert systems development and maintenance efforts.

As can be seen in Figure ES-1, many of the classes of V&V techniques identified in Volume 2 as being applied to conventional software systems are also being researched for, or applied to, expert systems. This is particularly true in the areas of Static Testing (tests performed directly on the code itself) and Dynamic Testing (tests performed by running the code and evaluating the results). Fewer formal techniques are applied during the Requirements and Design phases of expert systems development (only five out of ten possible methods) and then only infrequently. This is primarily because the activities performed during these phases for expert systems are usually informal themselves. Requirements and Design documents for expert systems are often not written at all, or written after-the-fact, and thus cannot be used as a basis for V&V activities. When they are written, usually no more is done with them than to review them and, possibly, trace requirements to design elements.

Fifteen of 58 possible Static Testing techniques were researched or applied for expert systems (including four new ones). Most of the work in Static Testing of expert systems has focused on the development of automated tools to perform sophisticated syntactic checking of rule bases. The types of errors that may be found by such checkers include redundant or subsumed rules (one rule's conditions are a subset of another's), rule cycles (there is a path from a rule back to itself), unreachable or dead-end rules, inconsistent rules, and incompleteness (e.g., not all possible input values are covered). Some of the rule base checkers will perform semantic checks of the rule base using meta-constraints defined by the programmer, and others will perform checking on the fly during knowledge acquisition and/or refinement of the rule base. Other work in Static Testing has included conducting various kinds of inspections (e.g., structured walk-throughs and expert panel reviews), performance of dependency analyses of the output values on the inputs, and attempts at applying program proving techniques. A point of view becoming strongly accepted is that it may not be as vital to prove that a safety-critical expert system is totally error-free as it is to prove that if it fails, it will not fail badly (i.e., compromise safety).

In Dynamic Testing, there is a wide range of activities: 38 of 67 techniques have been researched or applied to expert systems. The state-of-the-art in the operational expert system world is still Ad Hoc Testing, or defining test cases at whim, with no systematic guidance. Newer work has focused on more systematic methods for specifying test case sets, such as Structural Testing (attempting to cover all rules or rule paths in the expert system), Random Testing (attempting to cover a representative sample of the possible inputs), and Performance Testing (to assure timing, memory, and other constraints are met). Some operational expert systems, such as those developed for safety-related

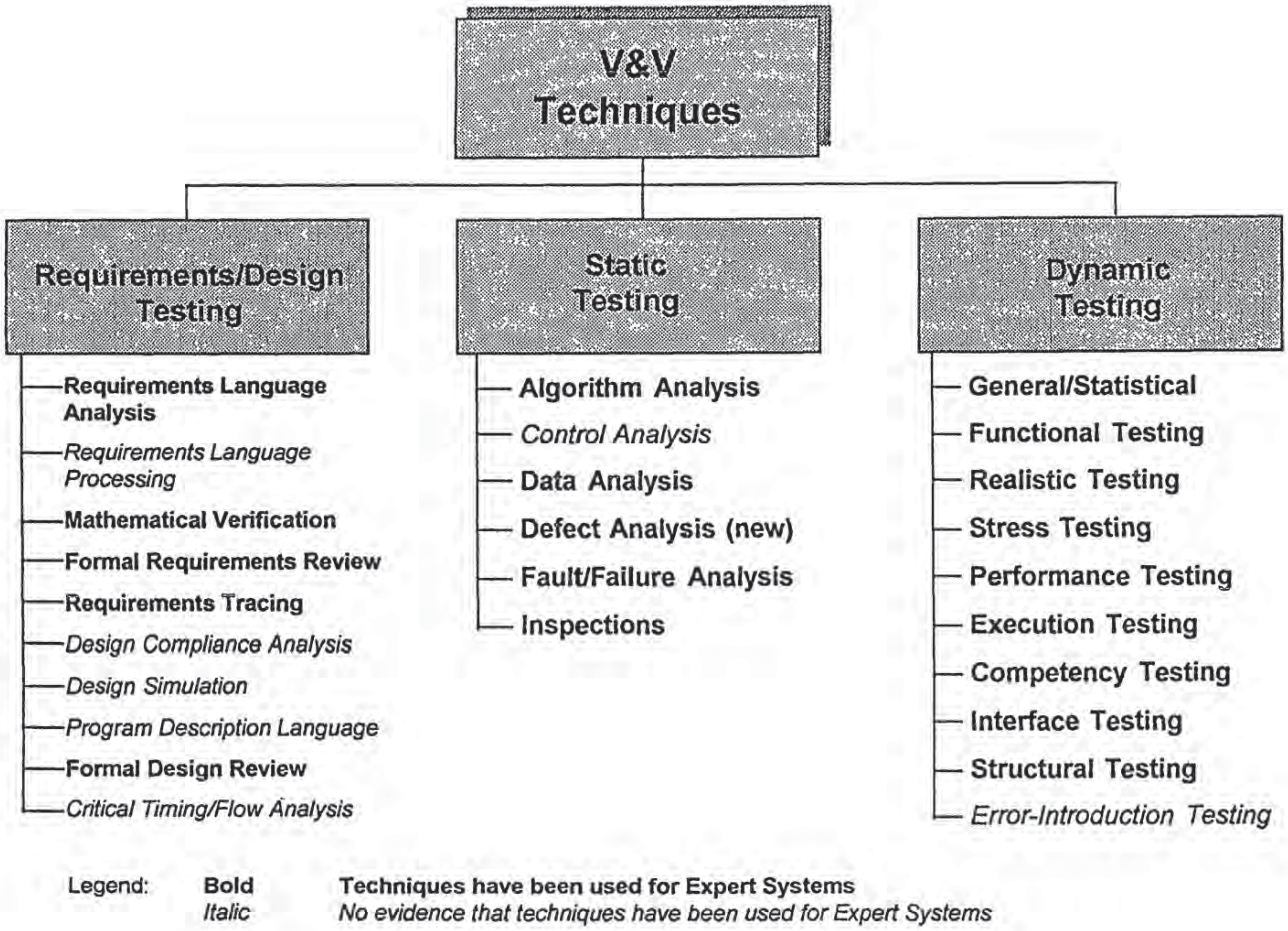

V&V Techniques
Requirements/Design Testing
Requirements Language Analysis
Requirements Language Processing
Mathematical Verification
Formal Requirements Review
Requirements Tracing
Design Compliance Analysis
Design Simulation
Program Description Language
Formal Design Review
Critical Timing/Flow Analysis
Static Testing
Algorithm Analysis
Control Analysis
Data Analysis
Defect Analysis (new)
Fault/Failure Analysis
Inspections
Dynamic Testing
General/Statistical
Functional Testing
Realistic Testing
Stress Testing
Performance Testing
Execution Testing
Competency Testing
Interface Testing
Structural Testing
Error-Introduction Testing
Legend: Bold Techniques have been used for Expert Systems
Italic No evidence that techniques have been used for Expert Systems

**Figure ES-1 Classes of V&V techniques which have been applied to conventional systems**

functions (e.g., NASA space shuttle diagnostics), do undergo various forms of Realistic Testing using scenario files, simulators, or actual field conditions. Lastly, there are a few automated tools to support generating, managing, or scoring test cases.

Upon analysis of the V&V techniques being applied to expert systems, it was found that there is sufficient coverage across all the components of expert systems and across all error types (static vs. dynamic, anomalies vs. invalidities). The challenge is in selecting the appropriate combination of techniques to use for performing V&V on a particular expert system that is both effective and cost efficient.

# 1. INTRODUCTION

This report is the third volume in the final report for the Expert System Verification and Validation (V&V) project. The ultimate objective is the formulation of guidelines for V&V of expert systems for use in nuclear power applications. This work is jointly sponsored by the Nuclear Regulatory Commission (USNRC) and the Electric Power Research Institute (EPRI).

## 1.1 Purpose and Scope of Activity 2

The purpose of Activity 2 was to survey and document techniques for expert system V&V. This report is a companion to Volume 2 that surveys techniques for conventional software V&V. As will be seen, there is (and should be) considerable overlap in the techniques being applied to both types of software. Thus, this report will reference and draw upon the contents of Volume 2 considerably.

The survey included both techniques being applied in the field to operational expert systems and those being researched in Artificial Intelligence (AI) laboratories. With the help of Dr. John Bernard, from the Massachusetts Institute of Technology (MIT), we surveyed V&V techniques being applied to expert systems for nuclear power applications (Bernard & Washio, 1989). However, the survey also encompassed work in other fields such as space operations, manufacturing, military, and other utilities. We contacted a diverse range of organizations including government agencies and laboratories, universities, contractors and other commercial concerns, and power utilities. We attempted to comprehensively cover the work being performed in the United States and opportunistically included work done abroad.

As in Activity 1, we covered both lifecycle management and testing techniques, focusing primarily on the testing techniques. Again, as in Activity 1, we examined V&V techniques applied to all phases of the development lifecycle, versus just to the testing phase. Finally, we examined both manual and automated techniques, providing a separate description of detailed automated tools.

As part of the survey effort, nine sites, where work was being performed in V&V of expert systems, were selected and visited.

## 1.2 Report Organization

The next section, 2.0, describes our technical approach to the Activity 2 survey, beginning with a description of our overall approach, then telephone surveys and site visits, followed by a description of our characterization and analysis of the techniques. Section 3.0 describes the reference lifecycle to be used for discussing and characterizing the techniques found in the survey. Section 4.0 presents a brief description of each of the techniques found. Section 5.0 describes separately the automated tools for expert system V&V that were found. A categorization and analysis of the techniques and tools follows in Section 6.0. This is followed by a summary in Section 7.0, which primarily contains recommendations for how the Activity 2 results can be applied in subsequent activities. Appendix A contains the Bibliography of materials collected over the course of the survey.

# 2. TECHNICAL APPROACH

## 2.1 Overall Approach

The detailed work plan for Activity 2 is shown in Figure 2.1-1. Three main "threads" can be seen in the Activity 2 work plan diagram. The first one in the left and lower middle involves telephone interviewing and reference document collection, and will be described below in Section 2.2. The second, in the upper middle involves site selection and survey, and will be described in Section 2.3. The last, on the right involves characterizing and analyzing the techniques, and will be discussed in Section 2.4.

## 2.2 Telephone Interviews and Data Collection

The first step in conducting telephone interviews was to develop a list of people to call. Names, addresses, and phone numbers of knowledgeable practitioners and researchers came from a number of sources throughout the activity period:

- Team members' existing professional contacts,
- Referrals from Dr. Bernard of people involved in nuclear power expert systems development and testing,
- Authors of papers on operational nuclear power expert systems (Artificial Intelligence and other Innovative Computer Applications in the Nuclear Industry, 1988, EPRI 1989b, 1988a, d, f, 1987d, Motoda, 1990, Moradian et al, Nelson,.1989, Osborne, 1986, Proceedings of the International Workshop on Artificial Intelligence for Industrial Applications, 1988),
- Attendees and speakers at the 1988, 1989, and 1990 AAAI, and IJCAI Workshops on V&V of Expert Systems,
- Members of standards organizations,
- Authors of papers collected from automated bibliographic search,
- Other references and acknowledgements in the papers we collected, and
- Referrals from other telephone contacts.

The list of names was organized into a Point of Contact (PoC) List, which was continuously updated and distributed to team members during the activity period. A list of the 97 names and organizations of the contacts is shown in Table 2.2-1.

Interview forms were prepared for collecting information from the telephone interviewees. After a few trial calls with the first draft of the form, it was shortened and modified to the one shown as Figure 2.2-1. The first page was followed by a totally blank page, on which answers to the discussion points on the bottom of the first page could be transcribed. The Activity 2 team members were trained in structured interviewing and the use of the form, and the

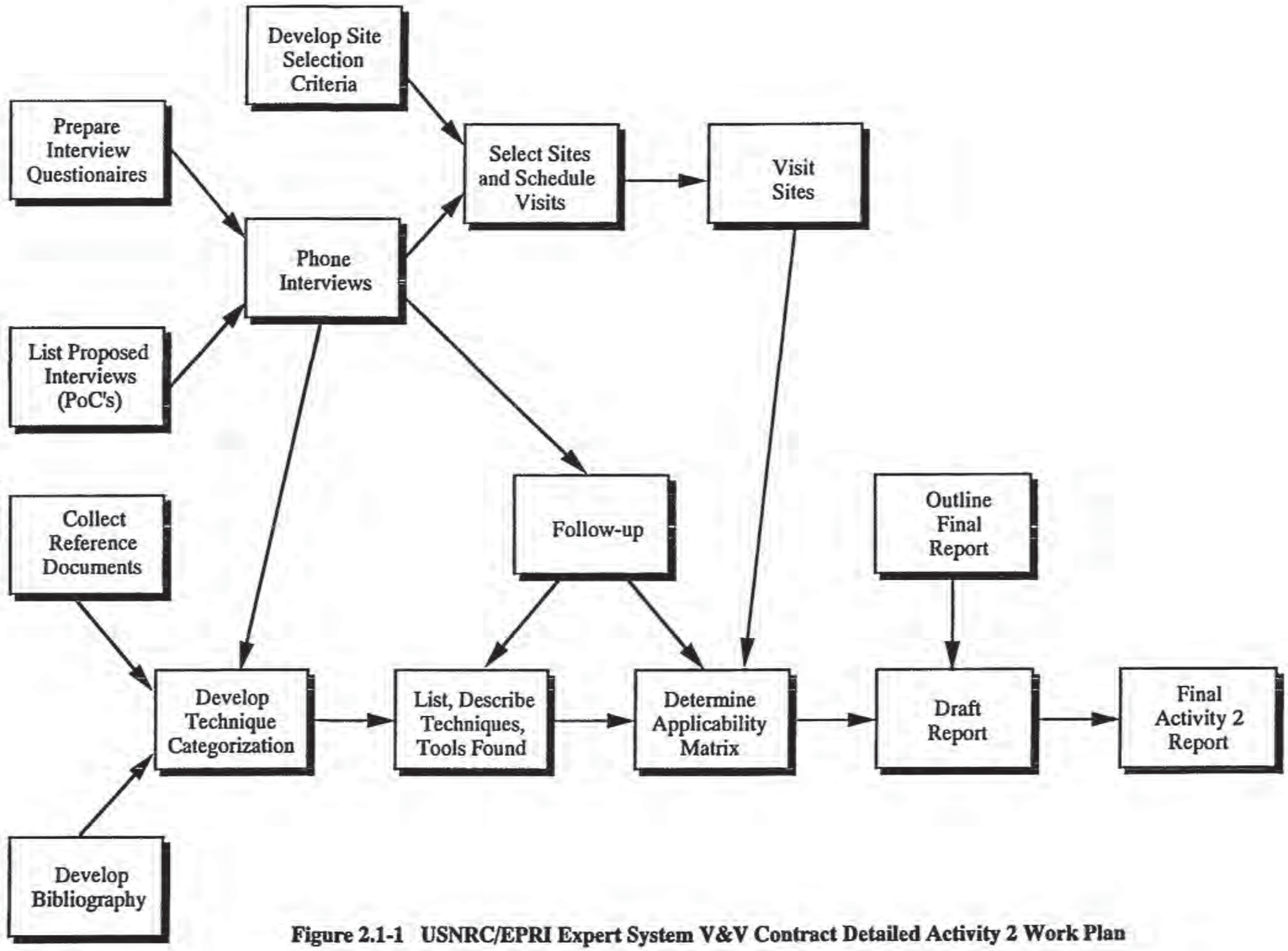

Develop Site Selection Criteria
Prepare Interview Questionaires
Select Sites and Schedule Visits
Visit Sites
Phone Interviews
List Proposed Interviews (PoC's)
Follow-up
Outline Final Report
Collect Reference Documents
Develop Technique Categorization
List, Describe Techniques, Tools Found
Determine Applicability Matrix
Draft Report
Final Activity 2 Report
Develop Bibliography

Figure 2.1-1 USNRC/EPRI Expert System V&V Contract Detailed Activity 2 Work Plan

Table 2.2-1 Persons Contacted for Telephone Interviews

| Point of Contact | Affiliation |
|---|---|
| Adelman, Leonard | George Mason University |
| Bahill, Terry | University of Arizona |
| Bartschat, Steffen | Ultrasystems |
| Bastl, D.W. | Forschungsgelande (Germany) |
| Bayse, Al | Federal Bureau of Investigation (FBI) |
| Bernard, John | Massachusetts Institute of Technology |
| Bloom, Howard | National Institute of Standards & Technology (NIST) |
| Bond, David | SAIC, COMSYSTEMS Division |
| Boose, John H. | Boeing Computer Services |
| Bray, Mike | EG&G Idaho Inc. |
| Buchanan, Bruce G. | University of Pittsburgh |
| Carbonara, Joe | Consolidated Edison - Indian Point 2 |
| Chee, Christine | BD Systems, Inc. |
| Cohen, Paul R. | University of Massachusetts |
| Combs, Jacqueline | Lockheed Missiles and Space Company, Inc. |
| Cragun, Brian J. | IBM |
| Cross, Steve | Defense Advanced Research Projects Agency (DARPA) |
| Culbert, Chris | NASA/Johnson Space Center |
| Duckworth, Jim | Worcester Polytechnic Institute |
| Edwards, Robert | Pennsylvania State University |
| Fausett, Mark | Rome Laboratory/COES |
| Franklin, Randolph | Renesselaer Polytechnic Institute |

Table 2.2-1 (Continued).

| Point of Contact | Affiliation |
|---|---|
| Freeman, Michael | NASA |
| Friedland, Peter | NASA-Ames Research Center |
| Fujii, Roger U. | Logicon, Inc. |
| Fussel, Louise | Rockwell Space Operations Company |
| Gabrielian, Armen | Thomson-SCF, Inc./Pacific Rim |
| Geissman, Jim | Abacus Programming Corporation |
| Gelperin, David | Software Quality Engineering |
| Garrett, Randy | Institute for Defense Analysis |
| Gilstrap, Lewey | Computer Science Corporation |
| Ginsberg, Allen | AT&T Bell Labs |
| Gowens, Jay | U.S. Army Institute for Research in Management Information |
| Griebenow, Ronald | NUS Corporation |
| Griesmer, James | Thomas Watson Research Center |
| Hajek, Brian K. | The Ohio State University |
| Hamilton, David | IBM |
| Harder, Bob | USAEPG; STEEP-ET-S |
| Harrison, Patrick | U.S. Naval Academy |
| Hayes-Roth, Frederick | Cimflex Teknowledge Inc. |
| Heindel, Troy | NASA/Johnson Space Center |
| Hirschberg, Morton | U.S. Army Ballistic Research |
| Holmes, Willard | U.S. Army Missile Command Research, Development & Engineering Center |
| Johnson, Sally C. | NASA |
| Kiguchi, Takashi | Hitachi, Ltd. |
| Kiss, Peter | Sentar, Inc. |
| Klein, Gary A. | Klein Associates |

Table 2.2-1 (Continued).

| Point of Contact | Affiliation |
|---|---|
| Laning, David | Intellicorp, Inc. |
| Lee, John C. | University of Michigan |
| Lehner, Paul | George Mason University |
| Lenat, Doug | Microelectronics and Computer Corp. |
| Leoni, Nicholas | Rochester Gas & Electric Company |
| Liebowitz, Jay | George Washington University |
| Linden, Theodore | Advanced Decision Systems, Inc. |
| Loganantharaj, R. | University of Southern Louisiana |
| Lupton, Lawrence | Chalk River Nuclear Laboratories |
| Lutsky, Patty | Digital Equipment Corporation (DEC) |
| Mahler, Ed | Dupont Corporation |
| Michalski, R.S. | George Mason University |
| Moradian, Ali | Westinghouse Electronic Corporation |
| Nazareth, Derek | University of Wisconsin-Milwaukee |
| Nelson, Robert | Georgia Power Company |
| O'Keefe, Robert | Rensselaer Polytechnic Institute |
| O'Leary, Daniel | University of Southern California |
| Odubiyi, Jide B. | Data Systems Technology |
| Osborne, Robert | Westinghouse Electric Corporation |
| Owens, Jerry | Navy Center for Applied Research in Artificial Intelligence |
| Owre, Fridtjov | Institutt fur Engergeteknikk |
| Parsaye, Kamran | Intelligence Ware |
| Pazzani, Michael | University of California, Irvine |

Table 2.2-1 (Continued).

| Point of Contact | Affiliation |
|---|---|
| Plant, Robert T. | University of Miami |
| Preece, Alun | Concordia University |
| Rossomando, Philip J. | General Electric Corporation |
| Rousset, Marie-Christine | L.R.I. - University' d'Orsay |
| Rushby, Dr. John | SRI International |
| St. Clair, Daniel | McDonnel Douglas Corporation |
| Sharma, Ravi S. | University of Waterloo |
| Sizemore, Nick L. | COMARCO, Inc. |
| Stewart, Tammy | USAEPG |
| Sudduth, Al | Duke Power Company |
| Surko, Pam | Science Applications International Corporation (SAIC) |
| Sztipanovits, Dr. | Vanderbilt University |
| Takahaski, Makoto | Tohoku University |
| Terano, Takoa | The University of Tsukuba, Tokoyo |
| Touchton, Robert | Pathfinder Advanced Computing, Inc. |
| Ulvila, Jacob | Decision Sciences Consortium, Inc. |
| Vesonder, Gregg | AT&T Bell Labs |
| Vignollet, Laurence | University of Savoie |
| Watson, David | Martin Marietta |
| Williams, Robert | U.S. Army Electronic Proving Ground |
| Williamson, Keith | Boeing Computer Services |
| Yen, John | Texas A&M University |
| Yokobayaski, Masao | Japan Atomic Energy Research |

**Figure 2.2-1 USNRC/EPRI V&V Interview Questionnaire**

I am X from SAIC, working under contract to the NRC and EPRI, on a survey task. We are interested in finding out what, if anything, you might be doing in the area of verification, validation or testing of expert systems or knowledge-based systems (ES/KBS V&V).

**SAIC Interviewer:** ____________ **DATE/TIME:**

**Person(s) Interviewed:** ______________

Contact list entry correct? Yes ____ No ____

FAX: E-MAIL: PHONE:

Title/Role:

Type of Work: Research ___ ES Development ___ Services ___

Study ___ Standards ____

Project/System Name:

Length of Work:

Number of People:

Customers? Yes ___ (see referrals) No ___

Funding source:

Can we visit? Yes ___ No ___

Project/System Description: (next page)

- Development/product plans
- Who should be interested (industry/ES type)
- Problem areas encountered
- Tool/technique needs identified
- Success?

Figure 2.2-1 (Continued).

Expert Systems Tested:

| **SYSTEM NAME** | **PLATFRM** | **OPSYS** | **SOFTWARE TYPE** | **SIZE ENV.** | **(#Rules,obj)** |
|---|---|---|---|---|---|
| | 1)PC<br>2)Apple<br>3)SUN 3)Unix<br>4)VAX 4)IBM<br>5)Other | 1)DOS<br>02)Apple<br>2)Embed<br>3)Stand<br>5)Other | 1)Real Time | 1)LISP, Prolog<br>2)Shell<br>3)Other Alone | 1)Small(<50)<br>2)Med(<500)<br>3)Lge (<3000)<br>4)Very Lge (>3000) |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

Testing Techniques:

| **TECH NAME** | **ES COMPON** | **TYPE ERRORS** | **AUTOMATED TOOLS?** | **EASE of SET-UP** | **P O W E R** |
|---|---|---|---|---|---|
| 1)EVA<br>2)random testing<br>3)other | 1)KB<br>2)InfEng<br>3)MMI<br>4)Shell<br>5)Other | Stat(1)<br>Dynam(2)<br>Anom(3)<br>Valid(4) | (Y or N) | 1(lo)<br>7(hi) | Ability to find errors<br>1(lo)<br>7(hi) |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

Automated Tools:

| **TOOL NAME** | **AVAIL** | **SOURCE** | **PLATFORM** | **OPERSYS** | **SOFTWARE ENV.** |
|---|---|---|---|---|---|
| | (Y or N) | | 1)PC<br>2)Apple<br>3)SUN<br>4)VAX<br>5)Other | 1)DOS<br>2)Apple<br>3)Unix<br>4)IBM<br>5)Other | 1)LISP,Prolog<br>2)Shell<br>3)Other |
| | | | | | |
| | | | | | |

**Figure 2.2-1 (Continued).**

Referrals (Colleagues/Customers):

| Name | Affiliations | Topic | Address | Phone |
|---|---|---|---|---|
| | | | | |
| | | | | |

Publication/Documentation References:

________________________________________

________________

Action Items:

| PERSON | REQUIRED ACTION | DATE REQUIRED |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

contacts were distributed among them. Weekly meetings were held during the heavy period of telephoning to collect referrals and other changes to the PoC list and to share information and interviewing hints.

A separate form was prepared for Dr. Bernard to fill out on operational or nearly operational expert systems within the nuclear industry. This form is reproduced as Figure 2.2-2. The aim was to draw upon his experience in writing his book, *Expert Systems Within the Nuclear Industry*, to gain an understanding of the state-of-the-art of expert system V&V within the nuclear industry. He sent along references with the forms, and if needed, follow up contacts were made.

In all, 138 PoCs were contacted which yielded the 97 doing current work mentioned above. However, many more people were called to generate these PoCs. This is because we would often have one name as an entry into the organization, and would chase through a number of referrals to obtain the best and most knowledgeable PoC in that organization. Also, some people were not doing work in the field themselves, but gave referrals to those who were. Then there were referrals by the referrals. We got to the point in the survey where PoCs were referring to each other and we had both the funder/sponsor and contractor/university PoCs of funding relationships on our list. This fact, and our limited resources, led us to limit the telephone survey, except for PoCs we knew were important, at some point so we could move on with the activity. A breakdown of the PoCs is shown in Figure 2.2-3.

Publications were collected from a number of sources. These included:

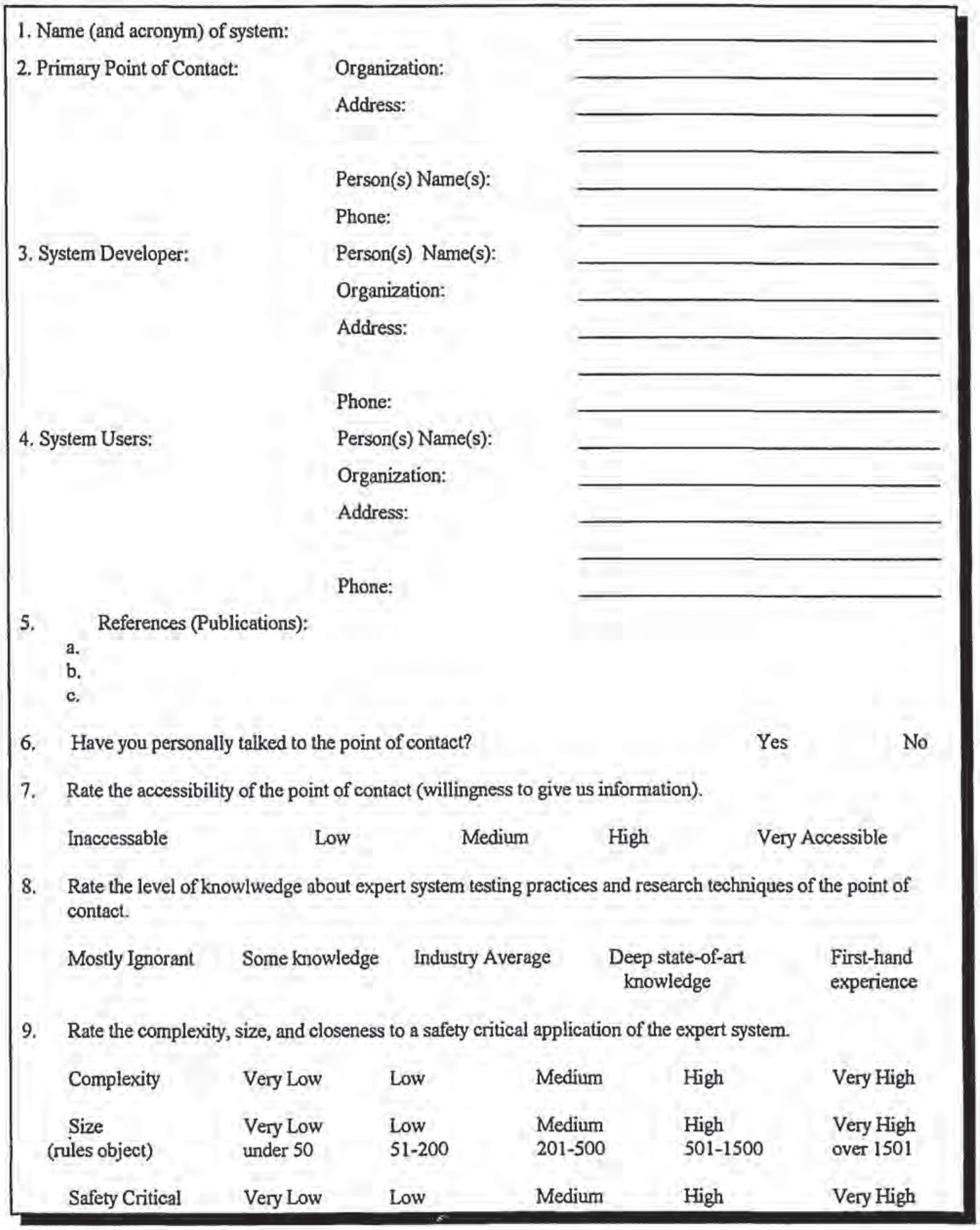
- Keyword-based search of the DIALOG and Defense Technical Information Center (DTIC) on-line computerized bibliographic services, followed by obtaining the most suitable publications,
- Conference and workshop proceedings and reports already on hand,
- Publications in Dr. Bernard's possession,
- Publications sent to us by PoCs after telephone interviews, and
- Publications collected at site visits.

The result was a very extensive library of materials on expert system V&V (well over 300 references). The bibliography for this library is included as Appendix B.

## 2.3 Site Selection and Visits

As a result of the telephone interview and data collection process which was described in Section 2.2, the project team determined that a number of sites offered the potential for obtaining significant additional information on expert system V&V techniques and tools. This preliminary list of sites was chosen after analyzing telephone interview data sheets and papers that were collected. Only those locations with robust ongoing expert system V&V activities that required an onsite, face-to-face meeting were included in this list. These sites, listed in Table 2.3-1, include private corporations, government facilities, universities, and members of the nuclear industry.

The locations listed in Table 2.3-1 exceeded the number which could be visited due to resource limitations. Therefore, Site Selection Criteria (SSC), delineated in Table 2.3-2, were developed. The SSC were used to assess each of the prospective sites. The SSC consist of 17 different parameters categorized as either technical, logistic, or balance



|  |  |  |
|---|---|---|
| 1. Name (and acronym) of system: |  |  |
| 2. Primary Point of Contact: | Organization: |  |
|  | Address: |  |
|  |  |  |
|  | Person(s) Name(s): |  |
|  | Phone: |  |
| 3. System Developer: | Person(s) Name(s): |  |
|  | Organization: |  |
|  | Address: |  |
|  |  |  |
|  | Phone: |  |
| 4. System Users: | Person(s) Name(s): |  |
|  | Organization: |  |
|  | Address: |  |
|  |  |  |
|  | Phone: |  |

5. References (Publications):
   a.
   b.
   c.

6. Have you personally talked to the point of contact? Yes No

7. Rate the accessibility of the point of contact (willingness to give us information).

| | | | | |
|---|---|---|---|---|
| Inaccessable | Low | Medium | High | Very Accessible |

8. Rate the level of knowlwedge about expert system testing practices and research techniques of the point of contact.

| | | | | |
|---|---|---|---|---|
| Mostly Ignorant | Some knowledge | Industry Average | Deep state-of-art knowledge | First-hand experience |

9. Rate the complexity, size, and closeness to a safety critical application of the expert system.

| | | | | | |
|---|---|---|---|---|---|
| Complexity | Very Low | Low | Medium | High | Very High |
| Size (rules object) | Very Low under 50 | Low 51-200 | Medium 201-500 | High 501-1500 | Very High over 1501 |
| Safety Critical | Very Low | Low | Medium | High | Very High |

**Figure 2.2-2 Survey Form**
**Nuclear Industry Expert Systems That Have Been Tested**

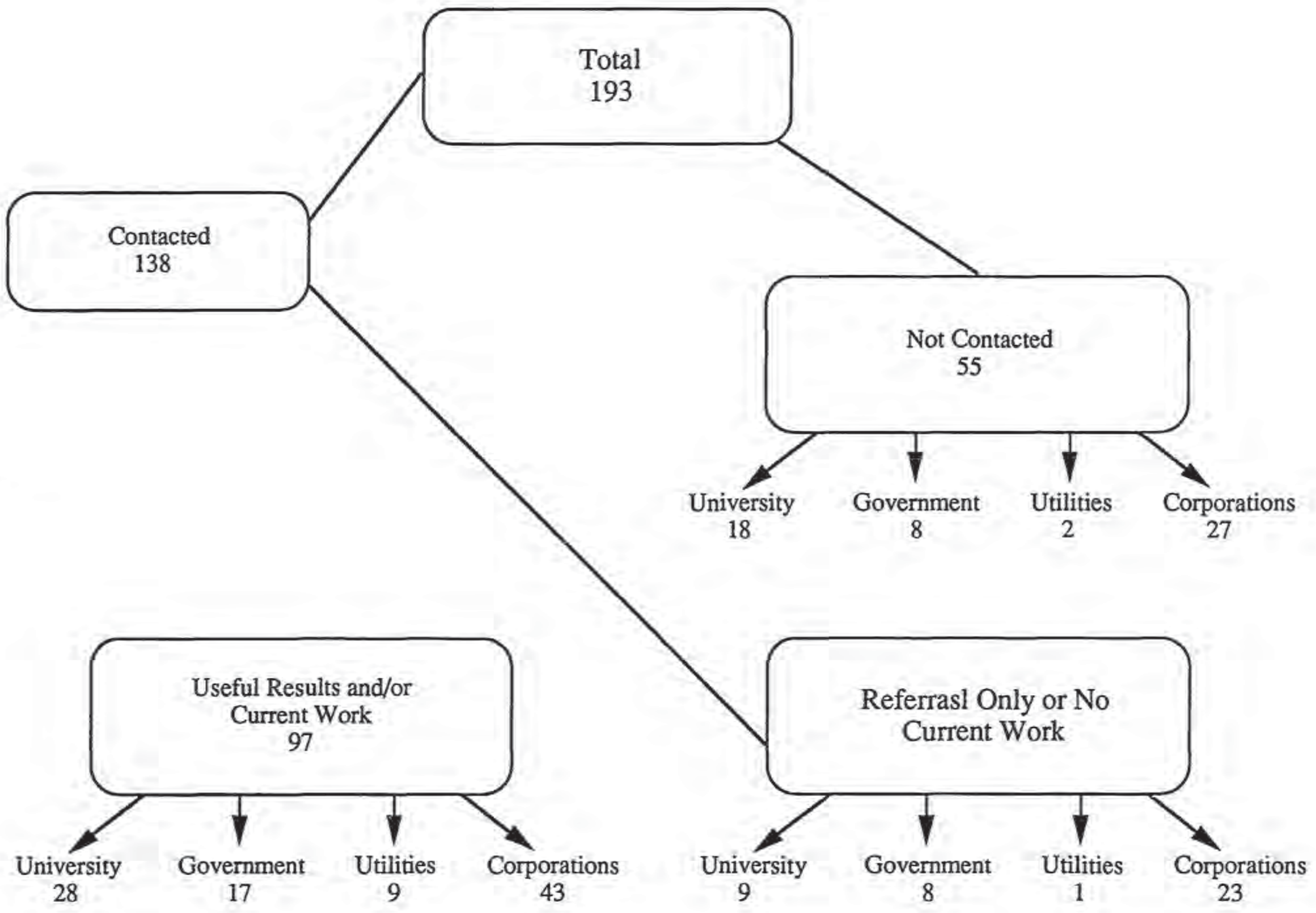


Figure .2.2-3Activity 2 Telephone Interviews

**Table 2.3-1. Preliminary Site List for Visit Evaluation**

| | |
|---|---|
| Near Washington, D.C. | Institute for Defense Analyses (IDA)<br>NIST (formerly NBS)<br>DARPA<br>Decision Sciences Consortium (DSC)<br>NASA Space Station Project<br>Aberdeen Proving Grounds<br>U.S. Naval Academy<br>B&W Nuclear Services |
| Remote Locations | Pathfinder Advanced Computing Technology<br>Lockheed Corporation<br>Darlington Nuclear Station<br>Digital Equipment Corporation (DEC)<br>Stanford Research Institute (SRI)<br>NASA Ames Research Center<br>Queens University<br>Worcester Polytechnic Institute (WPI)<br>University of Pittsburgh<br>Hartford Steam Boiler Insurance<br>Rensselaer Polytechnic Institute<br>Concordia University<br>University of Southern California<br>NASA Johnson Space Center<br>AT&T Bell Laboratories<br>Consolidated Edison Company<br>Indian Point 2 Nuclear Station<br>IBM |

**Table 2.3-2. Site selection criteria**
**NRC/EPRI Expert System V&V contract**

| Issues | Criteria | | Weight* |
|---|---|---|---|
| Balance | B1: | Representative sample of types of techniques | H |
| | B2: | Range over different types, sizes, and degrees of complexity of expert systems | H |
| | B3: | Sample both nuclear industry and other areas (medical, military, industrial, etc.) | M |
| | B4: | Sample both research and applied work | L |
| Technical | T1: | They have actually applied a V&V methodology to one or more expert systems | H |
| | T2: | They have a very well thought-out technical approach | H |
| | T3: | If Non-University, they have a requirement to V&V methodology | M |
| | T4: | They have developed or employed at least one V&V methodology | M |
| | T5: | Their approach to V&V is importantly different from the other sites | M |
| | T6: | They have adequately documented the methodology | M |
| | T7: | They have adequately documented the results of using the V&V methodology | M |
| Logistic | L1: | They are interested in cooperating with the survey effort | H |
| | L2: | Non-disclosure and/or any other legal agreements can be resolved | H |
| | L3: | Cannot obtain sufficient information over the telephone or via hard-copy reports | H |
| | L4: | People familiar with planning and performing the V&V are available | M |
| | L5: | Site-visit expenses are reasonable (e.g., in CONUS) | M |
| | L6: | Geographically close to high priority site already selected | L |

***Weights:** **L-Low** **M-Medium** **H-High**

issues. In addition, these criteria were assigned weights of high, medium, or low depending on their relative importance. It should be noted that sites in the Washington, D.C. metropolitan area were not rated because of their local access to the project team and their concomitant low visit expense.

Two project team members separately rated the sites in Table 2.3-1 using the SSC from Table 2.3-2. Although the two independent rankings did not agree on all the sites, they did select the same top five locations outside Washington, D.C. These five sites were: San Francisco area (Lockheed, NASA AMES Research Center, and Stanford Research Institute), Northeastern U.S. (DEC, Worcester Polytechnic Institute, Hartford Steam Boiler, Consolidated Edison Company - Indian Point 2, and Bell Laboratories), Jacksonville, Florida (Pathfinder Advanced Computing Technology), Montreal-Toronto (Darlington, Queens University, Concordia University), and Houston (NASA Johnson Space Center and IBM). These initial selected sites are presented in Table 2.3-3. Wherever possible, lower priority sites at the same location were included in actual trips to maximize the benefits of the visits.

After review of the SSC and recommended sites, the Washington, D.C., San Francisco, Houston, and Northeastern U.S. trips were selected. The USNRC provided a considerable amount of documentation that had been obtained from earlier visits to the Darlington nuclear plant in Canada. After review of this documentation, the project team agreed that this visit would not result in a significant acquisition of additional knowledge and the Montreal-Toronto site was eliminated from the list. EPRI provided the REALM Verification and Validation Plan which was developed by Pathfinder in Jacksonville. The availability of this document has reduced the benefits of a visit to Jacksonville and therefore this site visit was also eliminated. It should also be noted that some of the Washington, D.C. and Northeastern U.S. area sites were not visited due to a lack of interest, cooperation or proprietary/classified issues by the host organization. The actual sites which were visited (during May through August of 1991) are delineated in Table 2.3-4. In all cases, the personnel at these sites were open and cooperative in answering questions, making presentations, and providing documentation. A set of detailed discussions of the knowledge that was obtained during these site visits is presented in Appendix A of this report.

## 2.4 Technique Characterization and Analysis

In this report, we distinguish between V&V "techniques" and V&V "tools". A technique for expert system V&V is a method or procedure for performing some aspect of V&V on components or all of an expert system. There may be many organizations using or studying the technique and applying it in slightly different ways. An expert system V&V tool is an automated software program -- usually proprietary -- which embodies one or more expert system V&V techniques within it.

As we began collecting reference materials, we started a list of the techniques and tools that we had found to date. Worksheets were prepared for distilling the information about a particular technique/tool scattered across multiple publications into one concise form. One of these worksheets is shown in Figure 2.4-1. These formed the basis for writing Section 4.0 of this report by providing basic information and pointers to the appropriate publications. They also prompted a second round of telephone calls to the tool developers, in particular to obtain detailed information about the computing environment, availability, etc.

The techniques and tools were characterized along several dimensions to be able to begin some comparison of them. However, a thorough quality comparison and selection of the best ones to include in the final suggested methodology for V&V of expert systems for nuclear power applications will not begin until Activity 4 of the contract. In Sections 4.0 and 5.0, the expert system V&V tools and techniques are grouped into the appropriate classes and

**Table 2.3-3 Recommended site visits based on site selection criteria ranking**

| | |
|---|---|
| Washington D.C. Area | • Institute of Defense Analysis (IDA)<br>• ARPA<br>• Decision Sciences Consortium (DSC)<br>• U.S. Naval Academy, Annapolis, Maryland<br>• B&W Nuclear Services, *Lynchburg, Virginia* |
| San Francisco Area | • Lockheed Corporation<br>• Stanford Research Institute |
| Northeastern United States | • Digital Equipment Corporation (DEC), *Marlboro, Massachusetts*<br>• Worcester Polytechnic Institute (WPI)<br>• Hartford Steam Boiler Inspection and Insurance Company, *Hartford, Connecticut*<br>• Consolidated Edison Company, *Indian Point 2 Nuclear Station*<br>• AT&T Bell Laboratories, Warren, *New Jersey* |
| Jacksonville, Florida | • Pathfinder Advanced Computing |
| Montreal and Toronto, Canada | • Darlington Nuclear Station<br>• Queens University<br>• Concordia University<br>• |
| Houston, Texas | • NASA Johnson Space Center<br>• International Business Machines (IBM) |

**Table 2.3-4 Expert system verification and validation survey activity actual site visits**

| | |
|---|---|
| Washington, D.C. Area | • Institute for Defense Analyses (IDA)<br>• Decision Sciences Consortium (DSC)<br>• U.S. Naval Academy |
| San Francisco Bay Area | • Lockheed Corporation<br>• Stanford Research Institute (SRI) |
| Northeastern U.S. | • Digitial Equipment Corporation (DEC), *Marlboro, Massachusetts*<br>• Worcester Polytechnic Institute (WPI)<br>• AT&T Bell Laboratories, *Warren, New Jersey* |
| Houston, Texas | • NASA Johnson Space Center<br>• International Business Machines (IBM) |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| Acronym: |  |  | Automated Tool? | Yes | No |

Name:

Description:

Point(s) of Contact

Affiliation(s):

Bibliography References:

Commercial or Research?

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| Run Time Version Available? | Yes | No | Source Code Available? | Yes | No |

Cost?

Expert System Component(s):

Form(s) of KB:

KB's Shell/Language(s):

**Environment:**

Software Shell/Window System/DBMS/Compiler(s):

Programming Language:

Operating System(s):

Hardware Platform(s):

|  |  |  |  |
|---|---|---|---|
| General Error Types Handled: | Static | or | Dynamic |
|  | Anomalies | or | Invalidities |

Specific Error Types Found

Who Evaluates Tool/Technique Outcome: Expert Layman Programmer Other (explain)

Size of ES It Can Handle:

Theoretical Efficiency for Finding Errors:

Set-Up Preparation Required:

User Community:

Comments:

subclasses of the V&V technique classification hierarchy developed in Activity 1. The dimensions along which the techniques and tools are characterized in Section 6.0 include the following:

- The components of expert systems to which the technique/tool may be applied.
- Forms of knowledge base to which the technique/tool can be applied (rules, objects, networks, etc.).
- Phases of the software development lifecycle within which the technique/tool can be applied.
- Types of errors handled: anomalies (structural or syntax flaws) versus invalidities (errors in behavior/results regardless of structure).

# 3. REFERENCE LIFECYCLE FOR THIS ACTIVITY

A number of standards and reference documents have included waterfall lifecycle diagrams with insertion points for V&V activities. For the purposes of this report, we will use the NSAC-39 Software Development Lifecycle shown in Figure 3.0-1. This lifecycle diagram is widely known and accepted in the nuclear utility industry. It will be used to characterize the applicability of various expert system V&V techniques and tools to appropriate stages of the development lifecycle. Thus, if an expert system V&V technique or tool is applicable to design activities, but not to actual implementation activities, it will be related to the Design phase of the NSAC-39 Lifecycle. The lifecycle phases used to categorize expert system V&V tools and techniques are the following:

- Software Specifications: document and validate the requirements for the software system, including all performance specifications;
- Design: develop and document the top-level or architectural design, followed by development of the detailed design;
- Implement: code, debug, and test the actual software;
- Hardware-Software Integration: integrate the software with the delivery hardware platform and test; and
- Computer System Validation: validate the system as a whole against the requirements.

The bulk of expert systems development is usually performed under an iterative prototyping lifecycle versus a non-iterative waterfall-type lifecycle such as NSAC-39. Numerous references can be found quoting experience in using an iterative approach to implement expert systems (see EPRI, 1988e, 1989b; Artificial Intelligence and Other Innovative Computer Applications in the Nuclear Industry, 1988) and in defining specific lifecycles based on the iterative approach, with appropriate insertion points for V&V activities throughout (e.g., Benbasat & Dhaliwal, 1989; Culbert, 1987b, 1988b; EPRI, 1988b; Miller, 1990b; May, 1991, Richardson & Wong, 1988, Yen et al, 1990a,b). Computer Sciences Corporation (CSC) has extensively documented an Expert System Development Methodology (ESDM) (see CSC, 1989a-d; Gilstrap, 1990b, 1991; Sary, 1990) for the National Aeronautics and Space Administration (NASA), which has been applied successfully to at least three expert systems (Gilstrap, 1990a,c). An International Business Machines (IBM) survey (IBM, 1990) found that 41 of 62 respondents used at least a two-loop iterative lifecycle as their development process. Only five used a traditional waterfall lifecycle. In addition, Japanese expert system developers also have found the iterative model works well (Terano et al, 1990, 1991).

It is fully intended that the expert systems V&V methodology that will be developed under Activity 4 will be based on some form of an iterative lifecycle. However, the two types of lifecycles can be related in that a large subset of the phases in a waterfall lifecycle will be repeated successively in an iterative lifecycle. Thus, if an expert system V&V technique is related to a particular phase of NSAC-39, such as the design phase, it will also relate to the repeated design phases of an iterative lifecycle model. Using NSAC-39 for now will not preclude us from mapping techniques to the appropriate phases of a spiral lifecycle, and in fact will help us.

Our original aim in this report was to focus on the technical aspects of the techniques and tools for expert systems V&V that were found in the survey and de-emphasize software management issues. However, we found that there is universal agreement extending from the research labs to the operational expert system maintenance shops that V&V must occur throughout the development and maintenance lifecycle from the beginning on. There was wide agreement that waiting until the testing phase to begin V&V activities was a recipe for failure and/or cost overruns. So, a message that will be reiterated throughout this report is that V&V must be an integral part of the whole lifecycle. This is clearly a management topic.

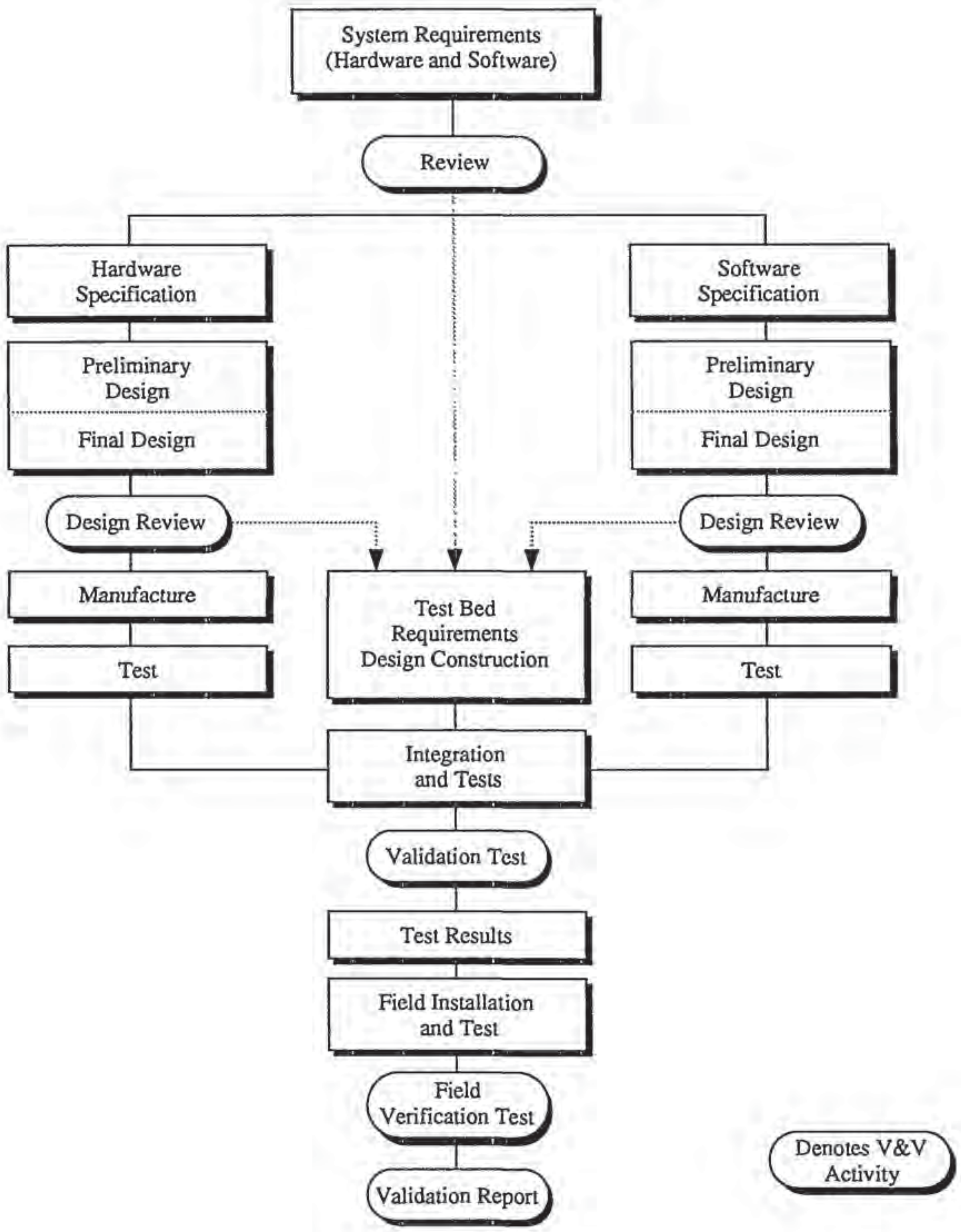


**Figure 3.0-1 Relationship of V&V Activities to Generic Project Activities, From NSAC-39 (1981)**

# 4. EXPERT SYSTEM V&V TECHNIQUE DESCRIPTIONS

## 4.1 Types of Techniques

In this section, the expert system V&V techniques found in the survey are described and characterized. Applicable bibliographic references will be given for each technique, so the reader may refer to the source for more information. The techniques have been categorized into the class/subclass hierarchy developed in Volume 2, shown in Figure 4.1-1.[1] The numbers in parentheses are the number of techniques within each subclass. Minor changes and additions were made to the hierarchy to accommodate new techniques, specific to expert systems, that were discovered during the course of the survey.

The changes in the class hierarchy evident in Figure 4.1-1 are the following:

- One new technique under Algorithm Analysis of Static Testing, called Confidence Weights Sensitivity Analysis,

- Two new techniques under Defect Analysis of Static Testing: Semantic Consistency Testing and Knowledge Acquisition/Refinement Aids, and

- One new technique under Performance Testing, called Using Generated Explanations.

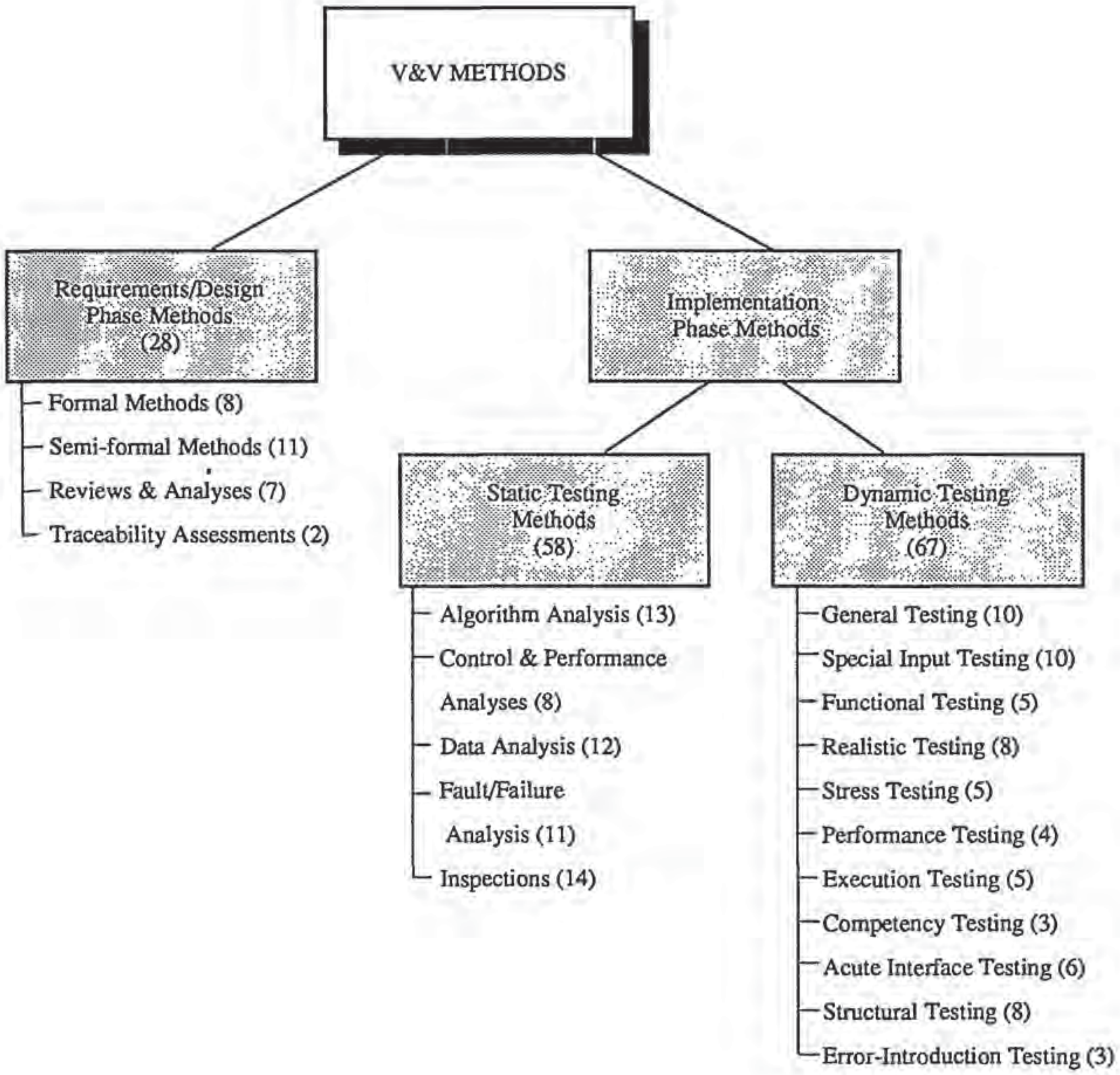
Further detailed changes will be discussed in subsections 4.2 through 4.4 below. Section 4.0 will conclude with a discussion of the state-of-the-art in expert system V&V in Section 4.5.

## 4.2 Requirements and Design Testing

The techniques categorized under Requirements and Design Testing are suitable for application during the Software Specification and Design phases of the NSAC-39 Software Development Lifecycle. Table 4.2-1 lists the subclasses of techniques for V&V of requirements and design. Under the "Applied to Expert Systems" column are references from people or organizations who have done research in or applied the techniques to expert systems. If the requirements and design are documented, then the work activities performed in these two phases of development are essentially similar for both expert systems and conventional software systems. As can be seen, most of the techniques applied have been somewhat informal (reviews and tracing) vs. formal (requirements or design language processing).

We have found that neither requirements/specification nor design documents exist for many expert systems, and when they do, they are usually written after the fact or late in the development process. For example, there is still no requirements document for the DEC XCON and XSEL expert systems (DEC interview, 1991). If no requirements document exists, it is impossible to do any real requirements V&V. Many sources (see table) agree on the need for a requirements document at some point in the development lifecycle in order to have requirements reviews and perform requirements tracing.

[1] The numbering and names of the V&V classes and subclasses differs slightly from the full description that appears in Volume 2, A Survey of Conventional V&V Methods. This is because Volume 2 was revised to take into account the results of this and all the following tasks of this project.



* Number in parentheses indicate the number of individual number of V&V Methods of that type.

**Figure 4.1-1. Classes of Conventional V&V Methods Organized by Life-Cycle Phase**

Table 4.2-1 Requirements and design V&V techniques applied to expert systems

| Testing Classes/Subclasses | Description | Applied To Expert Systems |
|---|---|---|
| 1.1 Requirements Language Analysis | Expression of requirements specifications in a special requirements language and analysis of results for completeness, consistency, feasibility, and testability. | Plant & Gold, '90; Rushby & Whitehurst, '89; Rushby, et al., '91 a,b |
| 1.2 Requirements Language Processing | Processing or execution of the requirements language to permit examination of implications of the specification. | None |
| 1.3 Mathematical Verification of Requirements | Translation of requirements into mathematical form for proving various properties (security, ultra-hi reliability). | Rushby, et al., '91 a,b; Rushby & Whitehurst, '89 (formal specs: constraint satisfaction, model inversion, approximate semantics) |
| 1.4 Formal Requirements Review | Review by special personnel of the adequacy of the requirements specification according to detailed pre-established set of criteria and procedures. | Batarekh et al., '91; Benbasat, '89; May '91; Path- finder, '91 (V&V Plan); ICF, '91; EPRI, '87c; <IDA site visit>, '91 |
| 1.5 Requirements Tracing/ Traceability Analysis | Identification of individual requirement aspects and tracing of these to design aspects, and from the design to aspects of the implemented program. | Rossomando, '89 (RT/T); Green & Keyes, '87; Johnson, '88; <Chee interview>, '91; Path-finder '91 (V&V Plan); EPRI, '87c, '88b; Bond, '88; Miller, '90a |
| 1.6 Design Compliance Analysis | Verification process that design is compliant with – realizes – all aspects of requirements. | None |
| 1.7 Design Simulation | Representation of the design in some executable language (e.g., SLAM-2) and execution of this representation. | None |
| 1.8 Program Description Language (PDL) Analysis and Processing | Representation of detailed program design in a special description language for analysis and execution tests. | None |

| Testing Classes/Subclasses | Description | Applied To Expert Systems |
|---|---|---|
| 1.9 Formal Design Review | Review by special personnel of the adequacy of the design according to detailed pre-established set of criteria and procedures. | Benbasat, '89; Preece, '90; ICF, '91; <Osborne interview>, '91, Pathfinder, '91 (V&V Plan); May, '91; EPRI, '87c, '88b |
| 1.10 Critical Timing/Flow Analysis | Analysis of the process and control timing aspects of the design to determine if requirements related to the aspects are satisfied. | None |
| **NEW:** None | | |

Only two research organizations, Stanford Research Institute (SRI) and the University of Miami, have investigated formal methods for requirements language processing and mathematical verification of expert systems. SRI's work by far is the more sophisticated, given their background in computer security, flight systems, and other ultra-high reliability systems. Dr. Rushby and his associates at SRI have attempted to apply many forms of formal proofs of correctness to rule bases, including constraint satisfaction, model inversion, and approximate semantics, and have discovered problems in applying each of the techniques. The main concern is that the effect of the inference engine's conflict resolution strategy on the order of rule firing cannot be explicitly represented in the rule base. Therefore, constructing a proof on the rules alone tells you nothing about the actual execution of the software.

Since requirements and design documents for expert systems are rarely written, the state-of-the-art in requirements and design V&V for expert systems is to do nothing. When documents are written, expert system developers may conduct requirements and design reviews and/or perform requirements tracing. Some research, so far with negative results, is being done in formal proof-of-correctness methods applied to expert systems. However, no other work is ongoing in applying formal methods for V&V of expert system requirements or designs.

## 4.3 Static Testing

The techniques categorized under Static Testing are suitable for application during the Implement phase of the NSAC-39 Software Development Lifecycle. Table 4.3-1 lists the subclasses and techniques for Static Testing, defines them, and lists the appropriate references for those that have been applied to expert systems. Each of the subclasses of Static Testing which have been used for expert systems will be discussed in this section in turn.

Under **Algorithm Analysis**, though Metric Analyses have been proposed by Miller, only Program Proving has actually been attempted. Rushby and his associates at SRI have done the bulk of the work, with the somewhat negative results as discussed in Section 4.2 above. The papers by Castore and Wood & Frankowski propose formal methods based on modal logic and Dijkstra's invariance theorem, respectively, but neither method has been applied. A new technique, Confidence Weights Sensitivity Analysis, has been applied to expert systems to generate measures of the sensitivity, accuracy, or bias in the confidence factors, weights or probabilities placed on rule conclusions. These methods are only appropriate for rule bases utilizing some form of uncertainty management and depend somewhat on the type of uncertainty management used. However, they do not find errors, but instead provide measures that can be indicators of potential errors.

Under **Control and Performance Analysis**, the techniques listed are very tailored to conventional software, and we found no evidence of any of them being applied to expert systems. Nor was there any evidence of new Control Analysis techniques being developed specifically for expert systems.

Under **Data Analysis**, there has been some work in dependency analysis applied to expert systems, specifically in specifying the input variables on which each output variable is dependent (Franklin et al, 1989, 1988a, b, Riber, et al, 1991). This technique does not directly identify errors, but can be used as a debugging tool. Rossomando (1989), of the GE Astro Space Division, has proposed an environment for V&V of expert systems consisting of a number of separate tools, one of which would be a cross reference list generator. Other than these two techniques, none of the Data Analysis techniques have been applied to expert systems, nor have any new ones been developed for expert systems.

The **Defect Analysis** subclass is where the bulk of work in static testing of expert systems has been done. A number of groups have developed automated tools for performing Syntactic Checking of rule bases (see Table 4.3-1). The techniques have also been applied manually to small rule bases (Groundwater, 1990, Rushby & Crow, 1990). The

Table 4.3-1 Static testing V&V techniques applied to expert systems

| Classes/Subclasses | Description | Applied to Expert Systems |
|---|---|---|
| **2.1 ALGORITHM ANALYSIS** | | |
| .1 Analytic Modeling | Representing the program logic and processing in some kind of model and analyzing it for sufficiency. | None |
| .2 Cause-effect Analysis | Identifying the triggers of processes, their effect during activation in states of variables, and the final terminating conditions. | None |
| .3 Symbolic Execution | Representing the data computations as algebraic equations and solving these algebraically through the whole program. | None |
| .4 Trace-assertion Method | An algorithm specification (or representation) method involving description of the sequence of invocations of system modules, including I/O values, in terms of v. assertions axiomatic about the traces, in a "black-box" fashion. | None |
| .5 Functional Abstraction | Representing design or programs as a series of mathematical functions based on a small set of primitive programming functions (e.g., iteration, sequence, select, etc.) then, recursively, dividing parent functional specifications into sub-specifications and mathematically verify equivalence. | None |
| .6 L-D Relation Methods | An alternative to functional specifications for non-deterministic programs using relations and the competence set of states in which termination is guaranteed. | None |
| .7 A-7 Table Formats | Using table formats to present the information in trace-assertion specifications. | None |

Table 4.3-1 (continued)

| Classes/Subclasses | Description | Applied to Expert Systems |
|---|---|---|
| **2.1 ALGORITHM ANALYSIS (Cont.)** | | |
| .8 Program Proving | For each code segment, developing formal specifications of functional intent and specific I/O characteristics; for actual or symbolic input then proving via some proof procedure that the segment performed as intended. | Wood & Frankowski, '90 & '89 (Dijstras's invariance theoreom); Castore, '87; Rushby, '88; Rushby, et al, '90; Rushby & Whgitehurst, '89; Rushby, et al, '91a,b; Waldinger & Stickel, '91 (approximate semantics) |
| .9 Metric Analyses | Computation of various complexity metrics for the program. | Miller, '90 (suggested) |
| **NEW:** | | |
| .10 Confidence Weights Sensitivity Analyses | Using statistical analyses to measure the sensitivity, accuracy or bias in the confidence factors/weights placed on rules' conclusions. | Lehner, '88 (signal detect to measure nodes in inference network-accuracy, bias); IEE, '87 (confidence weights); Gaschnig, '83; O'Neil & Glowinski, '90; O'Leary, '88; O'Leary & Kanderin, '88 (weights); O'Leary, '89; O'Keefe & O'Leary, '91 |

Table 4.3-1 (continued)

| Classes/Subclasses | Description | Applied to Expert Systems |
|---|---|---|
| **2.2 CONTROL ANALYSIS** | | |
| .1 Control Flow Analysis | Analyzing the program into a series of, e.g., decision and process actions and representing all of the possible alternative process sequences in the program. Often used to assess whether program is well-structured and has no unreachable code. | None |
| .2 State Transition Diagram Analysis | Determining the condition (variable states, etc.) that trigger the onset and cessation of program processes. | None |
| .3 Program Control Analysis | Related to 3.2.1 and 3.2.2 but concerned more with the sequential aspects leading to a particular execution path in a program. | None |
| .4 Operational Concept Analysis | Analysis of the manner in which the software system interacts with and is dependent upon states of the environment and external decisions, especially of human operators. | None |
| .5 Calling Structure Analysis | Module by module analysis of hierarchically structured programs involving procedure calls to determine what sequence of higher level module calls led to the invocation of a particular module, and what modules in turn are called by it. | None |
| .6 Process Trigger/Timing Analysis | Analysis of the conditions which activate a process (similar to 3.2.2) with special concern for the timing of activation relative to other processes. | None |
| .7 Worst-case Timing Analysis | Analysis to determine the longest execution-time path through a program often comparing this to a reference safety limit. | None |
| .8 Concurrent Process Analysis | Analysis of the overlap or concurrency of different processes in multi-tasking, parallel processing, or concurrent processing programs. | None |

NEW: None

Table 4.3-1 (continued)

| Classes/Subclasses | Description | Applied to Expert Systems |
|---|---|---|
| **2.3 DATA ANALYSIS** | | |
| .1 Data Flow Analysis | Analysis of the data inputs, outputs, and controls to all program processes. | None |
| .2 Dependency Analysis | Determining what variables depend on what other variables, similar to "influence diagrams." | Franklin, et al, '89, '88a,b (ESPE); Ribar, et al, '91 |
| .3 Look-up Table Generator | Generating the location within various modules of data and control variables. | None |
| .4 Data Dictionary Generator | Generating a defined/used table of locations of all program variables. | None |
| .5 Cross-reference List Generator | Generating the location of all data variables, in form of cross-reference table of modules. | Rossomando, '89 (CR/T) |
| .6 Aliasing Analysis | Analysis of the aliases of variables used in the main procedure and passed as parameters/arguments to its called procedures (and theirs). | None |
| .7 Concurrency Analysis | Analyzing programs for existing or potential concurrent data-paths and processing. | None |
| .8 DB Analyzer | Checking the structure, normal form, declarations, and values of database. | None |
| .9 DB Interface Analyzer | Checking the interface(s) of a program with primary data input for error-detection and handling, consistency-checking, etc. | None |
| .10 Model Evaluation | Evaluating the adequacy and features of the data schema or meta-schema used to organize the DB and data structures. | None |
| NEW: None | | |

Table 4.3-1 (continued)

| Classes/Subclasses | Description | Applied to Expert Systems |
|---|---|---|
| **2.4 DEFECT ANALYSIS (NEW Subclasses)** | | |
| .1 Syntactic Checking | Converting a rule base into a structure (graph, table, matrix, etc.) to discover structural flaws that could be errors. | Becker, et al, '89a,b, '90, '91, <WPI site visit>, '91 (evidence flow graphs); Bahill, '87, '91, Kang & Bahill, '90 (Validator); Nguyen, et al, '85, '87 (Check, ARC); Preece, '90 a,b,c, Preece & Shingal, '91 (Cover); Rousset, '88 a,b (COVADIS); Culbert & Savely, '88, '89 (CSRV); Stachowitz, et al, '87a,b, '88, Chang, et al, '90a,b, Burris, McGuire '90, McGuire & Stiles, '90, <Lockheed site visit>, '91 ((D)EVA); Cragun & Steudel, '87 (ESC); Franklin, et al, '88, '89 (ESPE); Lopez, '90 (INDE); Suwa, et al, '82 (RCP for ONCOCIN); Ginsberg, '88a,b, '90 (KB-Reducer); Krishnamurthy et al, '87 (TVE); Groundwater, '89, Rushby & Crow '90 (hand-checks); Vick & Lindenmayer, '88 (unnamed tools); Wensley, '90, Laurent & Ayel, '89 (SACCO) Surveys, tutorials: Nazareth, '89; Lopez, '90; Rushby, '88; EPRI, '88b; Miller, '89d; Landauer, '89, '90 |

Table 4.3-1 (continued)

| Classes/Subclasses | Description | Applied to Expert Systems |
|---|---|---|
| 2.4 **DEFECT ANALYSIS (NEW Subclasses) (Cont.)** | | |
| .2 Semantic Checking | Use of meta-rules or constraints to check the internal semantic consistency of a knowledge base. | Becker, et al, '89a,b, '90, '91, <WPI site visit>, '91 (evidence flow graphs); Stachowitz, et al, '87a,b, '88; Chang, et al, '90a,b, Burris, McGuire, '90, McGuire & Stiles, '90, <Lockheed site visit>, '91 ((D)EVA); Wensley, '90, Laurent & Ayel, '89 (SACCO) |
| .3 Knowledge Acquisition/ Refinement Aid | Use of an automated tool during knowledge acquisition or refinement (maintenance) to prevent some types of errors from being created in the first place or to assure complete coverage of possible input values, possibly through machine learning techniques. | Desimone, '90; Ginsberg, et al, '85, '86 (SEEK); Klinker, et al, '87 (KNACK); <Lenat interview>, '91; Laird, '88 (Soar); Park & Wilkins, '90, Wilkins, '88 (ODYSSEUS); Wilkins & Buchanan (Antidote Algorithm); Pazzani & Brunk, '90 (KR-FOCL); Suwa, et al, '82 (RCP for ONCOCIN); Davis, '81 (Tereisias); Eschelman & McDermott, '86 (MOLE); Kahn, et al, '85 (MORE); Smith, et al, '85 (LAS); Marcus, '88 (SALT); Baum, et al, '89, Boose, '84, (AQUINAS/ERASAMUS) |

**Table 4.3-1 (continued)**

| Classes/Subclasses | Description | Applied to Expert Systems |
|---|---|---|
| **2.5 FAULT, FAILURE ANALYSIS** | | |
| .1 Failure Mode, Effects, Causality Analysis (FMECA) | Identification of the failure modes of each system component and analyzing the consequences of each failure type. Information gathered includes failure description, cause(s), defect(s), detection means, resultant safety consequences or other hazards, and recovery methods and conditions. | None |
| .2 Fault-tree Analysis (Event-Tree Analysis) | Beginning with a system hazard or failure, the analysis identifies or hypothesizes immediate and proximal causes, and describes the combination of environment and events that preceded, usual in the form of a directed graph or "and-or tree". Often accompanied by an "event tree analysis" showing relevant event-propagation information. | Morell, '89; Miller, '89b,d (suggested) |
| .3 Heuristic Testing | Emphasizes the importance of a prior fault prioritization and analysis to determine fault-enabling conditions. | Miller, '90 (suggested) |
| .4 Criticality Analysis | Identification of the critical points of failure in a program and the development of test cases to verify their accuracy and robustness. | None |
| .5 Failure Modeling | Analysis of failure data to develop metrics and causal hypotheses about system components' failure rates, fault-sources, and future behavior. | None |
| .6 Hazards/Safety Analysis | Analysis of the system (and requirements) to identify potential hazards or safety events, or to consider such events determined by dynamic testing. Full analysis involves 3.4.2 in addition. | None |
| .7 Common-Cause Failure Analysis | Identifying failures that affect apparently independent modules. | None |
| **NEW:** None | | |

Table 4.3-1 (continued)

| Classes/Subclasses | Description | Applied to Expert Systems |
|---|---|---|
| **2.6 INSPECTIONS** | | |
| .1 Informed Panel Inspection | Convocating a qualified group to review the quality of a program. | Miller, '89b,d; Culbert, '87a; IBM survey, '90; <Mahler interview>, '91; Wensley, '90; Ribar, et al, '91; <Moradian interview>, '91; Moradian et al, '90 |
| .2 Structured Walkthroughs | An analysis of a program module, usually by programmer, with audience of programming team members. | <DEC site visit>, '91; Rushby, '88; Miller, '89b,d; EPRI,'88b |
| .3 Clean-room Techniques | A number of analytic (and procedural) practices for extremely high reliability code production. | Miller, '89b,d (suggested) |
| .4 Peer Code-checking | Fellow programmers checking each other's code. | None |
| .5 Desk Checking | Inspecting program source code without benefit of automated tools. | IBM survey, '90; Oxman, '91 |
| .6 Formal Customer Review | A formal evaluation by the customer representatives of the adequacy of a program. | None |
| .7 Data Interface Inspection | Inspection of all data interfaces of a program for adherence to specification, error-handling, consistency-checking, and other features. | None |
| .8 User Interface Inspection | INSPECTION of all aspects of the interface to the user and operator for adequacy and other criteria. | None |
| .9 Standards Audit | Evaluation of the program to determine its compliance with set of governing standards. | Benbasat, '89; Goodwin & Robertson, '87; Soloway, '87 (RIME); Pathfinder, '91 (source code comments) |
| **NEW:** None | | |

automated tools are described in detail in Section 5.0. The tools usually transform the rule base into a graph or matrix form so structural flaws may be found. These forms, and example tools using them, include the following:

- Rule connection/inference graph ((D)EVA, ESPE),
- Evidence flow graph (WPI work),
- Petri net (INDE),
- Repertory grid (Aquinas),
- Rule/value dependency chart (RCP for ONCOCIN, ARC, COVER),
- Semantic network (KNACK, MOLE), and
- Decision table (ESC).

Some tools may also construct ancillary structures to the rule graph, such as proof trees for a logically inconsistent result (COVADIS), goal environments (KB-Reducer, COVER), or proof residues ((D)EVA).

The types of structural flaws that can be recognized with a Syntactic Checking tool/technique are shown in Table 4.3-2. Not all of the tools find every type of structural flaw, but most of them find most of the flaw types. The knowledge engineer using a structure checker must review the results to verify that the structural flaws found are truly errors and decide how to fix them.

Some of the automated syntax checking tools have additional Semantic Checking features (e.g., (D)EVA and SACCO). These tools use additional domain-specific semantic information about the rule base to check for semantic inconsistencies or incompleteness in the rule base. This additional information is created as an adjunct to the rule base of an expert system by the knowledge engineer, for use by the tool. The additional information may be in the form of: 1) meta-rules (rules about the rules) or constraints on individual variables such as the types, ranges or number of values they may hold; 2) meta-rules or constraints specifying incompatible values between variables; 3) the set-subset information inherent in object/frame hierarchies, or 4) control constraints or meta-rules on the ordering of rule or rule set firing.

Many researchers developing automated tools for Knowledge Acquisition (rule creation) and Knowledge Refinement (rule maintenance) have recognized the need for and incorporated features in their tools for checking the correctness of the new/modified rules. Subclass 3.4.3 in Table 4.3-1 lists these Knowledge Acquisition/Refinement Aids with rule testing features. Some of the tools (e.g., KR-FOCL, KNACK, ODYSSEUS, SEEK2) will even create and propose rules to fill gaps in the knowledge base or propose corrections to rules to the programmer. Thus, they show a kind of machine learning behavior.

Under the **Fault, Failure Analysis** subclass, there have been proposals by Miller and Morell to apply Fault-tree analysis and Heuristic Testing techniques to expert systems, but no experimental evidence of the efficacy of these techniques for expert systems. There is nothing in the Fault, Failure Analysis techniques that make them suitable for only certain types of conventional software, like those in Control or Data Analysis, so this may be a ripe area for exploratory research.

**Table 4.3-2 Types of syntactic errors found by automatic rule base syntax checkers**

**Redundancy**
- identical rules
- identical rule chains

**Subsumption** (a set of rule antecedents/consequents is a subset of another's)
- rule antecedents identical with subsumed consequents
- rule antecedents subsumed with identical consequents
- both rule antecedents and consequents subsumed
- subsumed rule chains of all of the above types

**Inconsistency** (Ambivalence)
- self-contradictory rule
- self-contradictory rule chain
- contradictory pair of rules (two rules with the same antecedents have different consequents)
- contradictory chains of rules

**Circularity**
- self-referent rules
- cycle in rule chain

**Unreachable rules** (include an antecedent that is never an input or created as a consequent by another rule)

**Dead-end rules** (with a consequent that is not a final output or that never satisfies the antecedent(s) of another rule)

**Irrelevant Conditions** (two or more rules can be merged into one by deleting irrelvant antecedents, eg. "if a and b then c";

"if a and not b then c" can become "if a then c")

**Incompleteness**
- not all values for an input variable handled (forward-chaining)
- not all legal combinations of values for variables are handled

Under the **Inspections** subclass, we find a number of techniques have been applied to expert systems. Quite a few projects have convened panels of experts to review the rules in rule bases for correctness, so there are many references under Informed Panel Inspection. DEC has applied Structured Walkthroughs to their XCON and XSEL expert systems, partly as a training device for junior knowledge engineers. Clean Room Techniques have been proposed for expert systems, though not applied.

We are sure Peer Code-Checking is done in many organizations developing expert systems (as it is done at IDA and SAIC) -- it just has not been documented. Desk Checking is widely used in the expert systems community, as in conventional software development: the IBM survey found 28 of 65 respondents used Desk Checking. No references to Formal Customer Review (Informal Customer Reviews however, were mentioned by many interviewees, such as IDA) or Data Interface Inspection were found, though like Peer Code-Checking, we are sure these techniques have been applied to expert systems, and there is every reason for them to be.

It is difficult to differentiate the Static Testing technique of User Interface Inspection from the Dynamic Testing technique of User Interface Testing, because in order to see all the screens and responses of a user interface, one must usually interact with the software, thus making the process dynamic. So, we have categorized User Interface Inspection and Testing techniques together under Dynamic Testing.

As for Standards Audits, DEC has instituted a standard, RIME, for how rules should be written in XCON and XSEL, and performed manual audits of new or modified rules to assure they met the standard before the creation of an automated tool to do the checking. Also, for REALM, an independent audit of the source code comments against a standard was included in the V&V Plan. Since expert systems are a relatively new area of software development, the application of standards to them is also new. As the field matures and standards become more prevalent, the application of this technique will naturally grow.

## 4.4 Dynamic Testing

The techniques categorized under Dynamic Testing are suitable for application during the Implement, Hardware-Software Integration, and Computer System Validation phases of the NSAC-39 Software Development Lifecycle. In Table 4.4-1 presents the subclasses of Dynamic Testing techniques, with references given in the third column for each of these. Each of the subclasses of Dynamic Testing which have been applied to expert systems will be discussed in turn, followed by some general observations.[2]

Under the **General/Statistical** subclass, most of the conventional software testing techniques have been applied to expert systems, and no new expert-system-specific techniques were found. Only Reliability Testing was not applied, and there is no reason for it not to be. In an example of inference engine testing, specifically IBM's The Integrated Reasoning Shell (TIRS), Unit/Module Testing, System Testing, and Regression Testing were applied (Bartschat, 1990). We found the same three well-proven techniques were popular among the large-scale expert system developers we interviewed (DEC, IDA, NASA). Random Testing, Metric-Based Testing, and Software Reliability Estimates have been researched or proposed, but not applied.

The fourth type of General/Statistical Testing is the use of Compilation Testing, a conventional technique that can also be applied to expert systems. The compiler is used to enforce a specified programming style or methodology

[2] The numbering and names of the V&V classes and subclasses differs slightly from the full description that appears in Volume 2, A Survey of Conventional V&V Methods. This is because Volume 2 was revised to take into account the results of this and all the following tasks of this project.

Table 4.4-1 Dynamic Testing V&V techniques applied to expert systems

| | Dynamic Testing Techniques | Description | Applied to Expert System |
|---|---|---|---|
| 3.1 | **General/Statistical** | | |
| .1 | Unit/Module Testing | General testing of single program modules. | Bartschat, '90; Bochsler, '88' <Bell Labs interview>, '91; Mehrotra & Johnson, '90; Ribar, et al, '91; <Szatkowski interview>, '91; <IDA site visit>, '91 |
| .2 | System Testing | Testing of the overall completed software system with test cases representative of general program characteristics including its logic and computation, and its timing. | Bartschat '90; Constatine & Ulvila, '89, '90; Bochsler '88; Ribar, et al, '91; Pathfinder, '91 (V&V Plan); <Bell Labs interview>, '91; EPRI, '87c, '88b; <IDA site visit>, '91 |
| .3 | Random Testing | Selecting test cases according to some random statistical procedure. | Rushby, '88; Rushby & Crow, '90; <WPI site visit>, '91 (Monte-Carlo analysis); Chang & Stachowitz, '88; Chang, et al, '89; DSC, '91; Duke, '88; Oliver '87; Miller, '89b, d, '90a |
| .4 | Compilation Testing | Using compiler diagnostics and problem-reports to find style/method/syntax patterns that are likely (though not always) to be sources of error. | Bahill, '87, '91 (Validator); Heller, '91, <DEC site visit>, '91 (SEAR); Rossomando, '89 (KBLINT, ME); Preece, '90 (Cover) |
| .5 | Reliability Testing | Selecting test cases to exercise particular aspects of the system believed to be unreliable; also extensive testing to assess component failure rates. | None |
| .6 | Regression Testing | Repetition of a test-suite after program modification to assess effects of changes. | Barker & O'Connor, '89; Bachant, '88; <DEC site visit>, '91; Bartschat, '90; Rushby, '88; Griesmer, '90; Oliver, '87; Roscoe, '88; <Bell Labs site visit>, '91; Bond '88; Miller, '89b, d; <NASA site visit>, '91 |

Table 4.4-1 (Continued).

| Dynamic Testing Techniques | Description | Applied to Expert Systems |
|---|---|---|
| .7 Metric-Based Testing | Selection of some aspect of the program for testing on the basis of the value of some metrics computer for it (usually *complexity*) | Rushby, et al, '90; Rushby & Crow, '90 (revealing subdomains); Miller, '90a (complexity factors) |
| .8 Statistical Record-Keeping | Collecting data on errors discovered for particular system modules to suggest which modules should be tested more thoroughly or even redesigned; especially important in the maintenance phase of the life cycle. | Odubiyi, '90 a,b; <IDA site visit>, '91 |
| .9 Software Reliability Estimation | Similar to 3.1.8 but applying sophisticated statistical estimation techniques to fault and error data, to guide continued data collections and to predict system and sub-system failures. | Barrett, '89 & '90 a,b |
| .10 Ad-hoc Testing | Test cases defined at whim by programmer(s) or expert(s) (current state of art, some attempt to "cover" input space). | Baum et al, '89; <Bell Labs site visit>, '91; <DEC site visit>, '91; Hassberger & Lee, '88; Oxman, '91; Rotchild, et al., '90; <Bartschat interview>, '91; Shumaker & Davis, '90; EPRI, '87c; >Friedland interview>, '91; <Mahler interview>, 91; <NASA site visit>, '91 |
| .11 Domain Testing | Analysis of the boundaries and partitions of the input space and selection of interior, boundary, extreme, and external test cases as a function of the orthganality, closeness, symmetry, linearity and convexity of the boundaries. | None |
| .12 Data-Flow Testing | Selection of test cases to explore data anomalies discovered by examination of the program's control flow graph. | None |

**New:** None

Table 4.4-1 (Continued).

| Dynamic Testing Techniques | | Description | Applied to Expert Systems |
|---|---|---|---|
| **3.2** | **Functional Testing** | | |
| .1 | Specific Functional Requirement Testing | Selecting test-cases to assess the implementation of specific required functions. | Bonissone & Wood, '88; Davis & Renckly, '88;; Kirk & Naser, '90; Bowens, '89; Green & Keyes, '87; Pau, '87; Pathfinder, '91 (V&V Plan); Rushby & Crow, '90; Rushby, '88; Bond, '88;; Miller, '89b,d, '90a; EPRI, '97c, '88b |
| .2 | Simulation | Generating special code to emulate various apsects of the to-be-implemented system. | None |
| .3 | Model-Based Testing | Use of an analytic or process model of implemented function. | Cohen et al, '91; Howe & Cohen, '91; Mars & Miller, '86 |
| .4 | Assertion Checking | Bracketing code segments with assertions which can be compiled into executable code to verify assertions during code operation. Similar to the static technique of program proving but involves actual execution of code and assertions. | None |
| .5 | Transaction-flow testing | Identifying the flow of information between people and computers, for user-driven systems, as well as the internal computer processing transformation of that information, and developing a suite of tests to exercise each of the processing steps. | None |

**New:** None

Table 4.4-1 (Continued).

| Dynamic Testing Techniques | Description | Applied to Expert Systems |
|---|---|---|
| **3.3 Realistic Testing** | | |
| .1 Field Testing | Testing of the program under actual installed conditions. | Constatine, '90; Moniger & Stewart, '88; IBM Survey, '90 (in parallel); <Surko interview>, '91 (in parallel); Davis, '88; Fieschi, '90; Preece, '90; Ribar et al, '91; Schumaker & Davis, '90; <IDA site visit>, '91 |
| .2 Scenario Testing | Lab or Field testing with highly realistic cases or situations. | <DEC site visit>, '91; Bonissone & Wood, '88; Hajek et al, '89; EPRI, '87c; Benbasat, '89; Daivs, '88; Kearsley, '88; Moniger & Stewart, '88; Groundwater, '89; Pathfinder, '91 (V&V Plan); EPRI, '87c, '88b; <Moradian interview> '91; Moradian et al, '90; <NASA site visit>, 91 |
| .3 Qualification/Certification Testing | Extensive testing to meet some set of high standards of quality or performance | Moniger & Stewart, '88; Bartschat, '90 |
| .4 Simulator-Based Testing | Using a simulator to generate realistic input data streams to the system to be tested. | <Lee interview>, '91; Duke, '88; <NASA site visit>, '91; Chang & Cheng, '90; Chang et al, '90; <Yyokobayshi interview>, '91; <Owre interview>, '91; Edwards et al, '90; EPRI, '90; Radwan et al, '89; <Bray interview>, '91; <Hajek interview>, '91; Miller, '89b,d; O'Keefe & O'Leary, '91 |
| **New:** None | | |
| **3.4 Stress Testing** | | |
| .1 Stress/Accelerated Life Testing | Exercising the program as rapidly, with as much data input, CPU tasking, and memory load, as possible. | None |
| .2 Robustness Testing | Testing the program with bizarre inputs under variously degraded conditions. | Miller, 90 (suggested) |

Table 4.4-1 (Continued).

| Dynamic Testing Techniques | Description | Applied to Expert Systems |
|---|---|---|
| .3 Limit/Range Testing (Boundary Testing) | Selecting test-cases to test (and exceed) the extreme ranges of allowable limits on variables/parameters. | Duke, '88; Green & Keyes, '87; Vinze et al, '90 |
| .4 Parameter Violation | Determining the various design parameters which led to the present implementation and systematically generating test-cases which violate these parameters. | None |
| **New:** None | | |
| **3.5 Performance Testing** | | |
| .1 Sizing/Memory Testing | Assessing the CPU and memory requirements of the program under various conditions. | Gaschnig, '83; EPRI, '87c |
| .2 Timing/Flow Testing | Assessing the rate of operation (and concurrency) of various program components and the rate of flow of information. | Bonissone & Wood, '88; Rossomando, '89; EPRI, '87c; Gaschnig, '83; Johnson, '88; Lopez, '90; Pau, '87; Bond, '88; Miller, '90a; <IDA site visit>, '91 |
| .3 Bottleneck Testing | Determining the location of undesired delays and processing queries in the program's operation. | <IDA site visit>, '91 |
| .4 Que size, register allocations, paging, etc. | Assessing any other performance aspect of the program. | Gaschnig '83; Pau, '87; EPRI, '87; <IDA site visit>, '91 |
| **New:** None | | |

Table 4.4-1 (Continued).

| Dynamic Testing Techniques | Description | Applied to Expert Systems |
|---|---|---|
| **3.6 Execution Testing** | | |
| .1 Activity Tracing | Monitoring and evaluating the results of a particular program function or activity. | Rossomando, '89 |
| .2 Incremental Execution | Halting program execution at multiple points to assess performance variable values and data storage characteristics. | None |
| .3 Results Monitoring | Similar to 3.6.1 but more focused on a particular outcome regardless of the activity that generated it. | None |
| .4 Thread Testing | Following control and data for a single function through multiple modules. | None |
| .5 Using Generated Explanations | Examining the explanations or rule traces produced by the expert system to evaluate if the reasoning process is correct. Usually the experts do the examination. | Miller, '89b,d; Neches, et al, '85a, b (XPLAIN); Rajamoney & DeJong, '87 |

**New:** None

Table 4.4-1 (Continued).

| Dynamic Testing Techniques | Description | Applied to Expert Systems |
|---|---|---|
| **3.7 Competency Testing** | | |
| .1 Gold Standard | Measuring program results against widely accepted standards. | None |
| **(NEW Subclasses)** | | |
| .1 Ground Truth | Standard is physical evidence in the real world, such as test cases and results constructed from actual incidences. | O'Keefe, et al, '87; T. O'Leary, et al, '90; Rushby, '88; <Moradian interview>, '91 |
| .2 Other Software, Models, Methods | Standard is another software program; a mathematical, physical, or software model; or other methods, possibly manual. | Rushby, '88; Moniger, '88; Daivs, '88 (instruct, methods); Oliver, '87; O'Keefe & O'Leary, '91 |
| .3 Human Experts | Standard is the performance or judgments produced by a human expert or practitioner or agreed upon by a panel of experts. | EPRI, '87c, '88b; Miller, '89b,d; Llinas, '87; Rushby, '88; Constatine, '90; Rotschile, et al, '90; Porter, '88 (Rheum medical cases); Dhuamel, '88 (SES medical cases); Oliver, '87; Bartschat, '90; Skingle, '89; Ribar, et al, '91 (practitioners); Pau, '87 (confusion matrices); IBM survey, '90; Vince, et al, '90 (blind comparison); O'Keefe et al, '87; Cuddigan et al, '87; Benbasat, '89; <DEC site visit>, '91, Fieschi (medical cases) |

Table 4.4-1 (Continued).

| | Dynamic Testing Techniques | Description | Applied to Expert Systems |
|---|---|---|---|
| .4 | Turing Test | Test to determine if the user cannot distinguish the system's behavior from that of a human when interfacing to both via terminal (usually) or when examining paper results. | O'Keefe & O'Leary, '91; Yyu et al, '79, Buchanan & Shortliffe, '85 (MYCIN); Hickam, '85 (ONCOCIN) |
| .2 | Effectiveness Procedures | Assessing the sequential effectiveness of the program against some external standard. | Miller, '89b,d (suggested) |
| .3 | Workplace Averages | Measuring program results against averages established in some workplace. | Constantine '90; Klein, '88; Miller, '89b,d |
| 3.8 | **Interface Testing** | | |
| .1 | Data Interface Testing | Testing the data interface to insure that all aspects of data I/O are correct, including buffering, change detection, checking, etc. | Boschler, '88; Chang, et al, '89; Chang & Cheng, '90; Ribar, et al, '91; <Bell Labs interview>, '91; Miller, '90a |
| .2 | User Interface Testing | Evaluation of the user interface from low-level ergonomic aspects to instrumentation and controls human factors to global consideration of ease-of-use and appropriateness, taking into account CONOPS and information analyses (below). | Schumaker & Davis, '90; DSC, '91; Constantine, '90 (questionnaires, MAU); Preece, '90c; Klein & King, '88 & Klein, '87 (AIQ); Cheng, et al, '90; Andriole & Hopple, '88; EPRI, '87c, '88b (MAU); <Owre interview>, '91; Bochsler, '88; Pau, '87; Miller, '90a; Conrath & Sharma, '91; Pathfinder, '91 (V&V Plan); <IDA site visit>, 91 |
| .3 | Information System Analysis | Determining that operator-needed information is well-organized and is available directly, and quickly, neither too much not too little, neither inaccurate or contradictory. | Cheng, et al, '90; Miller, '90a; EPRI, '87c |

Table 4.4-1 (Continued).

| Dynamic Testing Techniques | | Description | Applied to Expert Systems |
|---|---|---|---|
| .4 | Operational Concept Testing (CONOPS Testing) | Testing that the concept of operations is adequate, appropriate, and sufficiently flexible. | Miller, '90 (suggested) |
| .5 | Organizational Impact Analysis/Testing | Testing or analyzing the effect of the system on the user organization/corporate structure and/or methods after installation. | Conrath & Sharma, '91, <Sharma interview>, '91; DSC, '91; Gowens, '89; Preece, '90c; EPRI, '87c |
| New: None | | | |
| **3.9** | **Structural Testing** | | |
| .1 | Branch Testing | Generating test cases to exercise all branches from conditional or case control structures. | Gupta, et al, '90. |
| .2 | Path Testing | Augmenting branch testing to test various repetitions of program flow through interactive or loop structures of the program. | Chang & Stachowitz, '88; Chang, et al, '89; <Fuji interview>, '91; Hall, et al, '88; Rushby & Crow, '90; Rushby, et al, '90; Rushby, '88; Ribar, et al, '91; Miller, '89b, '90a |
| .3 | Statement Testing | Generating test cases to exercise specific (or all) program statements (or rules for expert systems) in the source code. | Groundwater, '89; Ayel & Vignollet, '89; Chang & Stachowitz, '88; Chang, et al, '89; <Surko interview>, '91; Gupta & Biegel, '90 (RITCaG); Bochsler, '88; IBM survey, '90; Miller, '90a; Roscoe, '88; EPRI, '87c, '88b; <IDA site visit>, '91 |
| .4 | Call-Pair Testing | Developing cases to test the argument and parameter interfaces among programs. | None |
| .5 | Linear Code Sequence and Jump (LCSAJ) | Selecting of test cases based on control-flow analysis, similar to branch testing, but often used in "lower" level languages such as assembler. | None |

Table 4.4-1 (Continued).

| Dynamic Testing Techniques | Description | Applied to Expert Systems |
|---|---|---|
| .6 Test-Coverage Analyzer | Determining what statements, paths, branches, etc. are exercised by a set of test cases and/or automatically generating test cases to test all the statements, paths, branches, etc. of a system. | Marcus, '88 (SALT); Bachant, '88; Vignollet & Ayel, '90 a,b (SYCOJET) |
| .7 Conditional Testing | Testing or statements involving Boolean or Relational (e.g., "A<B") tests. Selecting test cases corresponding to values equal to, less than, and greater than the values of the conditions. | Gupta & Biegel, '90 (RITCaG) |
| **3.10 Error-Introduction Testing** | | |
| .1 Error Seeding | Introducing errors of arbitrary kinds in a software system. | None |
| .2 Fault Insertion | Introducing modifications to a program which will induce a failure or fault of a particular kind. | Rushby, '88 (suggested) |
| .3 Mutation Testing | Introducing errors of various kinds in a program and determining whether a given suite of test-cases detect the errors. Used to assess power of one's testing techniques for discovering problems. | Hall & Heinze, '89; Morell, '89 |

**New:** None

that is designed to reduce the chances of producing errors in the code. An example is DEC's RIME methodology for the XCON and XSEL expert systems that is enforced by the SEAR compiler. Violations in the prescribed style are flagged as errors or warnings by the compiler even though they may produce correct results, and the programmer uses these messages to make his/her code conform to the recommended style/methodology.

Under the **Functional Testing** subclass, Functional Requirement Testing has been discussed and applied extensively. Cohen and his associates at the University of Massachusetts have applied Model-Based Testing to their Phoenix expert system that plans forest fire fighting activities (Cohen et al, 1990), and Mars and Miller (1986) discuss its application to medical diagnostic expert systems. Simulation testing (vs. Simulator-Based Testing under Realistic Testing) has not been mentioned in the literature, though we have anecdotal evidence that it is used during development to "stub out" portions of the expert system that have not yet been implemented. Assertion Checking depends on Program Proving techniques (see Section 4.3) being mature enough to result in tools for compiling and executing program assertions. Since Program Proving techniques applied to expert systems are still in their infancy, we would not expect to see Assertion Checking research yet.

Under the **Realistic Testing** subclass, all the conventional techniques have been applied. Qualification/Certification Testing has been applied to at least one inference engine, TIRS, as expected (Bartschat, 1990). Also, a large number of the nuclear power application expert systems have used Simulator-Based Testing, as well as other applied expert systems. Many developers have to rely on Field or Scenario Testing, unfortunately, to find many of their errors, because they do not use Static Testing or early Dynamic Testing techniques to find errors earlier in the development process.

Under the **Stress Testing** subclass, Limit/Range Testing has been applied and Robustness Testing has been proposed. All the types of Stress Testing could readily be applied to expert systems, so we see this as a ripe area for future research. All the techniques under the **Performance Testing** subclass have been applied. IDA is enthusiastic about Performance Testing, as it also found errors unrelated to performance, that had not been detected by other methods.

Under the **Execution Testing** subclass, only the new technique developed specifically for expert systems (Using Generated Explanations) has been applied. The other techniques are specific to the control structure of conventional software, so we would not expect them to be a good fit for expert systems.

The Gold Standard technique under the **Competency Testing** subclass has been the singularly most popular Dynamic Testing technique applied to expert systems. Since an expert system by definition models human expertise, it is logical to test its capabilities and answers against those of an expert or experts. The other Competency Testing techniques, Effectiveness Procedures and Workplace Averages, have been proposed and researched, but not yet applied.

The bulk of the work in the **Interface Testing** subclass has been in User Interface Testing. This involves having users actually use the system and provide feedback in the form of spoken or written comments, questionnaires, problem reports, or other means. If questionnaires or other human factors-based testing instruments are used, they are often subjected to Multi-Attribute Utility Analysis (MAU) to determine the relative importance of various features of the user interface (see Constantine, 1990 and DSC, 1991 in particular). Many researchers have developed their own detailed lists of rating factors for evaluating expert systems. For a thorough example, see Klein's AIQ list of factors (Klein & King, 1988, Klein, 1987).

Under the **Structural Testing** subclass, all but the Call-Pair Testing and Linear Code Sequence and Jump techniques have been researched for and/or applied to expert systems. These two techniques are dependent on the control structure of conventional software, so we would not expect to see them applied to expert systems. The bulk of the work has focused on designing one or more test cases for each rule in the system (Statement Testing) or converting the rule base into a directed graph representation and testing each path through the graph (Path Testing).

The last subclass, **Error-Introduction Testing**, has seen little work, other than proposals or discussions. Like Stress Testing, there is no reason these techniques could not be applied to expert systems, and this may be a ripe area for future research.

Three general observations on the tabled results may be made. The first is that, in contrast to the previous two major classes of techniques, most of the conventional Dynamic Testing techniques have been employed for expert systems, to a greater or lesser extent. Most of the few unused techniques are those which require the explicit control-paths of conventional programs missing from expert systems (e.g., Thread Testing, Call-Pair Testing, Linear Code Sequence and Jump, Transaction-Flow Testing, Control-Flow Testing, Data-Flow Testing).

The second observation is that the frequency of usage of Dynamic Testing techniques for expert systems is different from that expected for conventional systems. The four most frequently-used subclasses are (from most to less): General/Statistical (2.1), Competency (2.7), Realistic (2.3), and Interface testing (2.8). In contrast, based on our wide Activity 1 survey of conventional techniques, we believe that **Functional** and **Structural Testing** are the most frequently used Dynamic Testing subclasses, along with General/Statistical, for conventional software. Sufficient sampling of test cases generated from these three subclasses will insure competent testing of most procedural programs, across wide variations in application, structure, and function. The differing results for expert systems can be interpreted as reflecting the narrower and common focus of most of these systems: the development of high-level decision-support systems which demonstrate "intelligence" and/or "expertise" in their operation. Given such a focus, it is quite reasonable to emphasize the competency (vis-a-vis human capability), the performance under realistic conditions, and the interfaces of these systems. These three categories for expert systems correspond greatly to the general Functional-testing category for conventional systems. The absence of a high Structural Testing emphasis is perhaps partially explainable by the typical absence of explicit control paths in expert systems programs and by their relative lack of modularity (especially for rule based systems). Also, the static techniques used to examine knowledge bases provide means of examining structure without the need for dynamic execution.

The third major observation is that there are few new Dynamic Testing methods created especially for expert systems (most of these being new instances of the Gold Standard Subclass, 2.7.1). One could reasonably expect new techniques would be needed for several reasons: 1) to test inference engine reasoning and decision processes by themselves; 2) to dynamically test rule conflict-resolution effects and the effect of rule-ordering in interaction with the inference engine; 3) to test the various shell utilities; and 4) to test the various interfaces. Nevertheless, no special-purpose techniques were discovered for testing the decision procedures of inference engines per se. Rule conflict-sets and goal-tree backtracking traces are part of some commercial AI products' development environment (e.g., Quintus' PROLOG), but such usage was not reported in the literature we covered. No techniques for measuring tradeoffs among alternative knowledge representations (e.g., rules vs. frames vs. external databases vs. demons) were cited or even discussed as being potentially useful. Finally, there were almost no discussions of need for testing the interfaces between expert system knowledge representations and conventional procedural software.

A final observation derives not from the table results but from understanding the manner in which the various Dynamic Testing techniques were employed (the information being obtained from our phone interviews, site visits, and reading of the literature). When we really examine how this testing was accomplished, we must conclude that Dynamic Testing of expert systems is typically not very sophisticated, as compared to the rigor usually employed for conventional systems. It was not terribly uncommon for us to hear the opinion that testing was much less necessary for expert systems (even that there's no sense in testing the system, since you can't really test the expert human from whom the knowledge was derived). One possible explanation of this finding is based on the relative immaturity of expert system software development. Whereas principles of good software engineering and management have been developed and refined over the last 20 years for conventional programs, expert systems development has few established principles or standards, and many of the developers have little conventional software experience.

## 4.5 The State-of-the-Art

There is wide agreement on the vast need for expert system V&V (Aldridge, 1988, Bellman, 1989 & 1990b, Beltracchi, 1990, Cohen & Howe, 1988, Culbert et al, 1987c, Cullyer, 1989, Finlay et al, 1988, Friedman et al, 1988, Gearhart, 1989, Geissman, 1988a, b, Goodwin & Robertson, 1988, Green, 1987, Harrison, 1989, Hofmeister, 1986, IEE, 1987, Krishnamurthy et al, 1987, Lane, 1986, Linden, 1988, Llinas, 1987, Naser, 1988, O'Keefe, 1988, Rushby, 1988) particularly as more and more expert systems are being applied operationally. Within the last four to five years, there has been an explosive growth of papers in the field. The field has now reached a maturity where expert system V&V techniques are being implemented in automated tools and being applied to operational and developmental expert systems to test their efficacy in finding errors.

Within the expert system V&V class of **Requirements/Design Testing**, actual applications to real expert systems exist for: Requirements Language Analysis, Mathematical Verification of Requirements, Formal Requirements Review, Requirements Tracing, and Formal Design Review. The subclasses which have not received attention include: Requirements Language Processing, Design Compliance Analysis, Design Simulation, Program Description Language Analysis and Processing, and Critical Timing/Flow Analysis. The relatively sparse activity in this expert system V&V class can be attributed to the lack of or "after-the-fact" incorporation of a requirements and/or specification document for expert systems, though there are vocal proponents in the industry pushing for the writing of these documents for expert systems (e.g., Green, 1988, Miller, 1989c, 1990c, Hamilton et al, 1991, IBM, 1990, Linden, 1989 & 1990, Logananthara], 1990).

As previously discussed, the **Static Testing** V&V class consists of six subclasses. These six subclasses are further divided into a total of 46 techniques. Within the Algorithm Analysis subclass only two of the ten techniques, Program Proving and Statistical Sensitivity Analysis (a new technique), have received any significant attention and application to expert systems. The remaining eight techniques (Analytic Modeling, Cause-Effect Analysis, Symbolic Execution, Trace-Assertion Method, Functional Abstraction, L-D Relation Methods, A-7 Table Formats, and Metric Analyses) have not been utilized in expert system V&V.

None of the seven techniques in the Control Analysis subclass have been used for expert systems. Within the Data Analysis subclass, there have been a few applications of Dependency Analysis and Cross-Reference List Generator, but there is no evidence of any use of the remaining eight techniques. The majority of techniques in both these subclasses are dependent on the conventional structure of conventional software, so it would be surprising to see many attempts to apply them to expert systems.

Defect Analysis is the Static Testing subclass with the most activity and applications to expert system V&V. This includes all three techniques, but the greatest amount of work has centered on Syntactic Checking and Knowledge Acquisition/ Refinement Aids. This subclass has received the most attention in Static Testing of expert systems and contains the vast majority of automated tools for expert system V&V.

There has been meager activity in the Fault and Failure Analysis subclass. Only Fault-tree (Event-tree) Analysis and Heuristic Testing have been applied to some extent whereas the remaining five techniques (Failure Mode and Effects Analysis, Criticality Analysis, Failure Modeling, Hazards/Safety Analysis, and Common-Cause Failure) have not received any attention. Some of these techniques may not have been considered because they are usually associated with software that directly affects safety systems and most expert system software, especially in the nuclear industry, has not yet been involved with these systems. We see this to be a fertile area for future research.

The final subclass of Static Testing techniques is Inspections, which consists of nine techniques. Five of these have received some attention for expert system applications, with the majority of attention being paid to Informed Panel Inspections, Structured Walkthroughs, and Standards Audits. There have been one or two cases of Clean-Room Techniques and Desk Checking. No activity has been published in Peer Code Checking, Formal Customer Review, Data Interface Inspections, and User Interface Inspection, but we have anecdotal evidence that all have been applied to expert systems.

As with Static Testing, **Dynamic Testing** techniques have been divided into a total of ten subclasses which are themselves comprised of a total of 67 techniques. However, unlike Static Testing, there is considerably greater breadth in the use of Dynamic Testing techniques in that 40 of these 67 techniques had activity with regard to expert system applications. The Dynamic Testing subclasses which have shown the greatest activity include General/Statistical Testing, Realistic Testing, Competency Testing, Active Interface Testing, and Structural Testing. Within these classes, the most used techniques are: Module, System, Random, Regression, and Ad Hoc Testing; Specific Functional Requirement Testing; Field, Scenario, and Simulator-based Testing; Human Expert Competency Testing; User Interface Testing; and Statement Testing.

The applicable Dynamic Testing subclasses with the lowest activity in expert system V&V applications are Stress Testing, Execution Testing, and Error-Introduction Testing, so these may also be fertile areas for future research. The 11 techniques which were not employed are: Reliability Testing, Assertion Checking, Simulation, Accelerated Life Testing, Parameter Violation, Incremental Execution, Results Monitoring, Thread Testing, Call Pair Testing, Linear Code Sequence and Jump, and Error Seeding. Many of these techniques are highly dependent on the conventional structure of conventional software, so we would not expect to see them applied to expert systems.

The state-of-the-art in expert system verification and validation testing techniques directly reflects the immaturity of expert system technology when compared to conventional software and the large chasm between new expert system development interest and resources allocated to expert system V&V. The practice of not developing requirements and design documentation coupled with a concomitant deficiency in the involvement of V&V during the early stages of the software development process has resulted in the emphasis on expert system V&V being directed towards Dynamic Testing and away from Requirements/Design Testing. Most of the scientists and engineers involved in the development and V&V of expert systems do not have strong backgrounds in conventional software V&V where V&V activities are ingrained as an integral component of the entire software lifecycle. This is reflected in the dearth of Requirements/Design Testing techniques for expert systems and further substantiates the reliance on Dynamic Testing for expert systems. In summary, although there are some notable exceptions in the areas of Requirements/Design Testing and Static Testing techniques, the bulk of current expert system verification and validation activities occur only after the system is implemented.

# 5. AUTOMATED TOOLS

Expert system researchers and developers have created a number of automated tools to assist in performing V&V on expert systems. These tools perform both syntax and semantics checking of the rule base, serve as intelligent compilers, perform automated knowledge acquisition and refinement, and assist in test case generation and assessment. Most of the tools are in-house efforts and are not yet commercial ventures, though a few are available commercially.

Table 5.0-1 lists all 29 of the tools we found and descriptive information about each one. The Tool Type column lists the classification category(ies) within which each tool falls in our V&V technique hierarchy described in Section 4.0. The Availability column states whether a tool is a non-maintained historical tool, is commercially available, is a proprietary in-house corporate venture, or is a university research tool. The later may be available, upon negotiation with the university point of contact, but will probably be provided as is, with no documentation or support. The Knowledge Base Formats column lists the rule base formats the tool can handle, usually by the name(s) of the expert system building shell(s) within which the rules are created.

The Error Types Handled column states whether the tool assists in finding static (not requiring execution of the rule base) vs. dynamic errors, or anomalies (structural flaws) vs. invalidities (true semantic errors). The topic of error types is discussed in greater detail in Section 6.2. The Level of Sophistication column gives an indication of how complete the tool is and whether it incorporates state-of-the-art techniques compared to other tools in its classification hierarchy. Thus if a number of years has been spent developing the tool by a number of people, it is currently up-to-date with respect to the technology, the software is stable and bug-free, and it is close to being a commercializable product, its level of sophistication would be high.

The tools will be described in sections below according to the classification category in which they fall. In some cases, a tool provides services in more than one of the classification categories. In these cases, the characteristics of the tool relevant to each category will be discussed within each of the appropriate sections.

## 5.1 Syntax Checking Tools

The large majority of automated tools for expert system V&V perform syntax checking of rule bases. The "grandfather" of all tools in this category was the Rule Checker Program (RCP) for the ONCOCIN expert system developed in the early 1980's (Suwa et al, 1982). It was the first tool to check for rule conflicts, redundancy, subsumption, and missing rules to handle certain input(s) by constructing a table of rules (rows) vs. condition variables (columns) that appeared in the rule antecedent. RCP was an integral part of ONCOCIN, only handled rules in that format, and is no longer maintained.

ARC's predecessor, CHECK, was the second well-known syntax checker after RCP, adding checks for circular rules, unreachable clauses, and dead-end clauses over the RCP checks (Nguyen et al, 1985, 1987). It converted rules into the same table format that RCP used, called a "dependency chart" by the developers. Circular rules were found by converting the rule base into a directed graph structure. When CHECK was converted from working on rules in the LES (Lockheed Expert System) shell format to the ART shell format, it was renamed ARC for ART Rule Checker. This version included a check for unnecessary conditions and conflicting rule chains, which were discovered by searching the directed graph representation. Work on ARC at Lockheed has now been superseded by work on a new tool, (D)EVA.

Another historical tool, ESC (for Expert System Checker), developed at the University of Wisconsin, also used the same table representation as RCP. The developers called it a "decision table" (Cragun & Steudel, 1987).

Table 5.0-1 Description of automated tools for expert system V&V

| Name | Tool Type | PoC & Affiliation | Availability | Knowledge Base Formats | Error Types Handled | Level of Sophistication* | Technology Basis |
|---|---|---|---|---|---|---|---|
| ARC (formerly CHECK) | Syntactic Checking | Nguyen; Lockheed | Historic, Not Maintained | ART (formerly LES) | Static Anomalies | Low | Converts rules to dependency chart |
| Aquinas (formerly ETS) | Knowledge Acquisition | Boose, Baum, Bradshaw; Boeing | Corporate Proprietary | Erasmus (formerly OSP5 & KS-300) | Static Anomalies, Dynamic Invalidities | Medium | Repertory grid representation of rules, measures system performance with regression test codes |
| COVADIS | Syntactic Checking | Rousset; Universite d'Orsay | Research tool | MORSE | Static Anomalies | Low | Constructs all possible proof trees for inconsistent result (prove "FALSE") and presents to KE for correction |
| COVER | Syntactic Checking; Intelligent Compiler | Preece; Concordia University | Research tool | OPS5, XIPlus, Crystal | Static Anomalies | High | Converts rules to dependency chart and generate goal environments |
| CRSV | Syntactic Checking | Culbert; NASA | Available with CLIPS | CLIPS | Static Anomalies | Low | Generates execution traces and compares terms to strong data typing meta-constraints |
| (D)EVA | Syntactic Checking; Semantic Checking | Combs, Stachowitz, Chang, McGuire; Lockheed | Government software available from RADC | CLIPS, KEE, ART | Static Anomalies; Static Invalidities | Very High | Converts rules to directed graph, construct proof residues, tests against control semantic meta-constraints, uses frame taxonomies to find object/rule gaps |

NOTE (*): Level of sophistication indicates, for its tool type, the completeness and soundness of the code, its nearness to being a useable product and whether recent technology for that tool type is incorporated.

Table 5.0-1 (Continued).

| Name | Tool Type | PoC & Affiliation | Availability | Knowledge Base Formats | Error Types Handled | Level of Sophis-tication* | Technology Basis |
|---|---|---|---|---|---|---|---|
| ESC | Syntactic Checking | Cragun; IBM | Historic, not maintained | Insight II | Static Anomalies | Low | Converts rules to decision tables |
| ESPE | Syntactic Checking | Franklin; Rensselaer Polytechnic Institute | Research tool | ESDE | Static Anomalies | Low | Converts rules to directed graph |
| Evidence Flow Graphs | Syntactic & Semantic Checking; Random Test Generator | Green, Becker, Duckworth; Worcester Polytechnic Univ. | Nearing commercial-ization | CLIPS, ART, AFL | Static Anomalies, Dynamic Invalidities | High | Converts rules to evidence flow graphs, generate Monte-Carlo tests to test specification meta-rules |
| Expert/ Measure | Test Case Evaluation | Parsaye; Intelligence Ware | Commercial product | Auto-Intelligence | None Directly | Low | Statistics |
| INDE | Syntactic Checking | Pipard | Research tool | Forward chaining | Static Anomalies | Low | Converts rules to Petri net |
| KB-Reducer | Syntactic Checking | Ginsberg; AT&T Bell Laboratories | Historic, not maintained | Own rule format | Static Anomalies | Low | Generates goal environments |
| KR-FOCL | Knowledge Refinement | Pazzani; University of California, Irvine | Research tool | FOCL | Dynamic Invalidities | Medium | Explanation-based and empirical machine learning |

NOTE (*): Level of sophistication indicates, for its tool type, the completeness and soundness of the code, its nearness to being a useable product and whether recent technology for that tool type is incorporated.

Table 5.0-1 (Continued).

| Name | Tool Type | PoC & Affiliation | Availability | Knowledge Base Formats | Error Types Handled | Level of Sophistication* | Technology Basis |
|---|---|---|---|---|---|---|---|
| KNACK | Knowledge Acquisition | Klinker, McDermott; Carnegie-Mellon University | Historic, not maintained | Own semantic network format, specific to electromechanical systems | Static Anomalies | Low | Checks for gaps, inconsistencies in semantic network |
| LAS | Knowledge Refinement | Smith; Schlumberger-Doll Research | Historic, not maintained | Own rule format | Dynamic Invalidities | Low | Learning from justification structures (explanations) |
| MOLE (formerly MORE) | Knowledge Acquisition/ Refinement | Eshelman, McDermott; Carnegie-Mellon University | Historic, not maintained | Own semantic network format, specific to diagnostic systems | Static Anomalies; Dynamic Invalidities | Low | Disambiguates under-specified network & recognizes structural inadequacies based on test codes |
| ODYSSEUS | Knowledge Refinement | Park, Wilkins; University of Illinois | Research tool | MINERVA (formerly HERCULES) | Static Invalidities | Low | Explanation-based learning using context analysis |
| RCP for ONCOCIN | Syntactic Checking | Suwa, Shortliffe; Stanford University | Historic, not maintained | ONCOCIN | Static Anomalies | Low | Generates table of antecedent/consequent values relationships |
| RITCaG | Test Case Generator | Gupta; University of Central Florida | Research tool | ART | Dynamic Invalidities | Low | Generates test case per rule, condition, or context (rule set) |
| SACCO | Syntactic Checking; Semantic Checking | Laurent, Ayel; University of Savoie | Research tool | KOOL | Static Anomalies | Low | Constructs proofs for conjectures of incoherence provided by KE |

Table 5.0-1 (Continued).

| Name | Tool Type | PoC & Affiliation | Availability | Knowledge Base Formats | Error Types Handled | Level of Sophis-tication* | Technology Basis |
|---|---|---|---|---|---|---|---|
| SALT | Knowledge Acquisition/ Refinement | Marcus; Boeing | Historic, not maintained | Own dependency network format for constraint-satisfaction systems | Static Anomalies | Low | Analyzes dependency network; performs test coverage analysis |
| SEAR | Intelligent Compiler | Lutsky; DEC | Corporate proprietary | OPS5 | Static Anomalies | Medium | Checks for violations of R1ME methodology |
| SEEK2 | Knowledge Refinement | Ginsberg; AT&T Bell Laboratories | Historic, not maintained | EXPERT | Dynamic Invalidities | Medium | Uses statistics & heuristics to select rules for refinement, suggests refinements |
| SOAR | Knowledge Refinement | Laird; University of Michigan | Research tool | Own productions format | Dynamic Invalidities | Low | Modifies control preferences versus productions themselves |
| Sweeping Code | Knowledge Refinement | Lenat; MCC | Corporate proprietary | CycL | Static Anomalies | Low | Scans frame base for semantic constraint violations & suggests fixes |
| Sycojet | Test Case Generator | Vignollet; University of Savoie | Research tool | KOOL | Dynamic Invalidities | Low | Converts rule base to precedence graph, generates cases to cover data space and/or paths |
| Teiresias | Knowledge Acquisition | Davis; Massachusetts Institute of Technology | Historic, not maintained | EMYCIN | Dynamic Invalidities | Low | Model-based reasoning |

Table 5.0-1 (Continued).

| Name | Tool Type | PoC & Affiliation | Availability | Knowledge Base Formats | Error Types Handled | Level of Sophis-tication* | Technology Basis |
|---|---|---|---|---|---|---|---|
| TVE | Syntactic Checking | Krishnamurthy, Sztipanovits; Vanderbilt University | Historic, not maintained | OPS5 | Static Anomalies | Low | Converts rules to directed graph |
| Validator | Intelligent Compiler; Syntactic Checking | Bahill; University of Arizona | Commercial product | Backward-chaining rule bases, e.g. M.1 | Static Anomalies | Medium | Unknown |

It added no new checks to those performed by RCP but was applied to a different rule format, Insight II. TVE (Test and Validation Environment), also a historical tool, was developed at Vanderbilt University (Krishnamurthy et al, 1987). It converted the rule base into a directed graph representation to find cycles and provide partitionings of the graph based on various attributes for the knowledge engineer to view and check for errors.

The Expert System Parsing Environment (ESPE) tool developed at Rensselaer Polytechnic University followed a new tack (Franklin et al, 1988, 1989). It converted the rule base into a directed acyclic graph, from which were produced path counts, plots, and sensitivity analyses of the differences and similarities between goal and input values for review by the knowledge engineer. For instance, if two goal values depended on almost the same input value sets a potential error could exist, to be determined by the knowledge engineer. The error could be that one goal is a misspelling of the other, or one or more input values are wrong for one of the goals, or an additional input variable needs to be added to distinguish the goals better.

European efforts at expert system V&V tool development for syntax checking include COVADIS, INDE, and SACCO. COVADIS, developed at Universite d'Orsay (Rousset, 1988a, b) for a forward-chaining shell called MORSE, detects errors in rule bases by attempting to construct all possible proofs for FALSE, thus exposing inconsistencies. INDE (Lopez, 1990) also works on forward-chaining rule bases, converting them into a Petri net (similar to a directed graph) to detect inconsistencies and non-fireable rules. Similarly to COVADIS, SACCO constructs proofs for "conjectures of incoherence" provided by the knowledge engineer. Any successful proofs indicate potential errors. Some of the "conjectures of incoherence" may be structural (redundancy, conflict, cycles, unnecessary conditions, etc.) and some may be semantic, as discussed in Section 5.2.

Researchers at Worcester Polytechnic Institute have developed a methodology for converting rule bases into a directed graph structure they call an evidence flow graph (Becker et al, 1989 a, b, 1990, 1991). During translation into the graph structure, the system finds errors such as outputs not used as inputs by later rules, unused inputs, and lexical errors. By analyzing the graph, the system finds unreachable conclusions or conclusions that are always true and semantic violations (see Section 5.2 below). The graph structure is then used to generate test cases (see Section 5.5 below).

Two commercially available tools for syntax checking include CRSV (Cross Reference, Style, and Verification), available with the CLIPS expert system shell from NASA, and Validator, available from Terry Bahill at the University of Arizona. Validator also provides intelligent compiler capabilities, as described in Section 5.4 below. CSRV (Culbert & Savely, 1988, 1989) uses strong variable typing rules to check a rule base. These rules may specify, for instance, that a variable may only take on a positive integer value and nothing else. Validator's (Bahill, 1987, 1991, Kang & Bahill, 1990) syntactic error checker searches for use of illegal values, unused variable values, and errors due to incorrect use of negation in the antecedent or consequent of rules in addition to compile-types errors. The chaining thread tracer component will find dead-end and unreachable rules in either backward or forward chaining rule bases. Lastly, the completeness module will check for unused variables and redundant methods, rules or variables. Validator has been applied to a number of small expert systems and its performance is well documented in Dr. Bahill's book (Bahill, 1991).

KB-Reducer, developed at AT&T Bell Laboratories (Ginsberg, 1988a, b, 1990) was a historical tool that used a different representation than a table or directed graph to find rule base errors. The expert system is treated as a function that is transformed into a set of functions, one for each possible conclusion, that consists solely of input data terms. These functions are called "goal environments." By examining the intermediate results in producing these functions, the system detects inconsistencies and redundancies in the rule base.

The COVER tool developed at Concordia University (Preece, 1990 a, b, c, Preece & Shingal, 1991) uses the same "goal environment" technology as KB-Reducer did. They have spent considerable effort on developing heuristics to reduce the potential combinatorial explosion that could result from attempting to generate all possible goal environments. The tool finds a number of different types of errors:

- Redundant, conflicting and subsumed rules,
- Redundant, conflicting and subsumed rule chains,
- Cycles,
- Unreferenced and dead-end rules,
- Unsatisfiable conditions,
- Missing values,
- Missing rules, and
- Lexical errors as described in Section 5.4 below.

COVER has been applied to three expert system with hundreds of rules each, and seems to be fairly mature.

The most sophisticated tool for static checking of expert systems is the (D)EVA tool (DARPA Expert system Validation Associate) developed by Lockheed Corporation under DARPA (Defense Advanced Research Projects Agency) sponsorship (Stachowitz et al, 1987 a, b, 1988, Chang, et al, 1990 a, b, Burris, McGuire, 1990, McGuire & Stiles, 1990). The underlying structure used by (D)EVA for structure checking is the directed graph structure of the rule base used by most of the previously described tools. For redundancy and inconsistency checking, a restricted generate-and-test approach is used to generate a subset of possible input scenarios and test the rule base with them to check for anomalies. This approach is similar to COVADIS and SACCO, but uses heuristics to limit the number of scenarios tested. The tool also generates and uses "proof residues" for each goal, which is the set of conditions missing from the fact-base which are needed to prove the goal.

Quite a few checks are performed by (D)EVA. These include:

- Structure checks for:
  - dead-end and unreachable rules
  - subsumption
  - implication (redundancy via backward chaining, query-subquery, and residues)
  - cycles
  - irrelevance
  - ambiguity

- Logic checks for:
  - rule inconsistency
  - add/delete ambiguity
  - conflicts
  - add/delete conflicts

- Semantic checks (described in Section 5.2)

- Completeness checks (described in Section 5.2)
- Control checks (described in Section 5.2)
- Nonmonotonic reasoning checks (described in Section 5.2)
- Rule refiner aid (described in Section 5.3)

Since (D)EVA was developed under government sponsorship, it maybe available from the Rome Air Development Center, DARPA's contracting agent for the work.

## 5.2 <u>Semantic Checking Tools</u>

Three of the tools described in Section 5.1 above also perform some semantic checking. These are SACCO, Evidence Flow Graphs, and (D)EVA. As stated in Section 5.1, SACCO performs checks for violations of coherence constraints provided by the knowledge engineer. Some of those constraints may be semantic, such as: 1) incompatibilities between variable values, 2) authorized values and maximum arity of variables, and 3) specification of exclusive attributes and values thereof. In the Evidence Flow Graphs system, meta-knowledge can be specified by the knowledge engineer, both to guide test generation (see 5.5 below) and as constraints for semantic checking. These constraints include: 1) incompatibilities like SACCO, 2) sequencing, 3) authorized values for variables, and 4) conditional combinations of the first three.

(D)EVA includes many of the same semantic checks as SACCO and Evidence Flow Graphs, but adds many more. It performs the following types of semantic checks, guided by meta-constraints provided by the knowledge engineer:

- General semantic checks
  - incompatibilities
  - min/max cardinality of variables
  - authorized values
  - allowable subrelations in class hierarchies
  - interstate (of the knowledge base) integrity over time

- Completeness checks
  - frame omissions (incomplete class taxonomy, incomplete slot values, incomplete relation taxonomy)
  - rule omissions (logical completeness, numerical completeness, some subclasses in frame taxonomy not covered by rules)

- Control checks
  - sequencing of rules
  - necessary relationships between rules
  - exclusion relationships between rules
  - conditional relationships between rules
  - backward chaining interference

- Nonmonotonic reasoning checks
  - useless new world action rules
  - impossible plans

## 5.3 Knowledge Acquisition/Refinement Tools

A lot of research has occurred in the last decade in techniques for automating or semi-automating the knowledge acquisition and refinement process. We will not report on all that work here, but only on those knowledge acquisition/refinement tools that assist in correcting or testing the knowledge base during the process of knowledge acquisition or refinement. The "grandfather" of the automated knowledge acquisition/refinement systems was Teiresias, developed at Stanford University (Davis, 1981). The system used a meta-model of what it knew to assist an expert in tracking down the source of a wrong answer in the knowledge base. The source may have been a rule that fired when it should not have (requiring an addition, deletion, or change to the rule antecedent conditions) or a missing rule that should provide the correct answer or inhibit the rule providing the incorrect answer.

Three historical tools were developed by graduate students under the tutelage of John McDermott at Carnegie-Mellon University: MORE, MOLE, and KNACK. All worked on semantic network-based knowledge bases. After a user built a diagnostic knowledge base with MORE (Kahn et al, 1985), it looked for weaknesses in the generated rules according to eight strategies and looked for potential inconsistencies in the way a user assigned confidence factors to rules. MOLE, an enhancement of MORE, (Eshelman & McDermott, 1986) performed static analysis on the constructed semantic network to: 1) disambiguate an under-specified network, 2) assign support values, and 3) recognize structural inadequacies. It performed dynamic analysis on test cases to help: 1) discover missing knowledge, 2) guide in the revision of support values, and 3) further disambiguate the network. After initial knowledge acquisition, KNACK (Klinker et al, 1987) would search the knowledge base for gaps or conflicts and prompt the developer for corrections.

SEEK2 (Ginsberg et al, 1985, Ginsberg, 1986) was developed at Rutgers University, based on a prior rule refinement system named SEEK. It used statistics and heuristics to select potentially erroneous rules and to suggest refinements. SALT (Marcus, 1988) built a dependency network-based knowledge base for constraint-satisfaction problems, then analyzed that network for loops, gaps, overlaps, and potential interactions between constraints to suggest revisions to the user. SALT also generated a list of all the constraints and fixes so a later module could check off what had been tested as each test case was run. Both of these tools were historic and are no longer maintained.

Aquinas is a current knowledge acquisition system, based on the earlier ETS system, which closely followed Teiresias and is commonly quoted in the literature. Aquinas and ETS were developed at Boeing Computer Services (Baum et al, 1989, Boose, 1984). Aquinas provides facilities for creating, storing, and rerunning regression test case sets for the knowledge bases constructed with it. Soar (Laird, 1988) is another current system which, like Aquinas, is still in use. It was developed at the University of Michigan. Unlike other systems, Soar does not modify or delete an incorrect rule, instead it modifies the decisions within which the rule may be used via a process called "chunking."

The next three systems, like Soar, used variations of explanation-based learning to perform knowledge acquisition (Doyle 1986). LAS, the Learning Apprentice System, (Smith et al 1985) was a historical system developed at Schlumberger-Doll Research. It built a dependency network they called a "justification structure" from its learned rule base in order to diagnose errors. It could find overspecific or overgeneral rule antecedents or consequents, numeric parameter errors, and symbolic parameter errors. ODYSSEUS, a current research tool, based on the same principle of apprenticeship learning (learning by doing) as LAS, was developed at the University of Illinois (Wilkins, 1988, Park & Wilkins, 1990). A preprocessing stage removes incorrect and conflicting knowledge, and apprenticeship learning focuses on missing knowledge.

KR-FOCL (Pazzani & Brunk, 1990) was developed at the University of California at Irvine and is an extension of Quinlan's FOIL system, which is an extension of his well-known ID3 system (Quinlan, 1986, Burke & McNenny) for inducing knowledge bases from data sets. KR-FOCL can handle errors caused by overly specific or overly general antecedents based on missing or extra conditions or literals within conditions.

## 5.4 Intelligent Compilers

Many of the expert system building tools perform some minimally intelligent syntax checks at compile time, looking for use of undefined variables, incorrect rule formats, etc. Some of the expert system V&V tools extend this idea to include more ways for preventing the programmer from making mistakes, usually by checking for common simple syntax errors or enforcing some style on the rule base. These tools are the Validator and COVER tools described earlier in Section 5.1 and the SEAR intelligent compiler developed at Digital Equipment Corporation (DEC). Validator will check for unclosed comments, use of reserved words, misspellings, and other simple syntax errors that the expert system shell compilers do not check for, in addition to the more sophisticated syntactic checks discussed in Section 5.1. COVER includes checks for variable type clashes, missing definitions, missing goal statements, and the like in addition to the sophisticated syntactic checks discussed in Section 5.1.

SEAR enforces a style called RIME (R1 Implicit Made Explicit) on the programmers of the XCON (formerly called R1) and XSEL expert systems at DEC. Using SEAR, programmers select a control strategy template for a rule set or "framework" (Heller, 1991). SEAR then automatically generates the control rules for the framework and the programmer fills in the specific content rules for each part. As a result, rule sets follow strict control paradigms and rules themselves have a constrained structure (e.g., rules are only allowed to have one action in their then clauses).

## 5.5 Other Dynamic Testing Tools

We have found four tools that assist in the dynamic testing of expert systems other than Compiler Testing. One, Evidence Flow Graphs, has already been mentioned in Sections 5.1 and 5.2. The other three are Expert/Measure, a commercial tool, RITCaG, a university research tool, and SYCOJET, developed in Europe.

The Evidence Flow Graphs system performs Monte Carlo random test case generation based on all possible input combinations. The knowledge engineer provides meta-knowledge in the form of test specifications that constrain the number and type of random test cases generated and that give the expected results. The tests are generated and run, then the results are compared by the system to the test specifications and any violations are reported to the knowledge engineer.

Expert/Measure was developed by IntelligenceWare Inc. (Parsaye, 1988) as one component of their expert system development suite to provide a tool for evaluating the results of dynamic testing. Tailored primarily for diagnostic expert systems, Expert/Measure measures the distance between the expert system's diagnosis and the correct diagnosis provided by an expert, where a diagnosis is a list of possible causes and corresponding probabilities or confidence factors. Expert/Measure also calculates how similar the ordering of the potential causes is between the system and the expert.

RITCaG (Rule based, Intelligent, Test Case Generator) was developed by the University of Central Florida (Gupta & Biegel, 1990) to generate test cases for rule bases. RITCaG allows the tester to specify whether tests should be generated per rule set, per rule, or per antecedent condition. It generates a Knowledge Unit Template (KUT) for each rule that contains the conditions in the rule, the values of each of the conditions, the rule's rule set, and legal/illegal values for each condition. This information is used to generate test cases, according to the tester's specification, where each test case is both unique and relevant. Both error-containing and error-free test cases may be generated. Then, RITCaG runs the tests and flags any errors.

SYCOJET was developed at the University of Savoie to automatically construct a set of test cases for an expert system based on its precedence graph and a set of criteria for test coverage (Vignollet & Ayel, 1990a, 1990b). The criteria specify the desired coverage based on the elements (facts, rules) of the knowledge base or flow (branches, paths) along the predence graph.

## 5.6 Summary on Automated Tools

As can be seen from the previous sections, most of the effort on automated tool development for expert systems has been for syntactic checking of rule bases. Many of the sophisticated versions of these tools are approaching commercial software grade, and should be available for sale within a few years if industry interest warrants it. The primary job left to be done is to port the tools to handle the rule formats of the most popular expert system shells. Tools that perform syntactic checking of object-oriented or frame-based knowledge bases are still in infancy and will require more research before commercialization is likely.

Semantic checking is the next logical tier of capabilities to add to syntactic checking tools. Some teams have already begun adding this capability to their tools. One major issue must be resolved before semantic checking become commercially viable. The vendors of the popular expert system shells have to add capabilities for the meta-rules or constraints to be expressed within their format, vs. being a separate component (usually in a different rule format) useable only by the automated knowledge base checking tool. Then, porting and commercialization of the semantic checking tools may begin in earnest.

Ideally, these syntactic and semantic checking tools should be made available as part of the development environment for the popular expert system shells from the shell vendors themselves. This would certainly have to be the case for tools that perform checking during knowledge acquisition and/or knowledge refinement and for those that perform intelligent compilation of rule bases. If the industry interest and financial support is there, we should see a next generation of expert system shells in the marketplace with automated V&V tools included in their development environment. Also, a few "mega" expert system V&V tools should be commercially available that work across a variety of rule base formats.

There are still very few tools to support Dynamic Testing of knowledge bases and there is a strong need for research in this area. The primary issue is to determine how, when, and where automated tools could help. Two areas that have been thought of are: 1) proposing structured or random test cases by examining the knowledge base structure, and 2) managing, running, and scoring the results of large regression test suites. There is certainly room for creative research to discover other areas of Dynamic Testing where automated tools could help.

# 6. CATEGORIZATION AND ANALYSIS OF TECHNIQUES AND TOOLS

## 6.1 Components of Expert Systems

The components of an expert system were defined in Section 5.1 of Volume 2. For the purpose of categorizing and analyzing the expert system techniques and tools which were discovered during this activity, we will first discuss the definition of expert system components and then relate the techniques uncovered in this activity to their associated expert system component(s). In this way, the range of expert system verification and validation requirements will be compared to the currently available expert system techniques and conventional techniques, as previously discussed in Activity 1.

As illustrated in Table 6.1-1 there are four principal components of an expert system: Inference Engine, Knowledge Base, Interfaces, and Shell Utilities. Each of these components has distinctive features which are also presented in Table 6.1-1. The knowledge base contains information gleaned from experts in a form which is readily accessible within the expert system and can be periodically updated. The inference engine draws on the knowledge base and other input through interfaces to make recommendations, observations, and/or decisions regarding a specific problem. The interfaces provide data into the expert system from users, data channels, communications, other processes, and/or a data base. Finally, the shell utilities serve the same function as conventional software utilities in that general functions can be called as well as operating system or programming environments accessed.

The interface and shell utility components of expert systems are basically comparable to their counterparts in conventional software. The inference engine and knowledge base constitute the components which differentiate expert systems from conventional software. Generally, the knowledge base is the only component of expert systems that is not written in a conventional programming language (i.e., Fortran, Pascal, C, Cobol, etc). It should be noted that, as we have previously stated in Volume 2, software is defined to be an expert system only if it is <u>not</u> written in procedural language.

Section 5.3 of Volume 2 contains a table which is reproduced as Table 6.1-2 below that relates expert system components to their testing related features. Based on this table, the only expert system sub-components which have features that are not amenable to conventional software verification and validation techniques are the frames, objects, and external databases within knowledge bases and possibly other applications within the interface component. Volume 2 suggested that conventional V&V techniques would be appropriate for the three components typically written in procedural code: Inference engine, Interfaces, and Shell Utilities. It has been suggested that the inference engine is a narrow and fixed function, and should probably undergo certification or qualification procedures, as with other types of highlyreusable conventional programs. Certainly, special test suites or benchmark tests should be developed to test and characterize the engines of commercial shells as well as homegrown ones.

Only the knowledge base component is seen in Volume 2 as requiring new techniques. For those representation choices in Table 6.1-2 involving well-characterized potential defects (e.g., rules) it was suggested that special tools should be developed to test for these defects. Otherwise, other techniques would be required to test the validity of the represented knowledge.

Our present survey of techniques used for expert systems indicates that, in fact, a number of tools have been developed for testing anomalies in the knowledge base component, especially for rules (or rules and frames). On the other hand, while conventional techniques have been used to test the overall expert systems, there has been little emphasis on testing the other procedural components, particularly the inference engine.

Table 6.1-1 Components and general features of knowledge-based systems

| Components | Features | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Reusability[3] | | Typical[4] Program Language | | Commercial[3] Availability | | Complexity[3] | |
| INFERENCE ENGINE | H | | A, C, F, (Lp) | | H | | M | |
| Pattern Matcher | | H | | | | H | | M |
| Conflict Set Handler | | H | | | | H | | M |
| Basic Proof Procedures[5] | | H | | | | H | | M |
| Extended Proof Procedures[6] | | M | | | | L | | H |
| Proof Management[7] | | M | | | | M | | M-H |
| KNOWLEDGE BASE | L | | Lp, P | | L | | M | |
| C Rules | | | | | | | | |
| H Frames | | L | | Lp,P | | L | | L-M |
| O Objects | | L | | Lp,P | | L | | M-H |
| I Facts | | M | | S,C++ | | M | | M-H |
| C External DBs | | L | | Lp,P | | L | | L |
| E In-line Procedures[8] | | M | | n.a. | | L | | M |
| S | | L | | C,Pa | | L | | M |
| INTERFACES | M-H | | A,C | | M | | M | |
| User Interface | | M | | A,C,Lp | | H | | H |
| DBMS | | H | | SQL | | H | | H |
| Data Channels | | M | | A,C | | M | | M-H |
| Communication (ports/networks/channels) | | H | | A,C | | M | | M-H |
| Other Applications | | M | | A,C,Lp | | M | | M |
| SHELL UTILITIES | H | | A,C | | M | | M | |
| Environment Accesses[9] | | H | | A,C,Lp | | M | | M |
| Knowledge Acquisition | | H | | Lp,C | | M | | M |
| Functional Libraries | | M | | C,Lp | | M | | M-H |
| Other Utilities | | H | | C,Lp | | M | | M |

[3] H=High, M=Medium, L=Low
[4] A=Assembler, C=C Language, F=Fortran, Lp=LISP, P=Prolog, Pa=Pascal, S=Smalltalk
[5] Forward and Backward Chaining
[6] Opportunistic, fuzzy-probabilistic-reasoning or constraint-directed reasoning, truth maintenance
[7] Of goal tree, constraints, "truths", breath/depth of search
[8] Including Demons and Functions
[9] To operating system, programming environments, etc

Table 6.1-2 Components and typical testing related features of knowledge-based systems, with testing recommendations

| | FEATURES[10] | | |
|---|---|---|---|
| Components | Typically Written In or Involving Procedural Code | Very Narrow and Fixed Functional Requirements | Potential Defects, Well-Characterized and Amendable to Formal Testing |
| INFERENCE ENGINE | Y | Y | S |
| Pattern Matcher | Y | Y | Y |
| Conflict-Set Handler | Y | Y | Y |
| Proof Procedure | Y | Y | S |
| Proof Management | Y | S | S |
| KNOWLEDGE BASE | N | N | S |
| C Rules | N | N | Y |
| H Frames | N | N | S |
| O Objects | M | N | N |
| I Facts | N | N | Y |
| C External DB | M | N | S |
| E In-line Domons, | Y | N | N |
| S Procedures, Functions | | | |
| INTERFACES | Y | N | N |
| User Interface | Y | N | S |
| DBMS | Y | N | S |
| Data Channels | Y | N | N |
| Communication Ports/ Networks/Channels) | Y | S | S |
| Other Applications | M | N | N |
| SHELL UTILITIES | Y | N | N |
| Environment Accesses | Y | Y | S |
| Knowledge Acquisition | Y | N | N |
| Functional Libraries | Y | N | N |
| Other Utilities | Y | N | N |

[10] Explanation of table entries: Y=Yes, N=No, S=Somewhat, M=Mixed procedural and non-procedural code

## 6.2 Expert System Faults: Anomalies or Invalidities

We have characterized the types of faults that may be found in expert systems by V&V techniques and tools into a 2x2 matrix, which is shown in Figure 6.2-1. The target of the testing, the types of faults which the tests are designed to find, may be subcategorized into Anomalies or Invalidities. The type of testing, and thus the type of error behavior found, may be subcategorized into Static or Dynamic.

Anomalies are structural or syntax flaws, where the form of the software does not meet some guideline, coding standard, style, or logical format. Anomalies may or may not cause the expert system to perform incorrectly, but they are usually related in some way to incorrect performance. For instance, cycles in rule bases are usually incorrect, because they usually lead to infinite recursion where the expert system never returns an answer. However, in some cases cycles may be beneficial, or even required, as long as there are well-defined methods for the expert system to break out of the cycles eventually in all possible situations and avoid infinite recursion. This is why rule base cycles are defined as syntactic errors (vs. semantic) and are flagged with warning (vs. error) messages by expert system syntax checkers.

Invalidities are true content or semantic errors, where the behavior exhibited by the expert system will be incorrect, according to the requirements/specification document or other testing standard. Invalidities in rule bases may result in incorrect or inaccurate answers, misleading explanations, or incorrect confidence weights on answers.
Static errors are those that can be found without executing the expert system. Thus, they can be found by inspecting or performing operations on the expert system software/rule base itself directly. Dynamic errors are those that are found by executing the expert system on test cases or real data. Examples of dynamic errors that usually only can be found during dynamic execution are timing effects, cumulative processing effects (when the error accumulates over many successive executions of the same or multiple processes), and interactions with the operating system services.

As Figure 6.2-1 shows, we have found evidence of expert system V&V techniques and automated tools being applied to all four intersections in the matrix. For ferreting out Static Anomalies, Syntactic Defect Analysis has been applied in both syntax checking and knowledge acquisition/refinement tools. Example tools include (D)EVA and COVER. Also, dependency analysis tools (such as ESPE) have been applied. The Static Anomalies area has seen the most proliferation, by far, of automated tools of any of the four.

For finding Static Invalidities, there are the semantic defect analysis techniques and automated tools. These include semantic checking and knowledge acquisition/refinement tools (such as (D)EVA and SACCO). Applicable techniques include program proving and fault, failure analysis. No automated tools have been built to implement these techniques yet, but there may be potential in converting automated tools used for conventional software V&V, such as the SRI EHDM system (Rushby et al, 1991) for program proving. The challenging issue in program proving for expert systems is how to characterize and account for the behavior of the conflict resolution strategy of the inference engine as it interacts with the rule base. For fault, failure analysis, a promising research area would be the application of Nancy Leveson's fault tree analysis technique to expert systems.

Looking at the categories of expert system V&V techniques for Static Testing (see Figure 4.1-1), we can see how each map over the Static Anomalies and Static Invalidities areas of expert system fault types. As stated previously in Section 4.4, most of the algorithm analysis subclass techniques do not apply to expert systems, except potentially program proving. The control analysis techniques also do not apply to expert systems per se, except when they are embedded in the context of a larger, real-time system. Except for dependency analysis, the Data Analysis techniques also do not apply directly to expert systems. There has been extensive work in applying defect analysis techniques to both Static Anomalies (Syntactic Checking and Knowledge Acquisition/Refinement) and Static Invalidities (Semantic Checking, Knowledge Acquisition/Refinement). More work could be done in applying Fault,

|  |  | Type of Testing |  |
|---|---|---|---|
|  |  | ***Static*** | ***Dynamic*** |
| **TESTING TARGET** | **Anomalies** | • Syntactic Defect Analysis (many automated tools exist)<br>• Dependency Analysis (automated tools exist) | • Structural Testing (automated tools exist)<br>• Stress/Performance Testing<br>• Execution Testing<br>• Intelligent Compilation (automated tools exist) |
|  | **Invalidities** | • Semantic Defect Analysis (automated tools exist)<br>• Program Proving<br>• Fault, Failure Analysis | • Random Testing (automated tools exist)<br>• Functional Testing<br>• Realistic Testing<br>• Competency Testing |
|  | **Both** | • Inspections | • Interface Testing<br>• Error Introduction Testing |

**Figure 6.2-1 Comprehensive Expert System V&V Matrix**

Failure Analysis techniques to Static Invalidities. Lastly, Inspections can be used to find both Static Anomalies and Invalidities.

For finding Dynamic Anomalies, five types of testing focus more on anomalies than invalidities. These are: Compilation Testing, Stress Testing, Performance Testing, Structural Testing, and Execution Testing. All these types of tests are based on non-content related aspects of the expert system and are focused on finding syntax, structural or timing flaws. We have found a few automated rule base compilers that do checking (such as Validator and SEAR) and two automated tools to support structural testing (RITCaG and SYCOJET) of rule bases, but none for the other types of testing.

For finding Dynamic Invalidities, four types of testing focus on content or meaning errors in the expert system. These are: Random Testing, Functional Testing, Realistic Testing, and Competency Testing. We have found one tool to support random statistical testing, the Evidence Flow Graph tool, but none to support the other Dynamic Invalidities testing subclasses.

The remaining two Dynamic Testing subclasses that are not listed in Figure 6.2-1, Interface Testing and Error Introduction Testing, and the remainder of the General/Statistical subclass are focused neither on anomalies or invalidities, but can find either type of error. There are very few automated aids for Dynamic Testing, for either Anomalies or Invalidities. The selection, construction, and management of large sets of test cases or scenarios can be very time-consuming, so we see this as a ripe area for future research.

All the invalidities tools and techniques, both Static and Dynamic, will also find those structural anomalies that result in invalidities of the type for which the technique/tool is designed to look. However, people who have used both anomalies and invalidities techniques and/or tools, or built tools to look for both types of errors, have found that it is better to find and correct the anomalies first. This is because they are usually much easier to find and correct than the invalidities, and correcting them greatly reduces the volume of (potentially spurious) error messages that result from the invalidities testing. The same is true of the relationship between static and dynamic testing, in that it is usually more cost effective to perform static testing first. So, if a developer wishes to apply techniques from all four areas to his expert system, the best order, over the development lifecycle is:

1) Static Anomalies,

2) Static Invalidities,

3) Dynamic Anomalies, then

4) Dynamic Invalidities.

There is a lot of support in both diversity and number of automated tools for finding static anomalies, so automation should certainly be investigated for this step. Static invalidities, semantic defect analysis tools and the Program Proving and Fault, Failure Analysis techniques are in their infancy. Thus, at this time, the expert system developer may need to rely on Inspections, such as by an expert panel. Automated tools for either type of Dynamic Testing are also in their infancy, so current developers should select one or more testing techniques from each of the dynamic-anomalies and dynamic-invalidities boxes of the matrix. A mix of techniques vs. focusing on any one is best. A good example is the IDA (see Appendix A trip report) structural selection of a regression test suite of performance, random, and realistic test cases combined with User Interface Testing.

# 7. CONCLUSIONS

This report documents the results of Activity 2 of the Expert System Verification and Validation Guideline Project. The work described in this volume consisted of an exhaustive, comprehensive, and multi-faceted investigation of the current state-of-the-art of expert system V&V techniques and tools. The process involved a combination of telephone contacts, data base evaluation, literature collection, and onsite face-to-face meetings.

Activity 2 has provided a roadmap and established the groundwork for many of the future activities of this project. In Activity 4, the nuclear power plant verification methodology to be developed will draw upon the experience documented in this report to streamline the selection process for appropriate techniques and tools. Testing this verification method in Activity 5 will draw upon the testing methods which were discovered during this data collection effort to place our resources on the most fruitful testing schemes. In Activity 6, the knowledge based certification method will be developed while drawing on how other organizations both within and outside of the nuclear industry have accomplished this aspect of V&V as documented in this report. A number of different methods to develop validation scenarios were uncovered during the course of Activity 2. This information will assist in performing Activity 8, which is to develop and evaluate validation scenarios.

# Appendix A: Bibliography

Aldridge, J., *A Position Paper on Validation of Knowledge-based Systems, Validation and Testing of Knowledge-Based Systems Workshop Proceedings*, AAAI-88, The MIT Press, Cambridge, Massachusetts, August 1988.

American Nuclear Society Standards Committee and the Nuclear Power Engineering Committee of the Institute of Electrical and Electronic Engineers, Inc. Power Engineering Society, *American National Standard Applications Criteria for Programmable Digital Computer Systems in Safety Systems of Nuclear Power Generating Stations*, ANSI/IEEE-ANS-7.4.3.2, IEEE Service Center, Piscataway, New Jersey, 08554, 1982.

Andert, E.P., and C. Frasher, *Verifiable, Autonomous Satellite Control System*, IEEE Aerospace Applications Conference Digest, IEEE Service Center, Piscataway, New Jersey, 08554, 1989, February 1989.

Andriole, S.J., and G.W. Hopple, *Defense Applications of Artificial Intelligence*, (Lexington Books, Lexington Massachusets, 1988), Chapter 6, pp. 111-138.

Majumdar, C., D. Majumdar, and J.I. Sackett, Eds., *Artificial Intelligence and Other Innovative Computer Applications in the Nuclear Industry*, Plenum Press, New York, New York, 1988.

Ayel, M., and L. Vignollet, *Automatic Building of Test-Samples for Validating Knowledge Based Systems*, Verification, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, IJCAI-89, Morgan Kaufmann Publishers, Inc. Palo Alto, California 94303, August 1989.

Bachant, J., *Validating and Testing XCON*, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts, 02142, August 1988.

Bahill, A.T., *Verifying and Validation Personal Computer-Based Expert Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

Bahill, A.T., M. Jafar, and R.F. Moller, *Tools for Extracting Knowledge and Validating Expert Systems*, Proceedings of the 1987 IEEE International Conference on Systems, Man and Cybernetics, IEEE Service Center, Piscataway, New Jersey, 08854, October 1987.

Barker, V.E., and D.E. O'Connor, *Expert Systems for Configuration at Digital: XCON and Beyond*, Communications of the ACM. Vol. 32, No. 3, March 1989.

Barrett, B.W., *A Software Quality Specification Methodology for Knowledge-Based Systems, Knowledge Based Systems Verification, Validation and Testing*, Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts, 02142, July 1990.

Barrett, B.W., *A Hypothetical Application of the Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) to Knowledge-Based Systems (KBSs)*, Expert Systems with Applications, Special Issue: Verification and Validation of Knowledge-Based Systems, Vol. 1, No. 3, pp. 281-289, 1990.

Barrett, B.W, *Software Reliability Models: Can They be Applied to Expert Systems? Verification, Validation and Testing of Knowledge-Based Systems?*, Workshop Proceedings, IJCAI-89, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 1989.

Bartschat, S.J., *Test Strategies for Expert System Applications*, Presented at the IBM Testing Interdivisional Technical Liaison (ITL) Meeting, August 1990.

Batarekh, A., A.D. Preece, A. Bennett, and P. Grogono, *Specifying an Expert System*, Expert Systems with Applications, 2(3), 1991.

Baum, L.S., D.B. Shema, J.H. Boose, and J.M. Bradshaw, *Acquiring and Verifying Control Knowledge for a Blackboard System*, Verification, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, IJCAI-89, Morgan Kaufmann Publishers, Inc., Palo Alto, California, 94303, August 1989.

Becker, L.A., P.G. Green, and J. Bhatnager, *Evidence Flow Graph Methods for Validation and Verification of Expert Systems*, Worcester Polytechnic Institute, Worcester, Massachusetts. Contract NAG1-809, July 1989.

Becker, L., J. Duckworth, P.E. Green, B. Michalson, D. Gosselin, K. Nainani, and A. Pease, *Translating Expert System Rules into ADA Code with Validation and Verification, Final Report August 1990*, Worcester Polytechnic Institute, Worcester, Massachusetts., 01609.

Becker, L.A., P.E. Green, R.J. Duckworth, J. Bhatnager, and A. Pease, *Evidence Flow Graphs for VV&T. Verification, Validation and Testing of Knowledge-Based Systems*, Workshop Proceedings, IJCAI-89, Morgan Kaufmann Publishers, Inc., Palo Alto, California, 94303, August 1989.

Becker, L., J. Duckworth, P. E. Green, and A. Khanna, *Translating Expert System Rules into ADA Code with Validation and Verification [KRAM 2-1], Final Report*, March 1991. Worcester Polytechnic Institute, Worcester, Massachusetts, 01609.

Bellman, K.L, and D.O. Walter, *Analyzing and Correcting Knowledge-based Systems Requires Explicit Models*, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, The MIT Press, Cambridge, Massachusetts, 02142, AAAI-88, August 1988.

Bellman, K.L., *Testing, Verifying and Validating Autonomous Systems: Big Plans for Expert Systems and Big Challenges for Us*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts, 02142, July 1990.

Bellman, K.L., *Testing and Evaluating Knowledge-Based Systems: There's a Lot We Can Do Now*, AAAI, The MIT Press, Cambridge, Massachusetts, 02142, January 1989.

Bellman, K.L., *The Modeling Issues Inherent in Testing and Evaluating Knowledge-Based Systems*, Expert Systems with Applications, Special Issue: Verification and Validation of Knowledge-Based Systems, Vol. 1, No. 3, pp. 199-215, 1990.

Beltracchi, L., *Expert Systems and Nuclear Safety*, Proceedings of the American Nuclear Society 1990 Winter Meeting, The American Nuclear Society, La Grange Park, Illinois, November 11, 1990.

Benbasat, I., and J.S. Dhaliwal, *A Framework for the Validation of Knowledge Acquisition.* Knowledge Acquisition, Vol. 1, No. 2, pp. 215-233, June 1989.

Bhatnagar, R., D.W. Miller, B.K. Hajek, and J.E. Stasenko, *An Integrated Operator Advisor System for Plant Monitoring, Procedure Management, and Diagnosis*, Nuclear Technology, Vol. 89, March 1990.

Bochsler, D.C., and M.A. Goodwin, *Software Engineering Approach to Expert System Design and Verification*, Second Conference on Artificial Intelligence for Space Applications, pp 47-60, NASA, Johnson Space Center, Houston, Texas, August 1988.

Boehm, B.W., *A Spiral Model of Software Development and Enhancement*, IEEE Computer, pp. 61-72, May 1988.

Bond, D., D. Lane, and P. Shikli, *Knowledge Management in Expert Systems*, Technical Report, SAIC COMSYSTEMS Division, San Diego, California, 92121, May 1988.

Bonissone, P.P., and N.C. Wood, *Plausible Reasoning in Dynamic Classification Problems*, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts, 02142, August 1988.

Boose, J.H., *Personal Construct Theory and The Transfer of Human Expertise*, AAAI-84, Proceedings of the National Conference on Artificial Intelligence, Vol. 1, The MIT Press, Cambridge, Massachusetts, 02142, pp. 27-33, August 1984.

Botten, N., A. Kusiak, and T. Raz, *Knowledge Bases: Integration, Verification, and Partitioning*, European Journal of Operational Research, Number 42, pp 111-128, 1989.

Boyer, R., and J.S. Moore, *Proof Checking the RSA Public Key Encryption Algorithm*, Readings in Artificial Intelligence and Software Engineering, ed., Charles Rich, and Richard C. Waters, pp. 87-96, 1986, Morgan Kaufmann Publishers, Los Altos, California.

Burke, R.A., and R.W. McNenny, *VV&T Advantages Provided by the ID3 Algorithm*, Technical Report. Available from NASA, Johnson Space Center, Houston, Texas.

Burris, L.B., C. Chang, J.B. Combs, R. McBeth, R. Stachowitz, and T. Stock, *The Expert Systems Validation Associate: An Overview*, Technical Report, The Lockheed Artificial Intelligence Center, Palo Alto, California, 94304.

Cain, D., E. Choi, M. Divakarani, V. Longo, T. Wilson, and B. Braithwaite, *EPRI GEMS: Expert Systems for Technology Transfer*, Technical Report, The Electrical Poweer Research Institute, Palo Alto, California 94303, 1991.

Callahan, P.H., *Expert Systems for AT&T Switched Network Maintenance*, AT&T Technical Journal, Vol. 67, Issue 1, pp. 93-103, AT&T Technical Journal, Short Hills, New Jersey, 07078, January 1988.

Castore, G., S. Kalpin, and B. Schrag, *Knowledge-Based Control Systems*, Technical Report, Honeywell Inc., Minneapolis, Minnesota.May 1987.

Castore, G., *A Formal Approach to Validation and Verification for Knowledge-Based Control Systems,* Proceedings of First Annual Workshop on Space Operations Automation (SOAR '87), NASA, Johnson Space Center, Houston, Texas, pp. 197-202, August 1987.

Chang, C.L., R.A. Stachowitz, and J.B. Combs, *Validation of Nonmonotonic Knowledge-Based Systems,* Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts, 02142, July 1990.

Chang, W.C., and J.F. Cheng, *The Utility Experience of Implementing the Emergency Operating Procedure Tracking System,* Proceedings of Applications for the Electric Power Industry Conference, Electric Power Research Institute, Palo Alto, California, 94303, June 1990.

Chang, C.L., J.B. Combs, and R.A. Stachowitz, *A Report on the Expert Systems Validation Associate (EVA),* Expert Systems with Applications, Special Issue: Verification and Validation of Knowledge-Based Systems, Vol. 1, No. 3, pp. 217-230, 1990.

Chang, C.L., and R.A. Stachowitz, *Testing Expert Systems,* Second Annual Workshop on Space Operations Automation and Robotics (SOAR '88), NASA, Johnson Space Center, Houston, Texas, July 1988, pp. 131-135.

Chang, C.L., R.A. Stachowitz, and J.B. Combs, *Testing Integrated Knowledge-based Systems,* IEEE International Workshop on Tools for Artificial Intelligence: Architectures, Languages and Algorithms, IEEE Service Center, Piscataway, New Jersey, 08854, pp. 12-18, October 1989.

Chee, C., *Comments on Space Station Freedom Program SSFP Software Life-Cycle Management Standards for Flight and Ground Software,* Private Correspondence to Dr. Michael Freeman, June 1990.

Cheng, J.F., R. Chiang, C.C. Yao, A.J. Spurgin, D.D. Orbis, B.K.H. Sun, D. Cain, and C. Christeinsen, *Evaluation of an Emergency Operating Procedure Tracking Expert System by Control Room Operators,* Proceedings of Applications for the Electric Power Industry Conference, Electric Power Research Institute, Palo Alto, California, 94303, June 1990.

Cohen, P.R., D.M. Hart, and J.K. Devadoss, *Models and Experiments to Probe the Factors that Affect Plan Completion Times for Multiple Fires in Phoenix,* Experimental Knowledge Systems Laboratory, Department of Computer and Information Science, University of Massachusetts, submitted to IJCAI-91, Commission Electrotechnique Internationale Norme De La Cei, *Software for Computers in the Safety Systems of Nuclear Power Stations,* International Electrotechnical Commission IEC Standard Publication 880, First Edition, Bureau Central de la Commission Electrotechnique Internationale, 3 rue de Varemoe, Geneve Suisse, 1986.

Cohen, P.R., and A.E. Howe, *The Invisible Hand: How Evaluation Guides AI Research,* Validation and Testing of Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts, 02142, August 1988.

Computer Sciences Corporation, *Expert System Development Methodology Standard, Technical Report, Revision 1,* Computer Sciences Corporation, Chantilly, Virginia, August 1989.

Computer Sciences Corporation, *Expert System Development Methodology Reference Manual, Technical Report, Revision 1,* Computer Sciences Corporation, Chantilly, Virginia, August 1989.

Computer Sciences Corporation, *Expert System Development Methodology User Guide. Technical Report, Revision 1*, Computer Sciences Corporation, Chantilly, Virginia, August 1989.

Computer Sciences Corporation, *Expert System Development Methodology Framework for Evaluation of ESDM, Preliminary Report, Working Materials*, Computer Sciences Corporation, Chantilly, Virginia, March 1989.

Conrath, D.W., and R.S. Sharma, *Evaluating Expert Systems using A Multiple-Criteria, Multiple-Stakeholder Approach*, University of Waterloo, Ontario, Canada.1991.

Constantine, M.M., and J.W. Ulvila, *Knowledge-Based Systems in the Army: The State of the Practice and Lessons Learned, with Implications for Testing*, Verification, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, IJCAI-89, Morgan Kaufmann Publishers, Inc., Palo Alto, California, 94303, August 1989.

Constantine, M.M., and J.W. Ulvila, *Testing Knowledge-Based Systems: The State of the Practice and Suggestions for Improvement*, Expert Systems with Applications, Special Issue: Verification and Validation of Knowledge-Based Systems, Vol. 1, No. 3. pp. 237-248, (1990).

Cragun, B.J., and H.J. Steudel, *A Decision-table-based Processor for Checking Completeness and Consistency in Rule-based Expert Systems*, International Journal of Man-Machine Studies, Vol. 26, No. 5, pp. 633-648, May 1987.

Crow, J., and J. Rushby, *Model-Based Reconfiguration: Integrating Diagnosis and Recovery*, SRI International, Contract No. NAS1-18969, Task 2, SRI Project 8200-120, Final Report, SRI International, Menlo Park, California, 94025, February 1991.

Crow, J., and J. Rushby, *Model-Based Reconfiguration: Toward an Integration with Diagnosis*, Presented at the AAAI-91 Conference, Anaheim, California, July 14-19, SRI International, Menlo Park, California, 94025, 1991.

Cuddigan, J.E., J. Norris, S.A. Ryan, and S. Evans, *Validating the Knowledge in a Computer-based Consultant for Nursing Care*, Proceedings of the Eleventh Annual Symposium on Computer Applications in Medical Care, pp. 74-78, SCAMC, Inc., Washington, DC 20004, November 1987.

Culbert, C., G. Riley, and R.T. Savely, *Expert System Development Methodology That Supports Verification and Validation*, ISA Transactions Vol 28, No. 1, pp. 15-18, 1989.

Culbert, C., G. Riley, and R.T. Savely, *Expert System Development Methodology Which Supports Verification and Validation*, Proceedings of the ISA/88 International Conference and Exhibit, pp. 1443-1447, NASA Johnson Space Center, Houston, Texas, October 1988.

Culbert, C., and R.T. Savely, *Expert System Verification and Validation*, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts, 02142, August 1988.

Culbert, C., and R.T. Savely, *CRSV and AAMP: Ongoing work at National Aeronautical and Space Agency/Johnson Space Center in KBS Verification and Validation*, Submittal to Verification, Validation and Testing of Knowledge-Based Systems Workshop, NASA Johnson Space Center, Houston, Texas, IJCAI-89, August 1989.

Culbert, C., G. Riley, and R.T. Savely, *Approaches to the Verification of Rule-Based Expert Systems*, SOAR '87 First Annual Workshop on Space Operations Automation and Robotics, SCAMC, Inc., Washington, DC 20004,August 1987.

Culbert, C., G. Riley, and R.T. Savely, *An Expert System Development Methodology Which Supports Verification and Validation*, 4th IEEE Conference on Artificial Intelligence Applications, IEEE Service Center, Piscataway, New Jersey, 08854, September 1987.

Culbert, C., G. Riley, and R.T. Savely, *Verification Issues for Rule-Based Expert Systems*, Third Conference on Artificial Intelligence for Space Applications, Part 1, NASA Johnson Space Center, Houston, Texas, November 1987.

Cullyer, W.J., *Expert Systems and Certification*, IEEE Colloquium on 'Expert Systems Liability,' Digest No. 115, London, United Kingdom, October 23, 1989, March 1-3, 1989.

Davis, R., *Interactive Transfer of Expertise: Acquisition of New Inference Rules. Readings in Artificial Intelligence*, Ed. B.L. Webber and N.J. Nilsson, Tioga Publishing Co., Palo Alto, California, pp. 410-428, 1981.

Decision Science Consortium, Inc., *Testing and Evaluating C3I Systems That Employ AI, Vol 1: Handbook for Testing Expert Systems*, Final Report, Contract No. DAEA18-88-C-0028, Decision Science Consortium, McLean, VA 22102, January 1991.

Desimone, R., and J. Rininger, *Expert System Validation and Verification*, SRI International, Contract DAAB07-86-D-A035, SRI Project 3002, Final Report, SRI International, Menlo Park, California 94025, August 1990.

Doyle, R.J., *Constructing and Refining Causal Explanations from an Inconsistent Domain Theory*, Proceedings of the AAAI-86 Fifth National Conference on Artificial Intelligence, Philadelphia, Pennsylvania, August 11-15, 1986, Vol. 1, Science, pp. 538-544, The MIT Press, Cambridge, Massachusetts, 02142.

Duhamel, A., R. Beuscart, J. Demongeot, Y. Mouton, and the SES Group, *SES (Septicemia expert system): Knowledge Validation From Data Analysis*, Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, IEEE Service Center, Piscataway, New Jersey, 08854, November 1988.

Duke, E.L., *Application of Flight Systems Methodologies to the Validation of Knowledge-Based Systems*, 2nd Annual Workshop on Space Operations Automation and Robotics (SOAR 1988), NASA Johnson Space Center, Houston, Texas, pp. 107-121.

Edwards, R.M., D.W. Ruhl, E.H. Klevans, and G.E. Robinson, *Development and Testing of a Diagnostic System for Intelligent Distributed Control at EBR-II*, Presented at Fast Reactor Safety Meeting, Snowbird, Utah, The Pennsylvania State University, University Park, Pennsylvania 16802, August 1990.

Electric Power Research Institute, *Reactor Emergency Action Level Monitor, Vol. 2: REALM User's Reference Guide, EPRI NP-5719, Vol. 2*, The Electric Power Research Institute, Palo Alto, California 94303, September 1988.

Electric Power Research Institute, *Proceedings: 1987 Conference on Expert System Applications in Power Plants*, Proceedings EPRI 2923-1, The Electric Power Research Institute, Palo Alto, California 94303, December 1988.

Electric Power Research Institute, *Reactor Emergency Action Level Monitor, Vol. 1: REALM Technical Report, EPRI NP-5719, Vol. 1*, The Electric Power Research Institute, Palo Alto, California 94303, September 1988.

Electric Power Research Institute, *Verification and Validation of Expert Systems for Nuclear Power Plant Applications, Final Report, NP-5978*, The Electric Power Research Institute, Palo Alto, California 94303, August 1988.

Electric Power Research Institute, *Conference on Expert Systems Applications for the Electric Power Industry Conference*, Orlando, Florida, June 5-8, The Electric Power Research Institute, Palo Alto, California 94303, 1989.

Electric Power Research Institute, *The EPRI Knowledge Acquisition Workshop Handbook. EPRI NP-6240, Final Report*, The Electric Power Research Institute, Palo Alto, California 94303, February, 1989.

Electric Power Research Institute, *Electric Power Research Institute Artificial Intelligence/Expert Systems Research and Development*, Ed. B. Sun, The Electric Power Research Institute, Palo Alto, California 94303, October 1987.

Electric Power Research Institute *Emergency Operating Procedures Tracking System. Interim Report NP-5250M*, Vol. 1, The Electric Power Research Institute, Palo Alto, California 94303, June 1987.

Electric Power Research Institute, *Survey and Assessment of Conventional Software Verification and Validation Techniques. Final Report*, TR-102106, Research Project 3093-01, The Electric Power Research Institute, Palo Alto, California 94303, February 1993.

Electric Power Research Institute, *Emergency Operating Procedures Tracking System. Interim Report NP-5250SP*, The Electric Power Research Institute, Palo Alto, California 94303, June 1987.

Electric Power Research Institute, *Approaches to the Verification and Validation of Expert Systems for Nuclear Power Plants, NP-5236, Final Report*, The Electric Power Research Institute, Palo Alto, California 94303, July 1987.

Electric Power Research Institute, *The BWR Emergency Operating Procedures Trackings System (EOPTS), Evaluation by Control-Room Operating Crews. Final Report NP-68-46*, The Electric Power Research Institute, Palo Alto, California 94303, May 1990.

Electric Power Research Institute, *Reactor Emergency Action Level Monitor Expert-System Prototype: Independent Review, EPRI NP-5719, Vol. 3*, The Electric Power Research Institute, Palo Alto, California 94303, April 1988.

Enand, R., G.S. Kahn, and R.A. Mills, *A Methodology For Validating Large Knowledge Bases*, International Journal of Man-Machine Studies, Vol. 33, 1990, pp. 361-371.

Eshelman, L., and J. McDermott, *MOLE: A Knowledge Acquisition Tool That Uses Its Head*, Proceedings of the AAAI-86 Fifth National Conference on Artificial Intelligence. Philadelphia, Pennsylvania, August 11-15, 1986, Vol. 2, Engineering, The MIT Press, Cambridge, Massachusetts, 02142, pp. 950-955.

Fieschi, M., *Towards Validation of Expert Systems as Medical Decision Aids*, International Journal of Bio-Medical Computing, Vol. 26, No. 1-2, pp. 93-108, July 1990.

Finin, T.W., *Interactive Classification: A Technique for Acquiring and Maintaining Knowledge Bases*, Proceedings of the Institute of Electrical and Electronic Engineers, Inc., Vol. 74, No. 10, pp. 1414-1421, IEEE Service Center, Piscataway, New Jersey, 08854, October 1986.

Finlay, P.N., G.J. Forsey, and J.M. Wilson, *The Validation of Expert Systems-Contrasts With Traditional Methods*, Journal of the Operational Research Society, Vol. 39, No. 10, pp. 933-938, October 1988.

Franklin, W.R., R. Bansal, E. Gilbert, and A. Prakash, *Sensitivity Analysis and Version Control of Expert Systems*, Technical Report, The MIT Press, Cambridge, Massachusetts, 02142, February 1989.

Franklin, W.R., R. Bansal, and E. Gilbert, *Sensitivity Analysis of Expert Systems. Validation and Testing of Knowledge-Based Systems Workshop*, Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts, 02142, August 1988.

Franklin, W.R., R. Bansal, E. Gilbert, and G. Schroff, *Debugging and Tracing Expert Systems*, Proceedings of the Twenty First Annual Hawaii International Conference on System Sciences, Vol. 3, Kona, Hawaii, pp. 159-167, IEEE Service Center, Piscataway, New Jersey, 08854, January 1988.

Friedman, J., L.A. Miller, and A. Peterson, *Concept Paper: An IV&V Approach to Validate Requirements for Expert Systems*, Technical Report, SAIC COMSYSTEMS Division, San Diego, California, May 2, 1988.

Gaschnig, J., P. Klahr, H. Pople, E. Shortliffe, and A. Terry, *Evaluation of Expert Systems: Issues and Case Studies. Building Expert Systems*, F. Hayes-Roth, D.A. Waterman, and D.B. Lenat (Eds), Don Mills, Addison-Wesley, Ontario, Canada, Chapter 8, pp. 241-280, 1983.

Gearhart, L.M., *Managing the Development and Deployment of Expert Systems*, Proceedings of the IEEE 1989 National Aerospace and Electronics Conference, NAECON, IEEE Service Center, Piscataway, New Jersey, 08854, May 1989.

Geissman, J.R., *Verification and Validation for Expert Systems: A Practical Methodology*, Proceedings from the 4th Annual Presentation of The Artificial Intelligence and Advanced Computer Technology Conference & Exhibition, Abacus Programming Corporation, Van Nuys, California.May 1988.

Geissman, J.R., and R.D. Schultz, *Verification & Validation of Expert Systems*, AI Expert, pp. 26-33, February 1988.

Gilstrap, L., *Expert System Development Methodology Case Study: Intelligent Tutoring System*, Prepared for the National Aeronautics and Space Administration, Goddard Space Flight Center, Greenbelt, Maryland, November 1990.

Gilstrap, L., *Management of Expert Systems Development*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts, 02142, August 1990.

Gilstrap, L., *Validation and Verification of Expert Systems*, 1991 Goddard Conference on Space Applications of Artificial Intelligence, May 14-15, pp. 241-248, 1991.

Gilstrap, L., *Evaluation of a Proposed Expert System Development Methodology: Two Case Studies*, Telematics and Informatics, Vol. 7, No. 3/4, pp. 467-477, The MIT Press, Cambridge, Massachusetts, 02142, 1990.

Ginsberg, A., *Theory Reduction, Theory Revision, and Retranslation*, Proceedings Eighth National Conference on Artificial Intelligence - AAAI-90, Vol. 2, pp. 777-782, The MIT Press, Cambridge, Massachusetts, 02142, July 1990.

Ginsberg, A., *Theory Revision via Prior Operationalization. Proceedings AAAI-88 The Seventh National Conference on Artificial Intelligence*, Vol. 2, pp. 585-589, The MIT Press, Cambridge, Massachusetts, 02142, August 1988.

Ginsberg, A., S. Weiss, and P. Politakis, *Seek2: A Generalized Approach to Automatic Knowledge Base Refinement*, IJCAI-85, Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Vol. 1, pp. 367-374, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 1985.

Ginsberg, A., *Knowledge-Base Reduction: A New Approach to Checking Knowledge Bases for Inconsistency & Redundancy*, Proceedings AAAI-88 The Seventh National Conference on Artificial Intelligence, Vol. 2, pp. 585-589, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Ginsberg, A., *A Metalinguistic Approach to the Construction of Knowledge Base Refinement Systems*, Proceedings AAAI-86 Fifth National Conference on Artificial Intelligence, Vol. 1, pp. 436-441, The MIT Press, Cambridge, Massachusetts 02142, August 1986.

Goodwin, M.A., and C.C. Robertson, *Expert System Verification Concerns in an Operations Environment*, First Annual Workshop on Space Operations Automation and Robotics (SOAR '87), Houston, Texas, NASA Johnson Space Center, Houston, Texas, August 1987.

Goodwin, M.A., and C.C. Robertson, *A Systemic View of Validating and Testing Knowledge-Based Systems*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88, NASA Johnson Space Center, Houston, Texas, August 1988.

Goodwin, M.A., and L.R. Fussell, *Use of Electronic Knowledge Repositories in VV&T of Knowledge-Based Systems*, Submittal to AAAI-88 Workshop on Validation and Testing of Knowledge-Based Systems, Rockwell Space Operations Company, Houston, Texas 77058.

Gordin, D., D. Foxvog, J.R. Rowland, P. Surko, and G.T. Vesonder, *OKIES: A Troubleshooter in the Factory*, AT&T Bell Laboratories, Warren, New Jersey 07060.

Gowens, Dr. J.W., *Expert System Evaluation Methodology (ASQBG-A-89-034)*, AIRMICS, Atlanta, Georgia, July 1989.

Green, C.J.R., *On the Use of Requirements in Development of Knowledge-Based Systems*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Green, C.J.R., and M.M. Keyes, *Verification and Validation of Expert Systems*, WESTEX-87: Proceedings-Western Conference on Expert Systems, Anaheim, California, IEEE Service Center, Piscataway, New Jersey, 08854, June 1987.

Green, C.J.R., *Evolutionary Approach to Verification and Validation of Expert Systems*, Proceedings - 1987 Fall Joint Computer Conference, Exploring Technology: Today and Tomorrow, pp. 760, IEEE Service Center, Piscataway, New Jersey, 08854, October 1987.

Griesmer, J.H., *Validating and Testing the FAME Expert System*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Grogono, P., A. Batarekh, A. Preece, R. Shinghal, and C. Suen, *Expert System Evaluation Techniques: a Selected Bibliography*, Expert Systems, 1991.

Groundwater, E.H., *Verification and Validation Plan for the Water Chemistry Expert Monitoring System (WCEMS)*, Technical Report, Science Applications International Corporation, 1710 Goodridge Drive, McLean, Virginia 22102, August 1989.

Gupta, U.G., and J. Biegel, *RIICaG: A Rule-Based Intelligent Test Case Generator*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Hajek, B.K., C.R. Hardy, D.W. Miller, and R. Bhatnagar, *A V&V Program for a Real Time Operator Advisor Expert System*, Proceedings of Applications for the Electric Power Industry Conference, The Electric Power Research Institute, Palo Alto California 94303, June 1990.

Hall, D.L., and D.T. Heinze. *The Use of Simulation Techniques for Expert System Test and Evaluation*, ISA Transactions, Vol. 28, No. 1, pp. 19-22, 1989.

Hall, L., M. Friedman, and A. Kandel, *On the Validation and Testing of Fuzzy Expert Systems*, IEEE Transactions on Systems, Man and Cybernetics, Vol. 18, No. 6, pp. 1023-1027, November/December 1988.

Hamilton, D., K. Kelly, and C. Culbert, *KBS V&V - State-of-the-Practice and Implications for V&V Standards*, IBM, Houston, Texas, 1991.

Harrison, P.R., P.A. Ratcliffe, D.S. Kimes, and D. Fairhead, *An Integrated Prototyping and Validation Model for Knowledge-Based System Design*, Knowledge Based Systems Verification, Validation and Testing of Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Harrison, P.R., P.A. Ratcliffe, *Towards Standards for the Validation of Expert Systems. Expert Systems With Applications*, Vol. 2, pp. 251-258, 1991.

Harrison, P.R., P.A. Ratcliffe, CDR J.M. Owens, USN, and CDR S.D. Harris, USN, *The Validation of Embedded Knowledge-Based Systems*, Proceedings of the Twenty-second Annual Modeling and Simulation Conference, May 1991.

Harrison, P.R., *Testing and Evaluation of Knowledge-based Systems, in Structuring Expert Systems: Domain, Design, and Development*, J. Liebowitz and D.A. DeSalvo, Eds., Yourdin Press, 1989, Chapter 11, pp. 303-329.

Harrison, P., J.M. Owens, and D.K. McBride, *Validation of Expert Systems that Implement Simulations*, Presented at the Military Operations Research Society, Mini-Symposium on Simulation Validation, October 1990.

Hassberger, J.A., and J.C. Lee, *A Simulation-Based Expert System for Nuclear Power Plant Diagnostics*, Nuclear Science and Engineering, Volume 102, pp. 153-171, 1989.

Heller, M., *AI in Practice, Once it Works, Does it Still Qualify as AI?* BYTE Magazine, January 1991.

Hofmeister, A.M., *Formative Evaluation in the Development and Validation of Expert Systems in Education,* Computer Intelligence, Vol. 2, No. 2. May 1986, pp. 65-67.

Howe, A.E., and P.R. Cohen, *Failure Recovery: A Model and Experiments*, Experimental Knowledge Systems Laboratory, Department of Computer Science, University of Massachusetts. Accepted for AAAI-91, The MIT Press, Cambridge, Massachusetts 02142,

Hu, S.D., *Expert Systems for Software Engineers and Managers*, Chapman and Hall, New York, New York, 1987.

Hughes, G., *Design and Validation of Advanced Operator Support Systems for a Role in Plant Safety*, Research Report, Central Electricity Generating Board, Berkeley, United Kingdom, June 1989.

ICF Information Technology, Inc., *Testing and Evaluating C3I Systems That Employ AI, Vol 3: A Guide to Developing Small Expert Systems*, ICF Information Technology, Inc., San Jose, CA, February, 1991.

Institution of Electrical Engineers, *Computing and Control Division Colloquium organized by Professional Group C4 (Artificial intelligence) on 'Testing Expert Systems*, Savoy Place, London, Institution for Electrical Engineers, London, United Kingdom, December 3, 1987.

International Business Machines (IBM), *Expert System Verification and Validation Survey RICIS Contract #069 Delivery 4 - Final Report*, IBM, Houston, Texas 77058, September 14, 1990.

International Business Machines (IBM), *Expert System Verification and Validation Study RICIS Contract #069, Phase 2 - Requirements Identification, Delivery 2 - current Requirements Applicability*, IBM, Houston, Texas 77058, January 13, 1990.

Johnson, S.C., *Validation of Highly Reliable, Real-Time Knowledge-Based Systems*, Second Annual Workshop on Space Operations Automation and Robotics (SOAR 1988), pp. 123-129, NASA Johnson Space Center, Houston, Texas, November 1988.

Juang, J.Y., J.M. Lin, and E. Freeman, *Characterizing Real-Time Capabilities of a Logic Program via Deductive Analysis*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Kahn, G., S. Nowlan, and J. McDermott, *MORE: An Intelligent Knowledge Acquisition Tool*, IJCAI '85, Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Los Angeles, California, pp. 581-584, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 18-23, 1985.

Kaisler, S.H., *Expert System Metrics*, Proceedings of the 1986 IEEE International Conference on Systems, Man, and Cybernetics, Vol. 1, pp. 114-120, IEEE Service Center, Piscataway, New Jersey, 08854, October 1986.

Kang, Y., and A.T. Bahill, *A Tool for Detecting Expert-System Errors*, AI Expert, pp. 46-51, February 1990.

Kearsley, G., *Validation of an Expert System: The CBT Analyst*, Journal of Computer-Based Instruction. Vol. 15, No. 2, pp. 61-64, Spring 1988.

Kiper, J.D., *Structural Testing of Rule-based Expert Systems*, Verification, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, IJCAI-89, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 1989.

Kirk, D., and J. Naser., *A Verification and Validation Methodology for Systems in Nuclear Power Applications*, Proceedings of Applications for the Electric Power Industry Conference, Electric Power Research Institute, Palo Alto, California, June 1990.

Kitamura, M.M. Takahashi, T. Washio, and K. Sugiyama, *Synthesis of Heuristic Knowledgebase for Supporting Development of Goal-Oriented Reactor Noise Analysis Programs*, Progress in Nuclear Energy, Vol. 21, Pergamon Press, Great Britain, pp. 213-221.

Klein, G.A , R. Caldwood, and D. MacGregor, *Critical Decision Method for Eliciting Knowledge*, IEEE Transactions on Systems, Man and Cybernetics, V19, No. 3, pp. 462-472, May/June 1989.

Klein, G.A., and J.A. King, *A Test for the Performance of Knowledge-Base Systems: AIQ*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Klein, G.A., *AIQ: A Test for the Performance of Expert Systems*, AOG/AAAI 87 Joint Conference, Dayton, Ohio, The MIT Press, Cambridge, Massachusetts 02142, October 1987.

Klinker, G., C. Boyd, S. Genetet, and J. McDermott, *A KNACK for Knowledge Acquisition*, AAAI 1987, Proceedings of the Sixth National Conference on Artificial Intelligence, Vol. 2, Seattle, Washington, July 13-17, pp. 488-493, The MIT Press, Cambridge, Massachusetts 02142, July 1987.

Kobialka, H.U., *Configuration Editing, Generation & Test within Working Contexts. Sigsoft Software Engineering Notes*, Vol 15, No. 6, pp. 173-182, December 1990.

Krishnamurthy, C., S. Padalkar, J. Sztipanovits, and B.R. Purvis, *Methodology for Testing and Validating Knowledge Bases*, Third Conference on Artificial Intelligence for Space Applications, Part 1. pp. 21-32, NASA Johnson Space Center, Houston, Texas, November 1987.

Kusiak, A., *Identification of Anomalies in Rule Bases*, Working Paper No. 89-12, June 1989. Available for purchase from The University of Iowa, Department of Industrial and Management Engineering, Ames, Iowa.

Laird, J., *Recovery from Incorrect Knowledge in SOAR*, Proceedings from the AAAI-88 Seventh National Conference on Artificial Intelligence, Vol. 2, St. Paul, Minnesota, August 21-26, pp. 618-623, The MIT Press, Cambridge, Massachusetts 02142, 1988.

Laita, L.M., et. al., *A Theoretical Approach to Verification of Knowledge Based Systems*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, August 1990.

Landauer, C., *Correctness Principles for Rule-Based Expert Systems*, Expert Systems with Applications, Special Issue: Verification and Validation of Knowledge-Based Systems, Vol. 1, No. 3, pp. 291-316, 1990.

Landauer, C., *Empirical Methods in Expert System Validation*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Landauer, C., *Principles of RuleBase Correctness*, Verification, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, IJCAI-89, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 19, 1989.

Lane, N.E., *Global Issues in Evaluation of Expert Systems*, IEEE Expert, pp. 121-125, August 1986.

Laurent, J.P., and M. Ayel, *Off-Line Coherence Checking for Knowledge Based Systems*, Verification, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, IJCAAI-89, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 1989.

Laznovsky, A., *Automated Testing Mechanisms*, Worcester Polytechnic Institute, Worcester, Massachusetts, June 24, 1991.

Li, X., *The Rule-Based Paradigm Considered Harmful - A Critical View of An Industry Practitioner*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Linden, T.A., *A Meta-Level Software Development Model That Supports V&V for AI Software*, Expert Systems with Applications, Special Issue: Verification and Validation of Knowledge-Based Systems, Vol. 1, No. 3, pp. 271-279, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303.

Linden, T.A., *Alternative Approaches to V&V for AI Systems*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Linden, T.A., *A Meta-Level Software Development Model that Supports V&V for AI Software*, Verification, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, IJCAI-89, August 1989.

Llinas, J., S. Rizzi, and M. McCown, *The Test and Evaluation Process for Knowledge-Based Systems*, Technical Report of Science Applications International Corporation, San Diego, California, June 1987.

Loganantharaj, R., *Towards a Better Design Methodology of Expert Systems for Verification and Validation*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Loganantharaj, R., *Static Analysis of Consistency and Redundancy of a Rule Based Expert System*, The Second International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems IEA/AIE-89, The University of Southern Louisiana, Lafayete, Lousiana 70504, June 1989.

Lopez, B., P. Meseguer, and E. Plaza., *Knowledge Based Systems Validation: a State of the Art*, AI Communications Vol. 3, No. 2, pp. 58-72, June 1990.

Marcus, S., *SALT: A Knowledge Acquisition Tool that Checks and Helps Test a Knowledge Base*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88. The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Mars, N.J. I., and P.L. Miller, *Tools for Knowledge Acquisition and Verification in Medicine*, Proceedings - the Tenth Annual Symposium on Computer Applications in Medical Care, Washington, D.C., October 25-26, 1986, pp. 36-42, IEEE Service Center, Piscataway, New Jersey, 08854.

May, R.S., *EPRI Guide for Verification and Validation of Critical Function Software*, Draft, Prepared for EPRI Computer Products Group, S. Levy Inc., Electric Power Research Institute, Palo Alto, California, June 6, 1991.

McGuire, J.G., and R. Stiles, *Detecting Interference in Knowledge Base Systems*, Intelligent Systems Review, Vol. II, No. 314, IEEE Service Center, Piscataway, New Jersey, 08854, 1991.

McGuire, J.G., *Uncovering Redundancy and Rule-Inconsistency in Knowledge Bases via Deduction*, Presented at the Fifth Annual Conference on Computer Assurances: Systems Integrity, Software Safety, and Process Security, IEEE COMPASS-90, June 1990.

Mehrotra, M., and S.C. Johnson, *Importance of Rule Groupings in Verification of Expert Systems*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Michalski, R.S., and J.B. Larson, *Incremental Generation of VL1 Hypotheses: The Underlying Methodology and the Description of Program AQ11*, Reports of the Intelligent Systems Group ISG 83-5, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana-Champaign, Illinois, January 1983.

Miller, L.A., *New Challenges for the Validation and Verification of Knowledge-Based Systems*, American Association for Artificial Intelligence Panel Presentation: Issues in Validation of Knowledge-Based Systems, Reno, Nevada, Science Applications International Corporation, 1710 Goodridge Drive, McLean, Virginia 22102, January 1989.

Miller, L.A., *Tutorial on Validation and Verification of Knowledge-Based Systems*, Proceedings of the Conference on Expert Systems Applications for the Electric Power Industry, Science Applications International Corporation, 1710 Goodridge Drive, McLean, Virginia 22102, June 1989.

Miller, L.A., *Dynamic Testing of Knowledge Bases Using the Heuristic Testing Approach. Expert Systems with Applications: An International Journal*, Special Issue: Verification and Validation of Knowledge-Based Systems, Vol. 1, No. 3, pp. 249-269, 1990.

Miller, L.A., *A Comprehensive Approach to the Verification and Validation of Knowledge-Based Systems*, Submission to IJCAI Workshop on Verification, Validation and Testing of Knowledge-Based Systems, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 1989.

Miller, L.A., *Verification and Validation of Knowledge-based Systems with Emphasis on Life-Cycle*, Technical Report, Science Applications International Corporation, 1710 Goodridge Drive, McLean, Virginia 22102, August 1989.

Miller, L.A., *Expert System Verification and Validation for Nuclear Power Industry Applications*, presentation of American Nuclear-Society 1990 Winter Meeting, November 1990.

Miller, L.A., *A Realistic Industrial-Strength Life-Cycle Model for Knowledge-Based System Development and Testing*, Knowledge Based Systems Verification and Validation Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Moninger, W.R., T.R. Stewart, and P. McIntosh, *Validation of Knowledge-based Systems for Probabilistic Forecasting*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Montalban, M., and I. Jebel, Outils d'Aide a la Validation de Bases de Connaissances. Specification de la Representation du Controle et des Interfaces Graphiques Adaptees *(Tools for Validating Knowledge Bases, Specification of Inspection Graphing and Adapted Graphic Interfaces)*, Centre National de la Recherche Scientifique. Paris, France, July 1, 1989.

Moradian, A., R.F. Gray, and P.L. Wilhelm, *Generator Artificial Intelligence Diagnostics On-Line Experience*, Proceedings of the 4th Incipient Failure Detection Conference, Electric Power Research Institute, Palo Alto, California.

O'Keefe, R.M., and D.E. O'Leary, *The Verification and Validation of Expert Systems*, Technical Report, Technical Report of Renselaer Polytechnic Institute, Pittsburgh, Pennsylvania, 1991.

O'Leary, D.E., and R. O'Keefe, *Verifying and Validating Expert Systems, Tutorial Program*, Eleventh International Joint Conference on Artificial Intelligence, University of Southern California, Los Angeles, California, August 1989.

O'Leary, T.J., M. Goul, K.E. Moffitt, and E.E. Radwon, *Validating Expert Systems*, IEEE Expert Vol 5, No 3, pp. 51-58, June 1990.

O'Leary, D.E., and N.A. Kanderin, *Validating the Weights in Rule-Based Expert Systems: a Statistical Approach*, International Journal of Expert Systems Research and Applications, Vol. 1, No. 3, pp. 253-279, 1988.

O'Leary, D.E., *Determining the Existence of Difference in Expert Judgement*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

O'Leary, D.E., *Methods of Validating Expert Systems*, Interfaces, Vol. 18, No. 6., pp. 72-79, Nov-Dec 1988.

O'Leary, D.E., *Extended Abstract - A Computer Intensive Statistics Approach to Validating Expert Systems*, Verification, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, IJCAI-89, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 1989.

O'Leary, D.E., *Verification of Frame-based Knowledge Base*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Wilhelm, and S.S. Palusamy, *Artificial Intelligence Applications to Electric Power Plant On-Line Diagnostics*, Pacific Rim International Conference on Artificial InteMoradian, A., P.L. lligence, Westinghouse Electric Corporation, Orlando, Florida 32817, 1990.

Moradian, A., E.D. Thompson, M.A. Jenkins, G.H. Tomlinson, and T. Trepagnier, *On-Line AI Diagnostics for Utility Generating Stations*, Society for Electric Power Research and Implementation (SEPRI), Westinghouse Electric Corporation, Orlando, Florida.

Morell, L.J., *Use of Metaknowledge in the Verification of Knowledge-Based Systems (Final Report)*, College of William and Mary, Williamsburg, Virginia, Grant GT 47-003-029, NASA Contractor Report 181821, April 1989.

Moriconi, R., *A Designer/Verifier's Assistant*, Readings in Artificial Intelligence and Software Engineering, ed., C. Rich, and R.C. Waters, Morgan Kaufmann Publishers, Los Altos, California, pp. 335-350, 1986.

Motoda, H., *The Current Status of Expert System Development and Related Technologies in Japan*, IEEE Expert, pp. 3-11, August 1990.

Myers, R.M., *Verification of Expert Systems: A Preliminary Analysis*, Proceedings of ROBEXS '87 Third Annual Workshop on Robotics and Expert Systems, pp. 231-239, ISA (Robation & Expert Systems V.3) Research Triangle Park, North Carolina, June 1987.

Naser, J., S. Bhatt, B. Su, and R. May, *Digital Software Verification and Validation Methodologies and Tools*, Proceedings of the American Power Conference, April/May 1991, pp. 602-605, Electric Power Research Institute, Palo Alto, California.

Naser, J.A., *Nuclear Power Plant Expert System Verification & Validation*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Nazareth, D.L., *Discussion of Empirical Methods in Expert System Validation*, Knowledge Based Systems Verification and Validation Workshop Proceedings, AAAI-90, ,The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Nazareth, D.L., *Issues in the Verification of Knowledge in Rule-Based Systems*, International Journal of Man-Machine Studies, Vol. 30 No. 3, pp. 255-271, March 1989.
Nazareth, D.L., *Verification of Rule-Based Knowledge Using Directed Graphics*, May 1990. Available upon request from University of Wisconsin-Milwaukee, School of Business Administration, Milwaukee, Wisconsin.

Neches, R., W.R. Swartout, and J. Moore, *Enhanced Maintenance and Explanation of Expert Systems through Explicit Models of their Development*, IEEE Transactions of Software Engineering, Vol. SE-11 No. 11, pp. 1337-1351 November 1985.

Neches, R., W.R. Swartout, and J. Moore, *Explainable (and Maintainable) Expert Systems*, IJAC-85, Proceedings of the Ninth International Joint Conference on Artificial Intelligence. Vol. 1, pp. 382-389, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 1985.

Nelson, R., *Expert System to be Tested for Plant Control and Heat Rate Optimization. Controls and Automation Update*, p. 3, Electric Power Research Institute, Palo Alto, California, October 1989.

Nguyen, T.A., W.A. Perkins, T.J. Laffery, and D. Pecora, *Knowledge Base Verification*, AI Magazine, Vol. 8, No. 2, pp. 69-75.

Nguyen, T.A., *Verifying Consistency of Production Systems*, Proceedings of the Third Conference on Artificial Intelligence Applications, Washington, D.C., IEEE Service Center, Piscataway, New Jersey, 08854, February 1987.

Nicoud, S., *Requirements for a Database System to Support Development of Large-Scale, Expert Systems*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

O'Keefe, R.M., O. Balci, and E.P. Smith, *Validating Expert System Performance*, IEEE Expert, pp. 81-90, Winter 1987.

O'Neil, M., and A. Glowinski, *Evaluating and Validating Very Large Knowledge-based Systems*, Medical Infomatics, Vol 15, No 3, pp. 237-251, July/September 1990.

Odubiyi, J.B., *Probability Based Dependency-Directed Backtracking: An Approach for Developing a Quality Metric for a Knowledge-Based System*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Odubiyi, J.B., *Probability Based Dependency-Directed Backtracking: An Approach for Developing a Quality Metric for a Knowledge-Based System*, PhD. thesis, Kennedy-Western University, April 1990.
Oliver, A.E.M., *Techniques for Expert System Testing and Validation*, Third International Expert Systems Conference, pp. 271-276, Learned Inf., Oxford, United Kingdom, June 1987.

Osborne, Dr. R.L., *Online, Artificial Intelligence-Based Turbine Generator Diagnostics*, AI Magazine Vol. 7 No. 4, pp. 97-103 Fall 1986.

Oxman, S.W., *Reporting Chemical Spills: An Expert Solution*, AI Expert, pp. 50-51, May 1991.

Park, Y.T., and D.C. Wilkins, *Establishing the Coherence of an Explanation to Improve Refinement of an Incomplete Knowledge Base*, AAAI-90, Proceedings Eighth National Conference on Artificial Intelligence, Vol. 1, pp. 511-516, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Parsaye, K., *Acquiring & Verifying Knowledge Automatically*, AI Expert, Vol. 3, No. 5, pp. 48-63, May 1988.

PATHFINDER, Advanced Computing Technology, Inc. *Realm: Validation and Verification Plan, Revision 3*, PATHFINDER, Advanced Computing Technology, Inc., jACKSONVILLE, fLORIDA 32216, April 1991.

Pau, L.F., *Prototyping, Validation and Maintenance of Knowledge Based Systems Software*, Third Annual Expert Systems in Government Conference - Proceedings, pp. 248-253, IEEE Service Center, Piscataway, New Jersey, 08854, October 1987.

Pazzani, M.J., and C.A. Brunk, *Detecting and Correcting Errors in Rule-Based Expert Systems: An Integration of Empirical and Explanation-based Learning*, Technical Report 90-38, Proceedings of the 5th AAAI Knowledge Acquisition for Knowledge-based Systems Workshop, The MIT Press, Cambridge, Massachusetts 02142, November 1990.

Pazzani, M.J., *A Set Covering Approach to Testing Rule-Based Expert Systems*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Plant, R.T., and D. Gold, *Increasing Expert System Reliability Through the Use of a Formal Specification*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, July 1990, The MIT Press, Cambridge, Massachusetts 02142.

Porter, J.F., L.C. Kingsland III, D.A.B. Lindberg, I. Shah, J.M. Benge, S.E. Hazelwood, D.R. Kay, M. Homma, M. Akizuki, M. Takano, and G.C. Sharp, *The AI/RHEUM knowledge-based Computer consultant System in Rheumatology: Performance in the Diagnosis of 59 Connective Tissue Disease Patients from Japan*, Arthritis and Rheumatism, Vol 31, No. 2, pp. 219-226, February 1988.

Preece, A.D., *The Role of Specifications in Expert System Evaluation*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

O'Keefe, R.M., and S. Lee, *An Integrative Model of Expert System Verification and Validation*, Expert Systems with Applications, Special Issue: Verification and Validation of Knowledge-Based Systems, Vol. 1, No. 3, pp. 231-236, 1990.

O'Keefe, R.M., *A Systems View of Validating Expert Knowledge Systems*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Preece, A.D., and R. Shinghal, *Practical Approach to Knowledge Base Verification*, Ed. M. Trivedi Proceedings of Applications of Artificial Intelligence IX, pp. 608-619, Concordia University, Montreal, Canada H3G 1M8, April 1991.

Preece, A.D., *Towards a Methodology for Evaluating Expert Systems*, Expert Systems, Vol. 7, No. 4, pp. 215-223, November 1990.

Preece, A.D., *A New Approach to Detecting Missing Knowledge in Expert System Rule Bases*, Submitted to the International Journal of Man-Machine Studies, Concordia University, Montreal, Canada H3G 1M8, November 1990.

Prerau, D.S., *Developing and Managing Expert Systems: Proven Techniques for Business and Industry*, Addison-Wesley Publishing Company, 1990.

*Proceedings of the International Workshop on Artificial Intelligence for Industrial Applications*, Institute of Electrical and Electronic Engineers, Inc., Hitachi City, Japan, IEEE, Order Dept., Piscataway, New Jersey 08854, May 1988.

Quinlan, J.R., *Induction of Decision Trees*, Machine Learning 1, Kluwer Academic Publishers, Boston-Manufactured in The Netherlands, pp. 81-106, 1986.

Radwan, A.E., M. Goul, T.J. O'Leary, and K.E. Moffitt, *Verification Approach for Knowledge-based Systems*, Transportation Research, Part A: General Vol 23A, No. 4, pp. 287-300, July 1989.

Rajamoney, S., and G. DeJong, *The Classification, Detection and Handling of Imperfect Theory Problems*, IJCAI'87, Proceedings of the Tenth International Joint Conference on Artificial Intelligence, Vol. 1, Milan, pp. 205-207, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 1987.

Real-Time Intelligent Systems Corporation (The), *Reference Manual: Activation Framework Operating System Environment AFC Version 2.5*, The Real-Time Intelligent Systems Corporation, Worcester, Massachusets 01606, 1991.

Real-Time Intelligent Systems Corporation (The), *User's Manual: Activation Framework Operating System Environment AFC Version 2.5.*, The Real-Time Intelligent Systems Corporation, Worcester, Massachusets 01606, 1991.

Ribar, G., F. Arcoleo, and D. Hollo, *LOAN PROBE: Testing a Big Expert System*, AI Expert, pp. 43-49, May 1991.

Richardson, K., and C. Wong, *Knowledge Based System Verification and Validation as Related to Automation of Space Station Subsystems: Rationale for a Knowledge Based System Lifecycle*, NASA, Marshall Space Flight Center, Third Conference on Artificial Intelligence for Space Applications, Part 2, pp. 25-30, National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22161, June 1988.

Ricks, W.R., and K.H. Abbott. *Evaluation of Knowledge-based Systems*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Rippon, S., *Three Computerized Control Rooms*, Nuclear News, pp. 60-63, October 1990.

Roscoe, G.S., *9370 Expert System Configurator Knowledge Base Verification*, Fourth International Expert Systems Conference, London, United Kingdom, pp. 159-167, Learned Info. Center, Oxford, United Kingdom, June 1988.

Rossomando, P.J., *Critique of the Paper: Importance of Rule Groupings in Verification of Expert Systems*, Proceedings Eighth National Conference on AI, AAAI-90, July 1990.

Rossomando, P.J., *A Knowledge Based Environment for the Verification and Validation of Knowledge Based Systems*, Verification, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, IJCAI-89, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 1989.

Rothschild, M.A., H.A. Swett, P.R. Fisher, G.G. Weltin, and P.L. Miller, *Exploring Subjective vs. Objective Issues in the Validation of Computer-based Critiquing Advice*, Computer Methods and Programs in Biomedicine, Vol. 31, No. 1., pp. 11-18, January 1990.

Rousset, M.C., *On the Consistency of Knowledge Bases: The COVADIS System*, Computational Intelligence, Special 8Issue, September 1988.

Rousset, M.C., *On the Coherence and Validity of Rule Bases in Expert Systems: COVADIS System*. Genie Logiciel & Systemes Experts, No. 10, pp. 76-80, January 1988.

Rushby, J., M.E. Stickel, and R.J. Waldinger, *Engineering for Artificial Intelligence Software*, SRI International, Contract No. R009406, SRI Project 7447, Final Report, SRI International, Menlo Park, California 94025, September 1990.

Rushby, J., and J. Crow., *Evaluation of an Expert System for Fault Detection, Isolation, and Recovery in the Manned Maneuvering Unit*, NASA Contractor Report 187466, Contract NAS1-18226, SRI International, Menlo Park, California 94025, December 1990.

Rushby, J., *Formal Specification and Verification of a Fault-Masking and Transient-Recovery Model for Digital Flight-Control Systems*, SRI International, SRI-CSL-91-03, CSL Technical Report, SRI International, Menlo Park, California 94025, June 1991.

Rushby, J., F. von Henke, and S. Owre, *An Introduction to Formal Specification and Verification Using EHDM*, SRI International, SRI-CSL-91-02, CSL Technical Report, SRI International, Menlo Park, California 94025, February 1991.

Rushby, J., F. von Henke, and S. Owre, *An Introduction to Formal Specification and Verification Using EHDM*, SRI International, SRI-CSL-91-02, CSL Technical Report, SRI International, Menlo Park, California 94025, February 1991.

Rushby, J., and R.A. Whitehurst, *Formal Verification of AI (Artificial Intelligence) Software SRI International Final Report*, SRI International, Menlo Park, California 94025, February 1989.

Rushby, J., *Validation and Testing of Knowledge-Based Systems: How Bad Can it get?*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Rushby, J., *Quality Measures and Assurance for AI Software*, NASA Contractor Report 4187, SRI International, Menlo Park, California 94025, October 1988.

Sary, C., L. Gilstrap, and L.G. Hull, *Expert System Development Methodology (ESDM)*, Presented at the 5th Conference on AI for Space Applications, Huntsville, Alabama, May 1990.

Schultz, Dr. R.D., and J.R. Geissman, *Bridging the Gap Between Static and Dynamic Verification*, Submission to AAAI-88 Workshop on Validation and Testing of Knowledge-Based Systems, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Schumaker, R., and L.C. Davis, *Expert Systems at the Navy Center for Applied Research in Artificial Intelligence (NCARAI)*, Expert Systems With Applications, Vol. 1, No. 1, pp. 71-77, 1990.

Science Applications International Corporation, *Guidelines for Verification and Validation of Expert Systems, Task 1: Review of Conventional Methods*, draft, April 1991. Prepared for Nuclear Regulatory Commission and Electric Power Research Institute.

Seale, M.R., *Summary Presentation of Relevant Work in Knowledge Base System (KBS) Management*, Intelligent Training Technology, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Sebo, D., D. Marksberry, and M. Modarres, *RSAS: A Reactor Safety Assessment System. Proceedings of the Seventh Power Plan Dynamics, Control, and Testing Symposium*, IEEE, Knoxville, Tennessee, IEEE Service Center, Piscataway, New Jersey, 08854, May 1989.

Shankar, N., *Mechanical Verification of a Schematic Protocol for Byzantine Fault-Tolerant Clock Synchronization*, SRI International, SRI-CSL-91-04, CSL Technical Report, June 1991.

Sizemore, N.L., *Test Techniques for Knowledge-Based Systems*, ITEA Journal, Vol. 11, No. 2, 1990.

Skingle, B., *Exams for the Expert System*, Computer Systems Europe, pp. 61-64, March 1989.

Smith, R.G., H.A. Winston, T.M. Mitchell, and B.G. Buchanan, *Representation and Use of Explicit Justifications for Knowledge Base Refinement*, IJAC-85, Proceedings of the Ninth International Joint Conference on Artificial Intelligence, Vol. 1, pp. 671-680, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 1985.

Soloway, E., J. Bachant, and K. Jensen, *Assessing the Maintainability of XCON-in-RIME: Coping with the Problems of a VERY Large Rule-Base*, Proceedings of the Sixth National Conference on Artificial Intelligence, Vol. 2, AAAI-'87, pp. 824-829, The MIT Press, Cambridge, Massachusetts 02142, July 1987.

St. Clair, D.C., W.E. Bond, and B.B. Flachsbart, *Using Output to Evaluate and Refine Rules in Rule-Based Expert Systems*, pp. 9-13, McDonnell Douglas Research Laboratories, McDonnell Douglas Corporation, St. Louis, Missouri, November 1987.

Stachowitz, R.A., and J.B. Combs, *Validation of Expert Systems*, Proceedings of the Hawaii International Conference on Systems Sciences, Kona, Hawaii, Lockheed Corporation, Palo Alto, California 94304, January 1987.

Stachowitz, R.A., C.L. Chang, T.S. Stock, and J.B. Combs, *Building Validation Tools for Knowledge-Based Systems*, First Annual Workshop on Space Operations Automation and Robotics (SOAR '87), NASA Johnson Space Center, Houston, Texas, August 1987.

Stachowitz, R.A., C.L. Chang, and J.B. Combs, *Research on Validation of Knowledge-Based Systems*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

*Structuring Expert Systems: Domain, Design, and Development*, Ed. J. Liebowitz and D.A. DeSalvo, Chapter 11, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

Surko, P.T., J.M. Achroff, and J.R. Wright, *Autotest 2: An Expert System for Special Services*,AT&T Bell Laboratories, Warren, New Jersey.

Suwa, M., A.C. Scott, and E.H. Shortliffe, *An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System*, The AI Magazine, Vol. 3, No. 4, pp. 16-21, Fall 1982.

Szatkowski, Dr. G.P., *Approaches to Verification/ Validation of Imbedded Decision Support Systems Applied to Launch Vehicle Operations*, Technical Report of General Dynamics/Space Systems Division, San Diego, California.

Tao, Y., H. Zhijun, and Y. Ruizhao, *Performance Evaluation of the Inference Structure in Expert System*, IJCAI87, Proceedings of the Tenth International Joint Conference on Artificial Intelligence, Vol. 2, pp. 945-950, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 1987.

Terano, T., H. Kongoji, K. Kaji, and K. Yamamoto, *Development of a Guideline for Expert System Evaluation - A Report on a Three-year Project*, Submitted to the AAAI-91 workshop on VV&T of KBS's held in Anaheim, California, July 1991, The MIT Press, Cambridge, Massachusetts 02142, .

Terano, T., H. Kongoji, K. Kaji, and K. Yamamoto, *Developing a Guideline for Expert Evaluation -- A Midterm Report*, The University of Tsukba, Tokyo, Japan, August 1990.

Terano, T., and S. Kobayashi, *Problem Analyses, Tool Evaluation, and Verification and Validation, Study: Three Steps for Knowledge-Based Systems Development Methodology*, Verification, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, IJCAI-89, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 1989.

Thornton, D.S., *Expert System Life Cycle and Test*, Technical Report, IBM Federal Systems Division, Gaithersburg, Maryland.

Tuthill, G.S., *Legal Liabilities and Expert Systems*, AI Expert, Vol. 6, No. 3, pp. 44-51, March 1991.

U.S. Army Information Systems Engineering Command (USAISEC), *Expert System Development Methodology (ASQBG-A-89-033)*, USAISEC, Fort Huachuca, Arizona, July 1989.

U.S. Army Information Systems Engineering Command (USAISEC), *Expert System Evaluation Methodology (ASQBG-A-89-034)*, USAISEC, Fort Huachuca, Arizona, July 1989.

Van de Brug, A., J. Bachant, and J. McDermott, *The Taming of R1*, IEEE Expert, pp. 33-38, Fall 1986.

Vick, S., and K. Lindenmayer, *Verification and Validation of Rulebased Systems for Hubble Space Telescope Ground Support*, The 1988 Goddard Conference on Space Applications of Artificial Intelligence, NASA, Goddard Space Flight Center, pp. 435-448, National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22161, August 1988.

Vignollet, L., and M. Ayel, *A Conceptual Model For Building Sets of Test Samples For Knowledge Bases*, ECAI Proceedings, Stockholm, August 1990, University of SAVOIE, France.

Vignollet, L., and M. Ayel, *SYCOJET: A Tool for Buildings Automatically Sets of Tests for Knowledge Bases*, Proceedings of the SPIE - The International Society for Optical Engineering, Orlando, Florida, April 1990, pp. 192-201.

Vinze, A.S., D.R. Vogel, and J.F. Nunamaker Jr., *Validation of a Knowledge-based System: The ICE Case*, Proceedings of the Twenty-third Annual Hawaii International Conference on System Sciences. Volume 3: Decision Support and Knowledge Based Systems Track. Kailua-Kona, Hawaii, pp. 239-246, IEEE Service Center, Piscataway, New Jersey, 08854, January 1990.

Waldinger, R.J., and M.E. Stickel, *Proving Properties of Rule-Based Systems*, Proceedings of the Seventh Conference on Artificial Intelligence Applications held at Miami Beach,Florida, February 24-28, 1991, pp. 81-88, IEEE Service Center, Piscataway, New Jersey, 08855.

Wensley, A.K.P, *Verifying and Validating an Audit Planning Expert System*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Wilkins, D.C., *Knowledge Base Refinement Using Apprenticeship Learning Techniques*, Proceedings from the AAAI-88 Seventh National Conference on Artificial Intelligence, Vol. 2, pp. 646-651, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Wilson, M., D. Duce, and D. Simpson, *Life Cycles in Software and Knowledge Engineering: a Comparative Review*, Knowledge Engineering Review Vol. 4, No. 3, pp. 189-204, September 1989.

Wise, B., *Summary of Work and Position*, Validation and Testing Knowledge-Based Systems Workshop Proceedings, AAAI-88, The MIT Press, Cambridge, Massachusetts 02142, August 1988.

Wolfgram, D.D., T.J. Dear, and C.S. Galbraith, *Expert Systems for the Technical Professional*, John Wiley & Sons, New York, 1987.

Wood, W.T., and E.N. Frankowski, *Verification of Rule-Based Expert Systems*, Expert Systems with Applications, Special Issue: Verification and Validation of Knowledge-Based Systems. Vol. 1, No. 3, pp. 317-322, Carnegie-Mellon University Library, Carnegie-Mellon University, EDSH 109, Pittsburgh, Pennsylvania 15213-3890, 1990.

Wood, W.T., and E.N. Frankowski, *Rule Based Expert System Verification*, Verification, Validation and Testing of Knowledge-Based Systems Workshop Proceedings, IJCAI-89, Morgan Kaufmann Publishers, Inc., Palo Alto, California 94303, August 1989.

Yager, R.R., *On Incomplete and Uncertain Knowledge Bases*, Institute of Electrical and Electronic Engineers, Inc., pp. 96-100, IEEE Service Center, Piscataway, New Jersey, 08854, September 1986.

Yen, J., J. Lee, and D. Hamilton, *Designing Verifiable Expert Systems*, Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence, IEEE Service Center, Piscataway, New Jersey, 08854, November 1990.

Yen, J., D. Hamilton, and J. Lee, *Towards a Design Methodology for Verification of Expert Systems*, Knowledge Based Systems Verification, Validation and Testing Workshop Proceedings, AAAI-90, The MIT Press, Cambridge, Massachusetts 02142, July 1990.

Yoon, J.P., *Techniques for Data and Rule Validation in Knowledge Based Systems*, Proceedings of the Fourth Annual Conference on Computer Assurance: Systems Integrity, Software Safety and Process Security, Gaithersburg, Maryland, IEEE Service Center, Piscataway, New Jersey, 08854, June 1989.

Yu, V.L., B.G. Buchanan, E.H. Shortliffe, S.M. Wraith, R. Davis, A.C. Scott, and S.N. Cohen, *Evaluating The Performance of a Computer-Based Consultant*, Computer Programs in Biomedicine, 9, pp. 95-102, 1979.

NRC FORM 335
(2-89)
NRCM 1102,
3201, 3202

U.S. NUCLEAR REGULATORY COMMISSION

**BIBLIOGRAPHIC DATA SHEET**

*(See instructions on the reverse)*

1. REPORT NUMBER (Assigned by NRC. Add Vol., Supp., Rev., and Addendum Numbers, if any.)

NUREG/CR-6316
SAIC-95/1028
Vol. 3

2. TITLE AND SUBTITLE

Guidelines for the Verification and Validation of Expert System Software and Conventional Software

Survey and Documentation of Expert System Verification and Validation Methodologies

3. DATE REPORT PUBLISHED

| MONTH | YEAR |
|---|---|
| March | 1995 |

4. FIN OR GRANT NUMBER

L1530

5. AUTHOR(S)

E.H. Groundwater, L.A. Miller, S.M. Mirsky

6. TYPE OF REPORT

7. PERIOD COVERED *(Inclusive Dates)*

8. PERFORMING ORGANIZATION — NAME AND ADDRESS *(If NRC, provide Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address; if contractor, provide name and mailing address.)*

Science Application International Corporation
1710 Goodridge Drive
McLean, VA 22102

9. SPONSORING ORGANIZATION — NAME AND ADDRESS *(If NRC, type "Same as above"; if contractor, provide NRC Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address.)*

Division of Systems Technology
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001

Nuclear Power Division
Electric Power Research Institute
3412 Hillview Avenue
Palo Alto, CA 94303

10. SUPPLEMENTARY NOTES

11. ABSTRACT *(200 words or less)*

This report is the third volume in a series of reports describing the results of the Expert System Verification and Validation (V&V) project that is jointly funded by the U.S. Nuclear Regulatory Commission and the Electric Power Research Institute to develop guidelines for the V&V of expert and other systems. The purpose of this activity was to survey and document techniques presently in use for expert systems V&V. Via extensive telephone contacts, site visits, and through bibliographic searches a wide sampling of expert system V&V was accomplished. The major finding was that V&V of expert systems is not nearly as established or prevalent as V&V of conventional software systems. There were few examples of V&V in the early stage of development. However, there is a very active research area concerning the development of methods to assess the knowledge bases of expert and knowledge-based systems.

12. KEY WORDS/DESCRIPTORS *(List words or phrases that will assist researchers in locating the report.)*

validation, verification, V&V expert systems, knowledge base, guidelines, scenarios, software quality assurance

13. AVAILABILITY STATEMENT

Unlimited

14. SECURITY CLASSIFICATION

*(This Page)*

Unclassified

*(This Report)*

Unclassified

15. NUMBER OF PAGES

16. PRICE

NRC FORM 335 (2-89)