RECEIVED

APR 21 1995

OSTI

NUREG/CR-6316

SAIC-95/1028

Vol. 4

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

Evaluation of Knowledge Base
Certification Methods

Prepared by
L. A. Miller, J. E. Hayes, S. M. Mirsky

Science Applications International Corporation

Prepared for
U.S. Nuclear Regulatory Commission

and

Electric Power Research Institute

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

AVAILABILITY NOTICE

Availability of Reference Materials Cited in NRC Publications

Most documents cited in NRC publications will be available from one of the following sources:

1. The NRC Public Document Room, 2120 L Street, NW., Lower Level, Washington, DC 20555-0001
2. The Superintendent of Documents, U.S. Government Printing Office, P. O. Box 37082, Washington, DC 20402-9328
3. The National Technical Information Service, Springfield, VA 22161-0002

Although the listing that follows represents the majority of documents cited in NRC publications, it is not intended to be exhaustive.

Referenced documents available for inspection and copying for a fee from the NRC Public Document Room include NRC correspondence and internal NRC memoranda; NRC bulletins, circulars, information notices, inspection and investigation notices; licensee event reports; vendor reports and correspondence; Commission papers; and applicant and licensee documents and correspondence.

The following documents in the NUREG series are available for purchase from the Government Printing Office: formal NRC staff and contractor reports, NRC-sponsored conference proceedings, international agreement reports, grantee reports, and NRC booklets and brochures. Also available are regulatory guides, NRC regulations in the *Code of Federal Regulations*, and *Nuclear Regulatory Commission Issuances*.

Documents available from the National Technical Information Service include NUREG-series reports and technical reports prepared by other Federal agencies and reports prepared by the Atomic Energy Commission, forerunner agency to the Nuclear Regulatory Commission.

Documents available from public and special technical libraries include all open literature items, such as books, journal articles, and transactions. *Federal Register* notices, Federal and State legislation, and congressional reports can usually be obtained from these libraries.

Documents such as theses, dissertations, foreign reports and translations, and non-NRC conference proceedings are available for purchase from the organization sponsoring the publication cited.

Single copies of NRC draft reports are available free, to the extent of supply, upon written request to the Office of Administration, Distribution and Mail Services Section, U.S. Nuclear Regulatory Commission, Washington, DC 20555-0001.

Copies of industry codes and standards used in a substantive manner in the NRC regulatory process are maintained at the NRC Library, Two White Flint North, 11545 Rockville Pike, Rockville, MD 20852-2738, for use by the public. Codes and standards are usually copyrighted and may be purchased from the originating organization or, if they are American National Standards, from the American National Standards Institute, 1430 Broadway, New York, NY 10018-3308.

DISCLAIMER NOTICE

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for any third party's use, or the results of such use, of any information, apparatus, product, or process disclosed in this report, or represents that its use by such third party would not infringe privately owned rights.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

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

Evaluation of Knowledge Base
Certification Methods

Manuscript Completed: February 1995
Date Published: March 1995

Prepared by
L. A. Miller, J. E. Hayes, 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

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED
me

MASTER

ABSTRACT

This report presents the results of the Knowledge Base Certification activity of the expert systems verification and validation (V&V) guideline development project which is jointly funded by the US 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. This activity is concerned with the development and testing of various methods for assuring the quality of knowledge bases. The testing procedure used was that of behavioral experiment, the first known such evaluation of any type of V&V activity. The value of such experimentation is its capability to provide empirical evidence for -- or against -- the effectiveness of plausible methods in helping people find problems in knowledge bases.

The three-day experiment included 20 participants from three nuclear utilities, the Nuclear Regulatory Commission's Technical training Center, the University of Maryland, EG&G Idaho, and SAIC. The study used two real nuclear expert systems: a boiling water reactor emergency operating procedures tracking system and a pressurized water reactor safety assessment systems. Ten participants were assigned to each of the expert systems. All participants were trained in and then used a sequence of four different V&V methods selected as being the best and most appropriate for study on the basis of prior evaluation activities. These methods either involved the analysis and tracing of requirements to elements in the knowledge base (requirements grouping and requirements tracing) or else involved direct inspection of the knowledge base for various kinds of errors. Half of the subjects within each system group used the best manual variant of the V&V methods (the control group), while the other half were supported by the results of applying real or simulated automated tools to the knowledge bases (the experimental group). The four groups of participants were similar in nuclear engineering and software experience characteristics.

The major conclusion was that the use of tools in static knowledge base certification results in significant improvement in detecting all types of defects, avoiding false-alarms, and completing the effort in less time. The simulated knowledge-checking tool, which was based on supplemental engineering information about the systems, was the most effective of the tools. Testing experience and extensive engineering experience was not associated with improved performance, implying that such expensive labor skills might not be necessary for V&V supported by effective automated tools.

TABLE OF CONTENTS

ABSTRACT	iii
EXECUTIVE SUMMARY	xi
1. INTRODUCTION	1
1.1 Purpose and Scope of knowledge Base Certification Activity	1
1.2 Relationship to Other Project Activities	1
1.3 Report Organization	3
2. SELECTION OF KNOWLEDGE BASE CERTIFICATION METHODS	5
2.1 Identification of General Selection Issues	5
2.2 Resolution of Issues	5
2.3 Selection of the Methods to be Experimentally Tested	7
2.4 Determining the Experimental Form of the Four Selected Methods	9
2.4.1 Requirements Tracing	10
2.4.1.1 Grouping of Requirements	10
2.4.1.2 Tracing of Requirements	10
2.4.2 Syntax Checking	12
2.4.3 Knowledge Checking	14
3. DESIGN OF BEHAVIORAL EXPERIMENT	21
3.1 Experimental Design	21
3.2 Performance Measures	24
3.3 Response Sheets	24
3.4 Selection and Assignment of Participants	24
3.5 Error Seeding	30
4. KNOWLEDGE BASE CERTIFICATION EXPERIMENT	41
4.1 Introduction and Overview	42
4.2 General Instruction and Training	42
4.2.1 Expert Systems Training	42
4.2.2 Verification and Validation Training	42
4.3 Systems A and B Training	42
4.4 Requirements Tracing	43
4.4.1 General Requirements Tracing Training	43
4.4.2 Experimental (Method 1) Requirements Tracing	44
4.4.2.1 Experimental (Method 1) Requirements Tracing Training	44
4.4.2.2 Experimental (Method 1) Requirements Tracing Materials	44
4.4.2.3 Experimental (Method 1) Requirements Tracing Experiment	45

4.4.3	Control (Method 2) Requirements Tracing	45
4.4.3.1	Control (Method 2) Requirements Tracing Training	45
4.4.3.2	Control (Method 2) Requirements Tracing Materials	45
4.4.3.3	Control (Method 2) Requirements Tracing Experiment	46
4.5	Syntax Checking	46
4.5.1	General Syntax Checking Training	46
4.5.2	Experimental (Method 1) Syntax Checking	47
4.5.2.1	Experimental (Method 1) Syntax Checking Training	47
4.5.2.2	Experimental (Method 1) Syntax Checking Materials	47
4.5.2.3	Experimental (Method 1) Syntax Checking Experiment	47
4.5.3	Control (Method 2) Syntax Checking	47
4.5.3.1	Control (Method 2) Syntax Checking Training	48
4.5.3.2	Control (Method 2) Syntax Checking Materials	48
4.5.3.3	Control (Method 2) Syntax Checking Experiment	48
4.6	Knowledge Checking	48
4.6.1	General Knowledge Checking Training	48
4.6.2	Experimental (Method 1) Knowledge Checking	49
4.6.2.1	Experimental (Method 1) Knowledge Checking Training	49
4.6.2.2	Experimental (Method 1) Knowledge Checking Materials	49
4.6.2.3	Experimental (Method 1) Knowledge Checking Experiment	49
4.6.3	Control (Method 2) Knowledge Checking	49
4.6.3.1	Control (Method 2) Knowledge Checking Training	49
4.6.3.2	Control (Method 2) Knowledge Checking Materials	50
4.6.3.3	Control (Method 2) Knowledge Checking Experiment	50
4.7	Debriefing	50
5.	EXPERIMENTAL RESULTS AND DISCUSSION	53
5.1	Three Essential Concepts	53
5.2	Conservatism of Analyses	54
5.3	Overview of Findings	54
5.4	Effects of System and Condition	56
5.4.1	Condition Effects	56
5.4.2	System Effects	62
5.4.3	System and Condition Effects	62
5.4.4	Method	63
5.5	Requirements Grouping and Tracing Findings	64
5.5.1	Requirements Grouping Findings	64
5.5.2	Requirements Tracing Findings	68
5.6	Syntax Checking Findings	70
5.7	Knowledge Checking Findings	72
5.8	Findings Based on Participants' Experience	75
5.8.1	Testing Experience with No Nuclear Experience	76
5.8.2	Testing Experience with Some Nuclear Experience	76

5.8.3	Importance of Significant Nuclear Experience	76
5.9	Undetected Errors	77
5.10	Participant Comments	78
6.	CONCLUSIONS AND RECOMMENDATIONS	81
6.1	Condition Effects	81
6.2	System Effects	81
6.3	Comparison of Tool-to-Tool Effects	81
6.4	Individual Method Experimental-to-Control Group Effects	81
6.5	Participants' Experience Effects	82
6.6	Conclusions	82
7.	REFERENCES	85
APPENDIX A - Seeded Errors		87
Table A1	System A - Syntax Checking Errors	88
Table A2	System A - Knowledge Checking Errors	97
Table A3	System B - Syntax Checking Errors	108
Table A4	System B - Knowledge Checking Errors	110
APPENDIX B - Experimental Means (Averages)		112
Table B1	Method Means	113
Table B2	Method and Condition Means	114
Table B3	System and Method Means	115
Table B4	Method, System, and Condition Means	116
Table B5	System Means	117
Table B6	Condition Means	118
Table B7	Means by System and Condition	119
Table B8	Analysis of System by Method Interaction	120

LIST OF FIGURES

Figure 1.2-1	Activity Dependencies and Interrelationships	2
Figure 2.4.1.1-1	Sample Grouped Requirements Printout	11
Figure 2.4.1.2-1	Sample Requirements, Traceability Map	13
Figure 2.4.2-1	A Subset of VERITE' Output for System A	15
Figure 2.4.2-2	A Subset of VERITE' Output for System B	17
Figure 2.4.3-1	Sample Output from Meta-Check System A	19
Figure 3.1-1	Depiction of the Experimental Design	22
Figure 3.1-2	USNRC/EPRI Knowledge Base Certification Experiment Agenda	25
Figure 3.3-1	Answer Sheet 2	29
Figure 3.4-1	Participant Survey for Activity 6	31
Figure 3.4-2	Experimental Teams	33
Figure 3.5-1	Erros, Locations	35
Figure 3.5-2	Mapping of Detectability Levels onto Difficulty Categories	39
Figure 5.4-1	Representative Significant Results for System and Condition	59
Figure 5.5-1	Representative Significant Results for Requirements Grouping	65
Figure 5.5-2	Representative Significant Results for Requirements Tracing	66
Figure 5.6-1	Representative Significant Results for Syntax Checking	71
Figure 5.7-1	Representative Significant Results for Knowledge Checking	73

LIST OF TABLES

Table 2.2-1	Rank-ordering of the Top 10 Rated Static-Testing Techniques for the Cost-Benefit Metric and the Extreme Effectiveness Metrics	8
Table 2.4.3-1	Errors Detectable From the Engineering Knowledge Compiled for the Knowledge-Checking Capability and Simulated Tool	19
Table 3.1-1	Planned Analysis-of-Variance Model for the Overall Experiment	23
Table 3.2-1	Description of the Planned Experimental Dependent Variables	27
Table 3.4-1	Experiment Participant Key Qualifications and Group Assignment	32
Table 3.5-1	Characterization of Possible Errors	37
Table 5.3-1	Significance of Findings for the 24 Dependent Variables	55
Table 5.3-2	Significance of Findings Based on Participant Experience	57

EXECUTIVE SUMMARY

This report presents the results of knowledge base certification activity of the Expert System (ES) Verification and Validation (V&V) Guideline Development project which is jointly funded by the U.S. Nuclear Regulatory Commission (USNRC) and the Electric Power Research Institute (EPRI). This activity encompasses the development and testing by experiment of ES Knowledge Base certification methods. Certification, as used here, means the determination by various static analysis means of the quality of the Knowledge Base. Such determination can assure the quality with respect to most faults, but full dynamic system testing is required for total assurance. The Volume 2 report for this project identified the Knowledge Base as the one component of expert systems which could not be adequately certified by current conventional software V&V methods. The Knowledge Base contains detailed subject matter information which can be in the form of "IF-THEN" rules, hierarchical "frames" or classes of associated attribute-value information, or simple facts.

This is the first known behavioral evaluation of the V&V of any type of software system. While empirical studies of programming have long been conducted (e.g., Miller, 1974, 1975), the equally complex tasks of testing, verifying, and validating software systems are seldom evaluated. The key contribution of the present behavioral study is that it provides empirical evidence concerning whether plausible knowledge base certification methods truly are effective in helping people find problems with knowledge bases. In addition, the experimental methodology developed here to study these particular tools is generic and could be used to address a number of other related types of tools and software issues.

The Knowledge Base certification method experiment used two real nuclear expert systems: a boiling water reactor emergency operating procedures tracking system and a pressurized water reactor safety assessment system, which were denoted as Systems A and B, respectively. The experiment was conducted at SAIC's offices in McLean, Virginia on September 22 through 24, 1992 and included 20 participants from nuclear power plant operating utilities (Baltimore Gas and Electric Company, Duke Power Company, and Yankee Atomic Electric Company), the USNRC Technical Training Center, University of Maryland Graduate Center, EG&G Idaho, Inc., and SAIC.

Ten of the participants were assigned to use and evaluate each of the two expert systems. Each of these groups was further divided into two groups of five participants. For each expert system, one group of five performed as a control group and another group of five was the experimental group using new methods and tools. The four groups of five participants were composed to include the same mix of utility and non-utility participants with a range of applicable nuclear and testing experience. Different sets of designed errors of varying type and difficulty of detection were seeded into the two expert systems for use in the separate evaluation tasks.

The experiment involved the completion of four separate tasks in Knowledge Base certification: requirements grouping, requirements tracing, syntax checking, and knowledge checking. These are all static analysis techniques which do not involve actual execution of the expert system code, but analytically examine the knowledge base contents from different perspectives. These particular methods were selected as being the best and most appropriate for study from over a hundred possible V&V techniques, using the selection criteria and methodology defined in Activity 1. A set of new automatable tools were used by the experimental group for each of the four tasks. Three of the four tools were designed as part of the project for the express purpose of certifying the Knowledge Base of expert systems. Although these three tools themselves were not implemented, they were designed, and their output for the error-seeded knowledge

bases were simulated for use by the participants. The fourth tool was an existing automated tool for syntax checking that is still experimental. The control group used traditional inspection (desk-checking) to perform the same four tasks. Each participant received basic training in expert systems, V&V, and Knowledge Base error detection as well as training specific to their particular expert system. In the experiment each participant filled out response sheets identifying the location of errors found, error classification, the confidence of these judgments, and the time required.

It is important to note that this study is an investigation of the efficacy of the four selected individual static analysis tools used singly. In actual Knowledge Base Certification, multiple methods would be employed sequentially to test for faults. (e.g., cf to Guideline V&V Packages G, H, and I in Section 2, Volume 7 of this report). Also, note that this study is not focused on evaluating the specific systems. Only one component, the knowledge base, was examined (exclusive of the inference engine, interfaces, plant state messages, etc.) and actually only a portion of this component was examined by each tool.

Results of the experiment were statistically analyzed using Analysis-Of-Variance (ANOVA) statistical tests with 24 dependent variables (including total number of correct responses, total time, and number of false alarms). The major results are delineated below:

- 1) The experimental group (with tools) outperformed the control group by a wide margin in all areas of the experiment (e.g., number of errors discovered, false alarms, and time to complete).
- 2) For syntax checking and knowledge checking, performance for the group using the knowledge checking tool was superior to that for the group using the syntax checking tool.
- 3) The System B group outperformed the System A group primarily because System B is less complex than System A.
- 4) Participants with low (0 to 4 years) nuclear experience performed as well as those with high (7 to 22 years) experience. Similarly, participants with low or no testing experience (self ratings of little or none) generally performed as well as those with high experience (self-ratings of moderate to extensive). These findings suggest that the amount of subject matter or testing experience had no significant effect on the ability of participants to detect errors.
- 5) A few seeded errors were either not found or the rule containing the error was identified but a completely incorrect diagnosis of the problem was given.
- 6) The method selection technique developed in Activity 1 was quite applicable in selecting Activity 6 methods to test.

The following conclusions are drawn from these results:

- 1) The use of tools or aids for use in static knowledge base certification results in a significant improvement in detecting all types of defects (including hard-to-find problems), avoiding false alarms, and completing this effort in less time.
- 2) Of the four tools developed and used in this KB certification experiment, knowledge checking and requirements grouping showed the greatest improvement in performance.
- 3) The syntax and knowledge checking tools led to the detection of the majority of errors in the rule bases (66% for System A, 83% for System B). Static testing methods, therefore, offer a highly effective -- and lower cost -- initial alternative to the traditionally-preferred dynamic testing methods which involve actual execution of system rules with "live" data and experienced professionals as observers judge performance accuracy ("oracles").
- 4) Even with the tools and aids provided, some hard "seeded" errors were still not detected by anyone. Other methods or tools are clearly required for KB certification, such as indicated in Packages G, H, and I in Section 2, Volume 7 of this report. However, over 85%, on average, of the rule base errors were judged detectable by the syntax and knowledge checking tools. Even a little increase in the present one-hour tool training could plausibly be expected to increase error detection significantly from the above levels.
- 5) Testing experience and extensive (7 to 22 years) subject matter experience did not improve the performance of the KB certification participants, as compared with less (2 to 4 years) subject matter experience. (It should be noted, however, that 95% of the participants had a college degree in a technical field.) Were this finding to be verified in an experiment with a larger number of participants, this discovery could be capitalized on by the nuclear industry to lower the costs of performing V&V on expert systems.

On the basis of the present findings, the recommendation is made that tools comparable in function to those tested here be strongly considered to be among those ultimately selected for certification of expert systems knowledge bases.

1 INTRODUCTION

This report is one of a series of eight individual reports for Science Application International Corporation's (SAIC's) 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 Electric Power Research Institute (EPRI) and the United States Nuclear Regulatory Commission (USNRC).

1.1 Purpose and Scope of the Knowledge Base Certification Activity

The purpose of Activity 6, as described in the statement of work, was to "Conduct an experiment to analyze the effectiveness of the knowledge certification method against an unstructured design ... review by experts, using ... [existing expert system] knowledge bases as examples. Add various errors to the knowledge bases. Using two teams of experts... evaluate the knowledge bases... Compare and document the performance of the evaluation methods."

As specified in the statement of work, two existing expert systems were used as testbeds for testing the knowledge base certification methods. The first is System A and was developed by Capri Technologies, Incorporated and General Electric Corporation for EPRI. This system is designed for use by boiling water reactor (BWR) operators, and facilitates the monitoring and carrying out of emergency operating procedures (EOPs) during abnormal reactor events and accidents. A generic version of System A is being used. This version was re-created specifically for the experiment. The second is System B and was developed by EG&G Idaho, Incorporated under contract to the USNRC for use in the USNRC Emergency Operations Center. The current version of System B pertains to pressurized water reactors (PWR), and assists the reactor safety team at the USNRC Emergency Operations Center in monitoring and projecting plant status during an emergency at a licensed nuclear power plant.

It should be noted that the object of this task was not to evaluate these systems. Rather, the systems were used to test the effectiveness of the V&V methods. Accordingly, only a portion of the systems was used -- that which corresponds to the knowledge base, excluding interfaces, the inference-engine, messages, etc. In order to permit the full testing of the V&V methods, a variety of errors were deliberately added to each system.

The testing of the effectiveness of the V&V methods was done for single methods, determining what percentage of the errors were caught by each, alone. In actual practice, a set of multiple methods would be employed sequentially so that faults not caught by one method could be caught by other methods.

1.2 Relationship to Other Project Activities

Figure 1.2-1 shows the relationship of Activity 6 to the other tasks in the project. As shown, Activity 6's results feed into Activity 4 and Activity 7. As a prelude to Activity 4, SAIC developed three different Knowledge-Base Certification methods as part of Activity 6. These methods were tested for both System A and B in this study.

One or more of these methods will likely become part of the recommended Activity 4 Verification Methodology.

Figure 1.2-1 TASK DEPENDENCIES AND INTERRELATIONSHIPS

1.3 Report Organization

Section 2 describes the selection of the three knowledge base certification methods. Section 3 discusses the experimental design and error seeding. Section 4 describes the actual three day experiment. Section 5 presents a statistical analysis of the results of the experiment, and Section 6 discusses the conclusions and recommendations that can be made based on these results.

2 SELECTION OF KNOWLEDGE BASE CERTIFICATION METHODS

2.1 Identification of General Selection Issues

Software certification involves specification of processes for evaluating the quality of an implemented software system with respect to particular evaluative criteria and testing environments. The goal of a knowledge base certification methodology is to specify the processes for determining the adequacy and accuracy of the completed knowledge base component of an expert system. In deciding which testing methods should be evaluated in the present experiment, two general issues were first considered.

The first major issue is whether certification of the knowledge base component could be sufficiently accomplished by static testing means or whether it requires dynamic testing and to what degree. The knowledge base constitutes the major software component which is specific to expert system applications. It can be extensively evaluated alone using analytical techniques. However, such static testing cannot reveal certain types of faults which depend on an interaction with the system's inference engine component and can be revealed only during dynamic testing. The relative merits of using static methods alone vs. using both static and dynamic need to be examined.

The second major issue is the problem of deciding on the criteria and process for determining which of the many V&V techniques, whether static alone or both static and dynamic, should be prime candidates for inclusion in an experiment evaluating knowledge base certification procedures.

These two issues are discussed in the following section. The rationale for actual selection of a few techniques to test experimentally is given in Section 2.3.

2.2 Resolution of Issues

Static vs. Dynamic Testing Methods. In the Activity 1 review of conventional V&V methods (Miller et al, 1993), static methods in general were judged cheaper, more practical to accomplish, and more amenable to automated testing than dynamic methods. These conclusions continue to be borne out in practice (e.g., Humphrey, 1989; McGarry, 1992). Since static methods are applicable to the knowledge base, more so than to any other expert system component, it is reasonable to emphasize this type for knowledge certification.

Use of static testing methods appears to be a reasonable and practical alternative when specific expert systems for the nuclear industry are considered (for a discussion of almost 300 of these, cf. Bernard & Washio, 1990). Many of these expert systems are directly interfaced to data channels supplying information and updates about the status of plant variables. Dynamic testing of these systems is greatly complicated because of the frequent unavailability of simulators or drivers to generate realistic data input for a desired test set. Knowledge base certification involving any extensive dynamic testing would be impractical for such systems. A reasonable approach to this problem is to undertake dynamic testing only when all possible knowledge base errors have been removed by static testing certification. For example, the static testing approach may involve running a tool which will identify errors. These errors are then corrected, and the tool is run again (a fix to the knowledge base could reveal further flaws). As this cycle continues, there will be a point in time when the tool reports no errors. At that point dynamic testing could commence.

There are two major considerations which further complicate dynamic testing of knowledge bases. First, one requires a perfect "oracle" to be present at all times during the tests to ensure the accurate and timely detection of errors during test executions. Second, one would have to generate enough test cases, covering all the precursor conditions, to ensure that a particular state would be encountered. This is very difficult to estimate, but could easily be on the order of hundreds of test cases per rule, if there is any extensive chaining between rules (and, therefore, to totals of tens of thousands of test cases for larger systems with 100 or more rules).

Given that static testing is preferable on practical grounds, the remaining concern is with certain types of knowledge base errors which can be detected only when exercised by the inference engine. This concern can be responded to in two ways. First, many of the known interactive problems derive from the inference engine firing the rules in a particular order, which is influenced by the way the knowledge base is structured. Such problems could be largely made detectable by static analyses which take into account the knowledge base structure and the inference engine's prioritization mechanism to determine if such things as two different rule orders for the exact same conditions could, analytically, lead to two different states. Second, and much more important, actual dynamic testing of the knowledge base solely with the inference engine and not with the data and other interfaces is very difficult to achieve. When all of the components of the expert system are exercised in a dynamic test, ostensibly as part of a knowledge base testing, this situation is indistinguishable from overall system-level validation, something that must be accomplished in any event. Thus, any remaining knowledge base errors, detectable only by dynamic testing, are reasonably deferred to the system validation stage of V&V. At this time, it would be very reasonable to determine sensitivity of the processing to the ordering of the rules by picking one or two well-understood test cases and permitting the rules in no more than a couple of ways (e.g. reverse order and a random-order.) If the outcomes are different for the different rule-orders, this should certainly be evaluated by the developers for significance. Given rule permutation testing, restriction of knowledge base certification to just static tests would therefore appear reasonable.

Criteria for selecting specific methods. In the review of conventional V&V methods (Volume 2), individual techniques were rated on eight factors, four concerning the power of the technique, and four concerning its ease-of-use¹. Two different metrics were then developed, based in different ways on these eight factors: a cost-benefit metric, and an C effectiveness metric. The latter was computed for three classes of V&V, based on combined estimates of required system integrity and system complexity. Class 3 involves the lowest level of V&V, for simpler systems with minimum required integrity, while Class 1 involves the most stringent V&V, for the most complex, highest required integrity systems. The effectiveness metric was computed using a different set of weights for the eight factors, based on a judgment of their differential importance as complexity and required integrity increases. The techniques were then separately ranked for each of the three effectiveness levels as well as for the cost-benefit metric. The top ranked of the 58 conventional static testing techniques provides one source of potential certification methods.

¹ Ratings were subjective 5-point scale judgments (with 5 being the highest ranking), based on a wide reading of the literature on conventional V&V methods. The four power factors were: (1) Broad Power -- the number of different defects detectable by the technique; (2) Hard Power -- a judgment about the capability of the technique to detect hard, difficult-to-find, problems; (3) Formalizability -- the extent to which a technique can lead to formal calculus or algebraic representations of the system, such that automated theorem-provers could be developed to implement the method; and (4) HCI Testability -- the extent to which the technique directly tests the Human-Computer Interface. The four ease-of-use factors were: (1) Ease of Mastery -- the ease with which a technique can be taught, understood, and applied; (2) Ease of Setup -- the labor and resources required to have the V&V technique ready to be applied to the program; (3) Ease of Running or Interpretation -- the ease of actually applying the technique and interpreting the findings; and (4) Usage -- the extent to which the technique is generally and commonly used.

A second source of techniques are the non-conventional static testing techniques discovered in the review of V&V techniques used for expert systems from this project's Volume 3 (Groundwater et al., 1991). Of the total of four new non-conventional static testing techniques identified, the most promising by far were "syntax checking" and "semantics checking"; the latter is called here knowledge checking. The former involves conversion of the knowledge, particularly the rule base, into a formal structure (e.g., graph, table, and matrix) to search for various kinds of defined syntactic anomalies. Knowledge checking involves use of various kinds of knowledge external to the knowledge base (particularly, knowledge of constraints) to check the semantic consistency or accuracy of the knowledge contained within the knowledge base. To permit comparison of these two new methods with the conventional ones, they were both rated on the four power and four ease-of-use factors developed for the conventional methods (discussed above). The cost-benefits metric and two effectiveness metrics were then computed (one for each of the two extreme V&V classes, Level 1 and Level 3), and these scores were then entered into the rankings of conventional methods.

It should be noted that both Systems A and B require Class 2 V&V (as discussed in Section 5 of Volume 1), based on their intermediate complexity and required integrity. However, to ensure that candidate methods examined covered the range of complexity and required integrity, an average of the two extreme V&V levels (1 and 3) was used during selection. This is in keeping with the task objective which is to test the methods, not the systems.

To obtain a reasonably-sized sample of the best techniques from which to choose, the top ten ranked techniques for each of the three metrics were identified; these are shown in the first four columns of Table 2.2-1. A total of only 15 methods account for the top ten scores in the three metrics, indicating considerable correlation between metrics. The top ten of these 15 were then determined by summing the rankings for each of the three metrics and ranking these from low to high. This result is shown in the fifth column of Table 2.2-1. These overall top ten methods provide a reasonable set from which to select methods to constitute those for the certification process and therefore those to be evaluated experimentally.

2.3 Selection of the Methods To Be Experimentally Tested

Of the top ten static V&V methods identified in Table 2.2-1 there is one which can be immediately identified as having limited utility as a V&V procedure for expert system knowledge bases. This method is Data Interface Inspection: knowledge bases typically contain very few aspects of direct interaction with data interfaces (these are typically accomplished via calls to special data-access routines programmed in conventional software languages). To a somewhat lesser extent, User Interface Inspection is similarly ill-satisfied by inspection of the knowledge base component alone, since all of the screen handling, most of the message presentation decisions, and much of the information context is accomplished by expert system components other than the knowledge base. For the remaining eight top-rated methods, there is one final selection consideration: the ease with which a method could be behaviorally evaluated. There are four that present difficulties against this criterion. These involve groups of people having specialized knowledge of the system being evaluated and representing special interest groups. Structured Walkthroughs typically involve presentations by the developers of code elements to their peers. Formal Customer Reviews involve presentations by system developers to a group who have commissioned the system. Peer Code Checking essentially involves system developers swapping their work for examination. Finally, Informed Panel Inspection usually involves a senior group of quality-conscious developers with special knowledge of either the potential implementation problems with systems of this type or the difficulties of achieving the application functionality.

Table 2.2-1 Rank-ordering of the top 10 rated static-testing techniques for the cost-benefit metric and the extreme effectiveness metrics¹

Static Testing Techniques	Cost-Benefit Metric Ranking	Effectiveness Metric Ranking		Ranking Overall Three Metrics ²
		Class 3.	Class 1.	
Syntax Checking	2	2	2	1
Semantic Checking	1	10	1	2
Structured Walkthroughs	36	1	10	3
Requirements Tracing	6	5	6	4
User Interface Inspection	6	9	7.5	5
Formal Customer Review	6	3	16	6.5
Peer Code Checking	6	4	15	6.5
Data Interface Inspection	9.5	8	14	8
Informed Panel Inspection	9.5	6	18.5	9
Desk Checking	6	7	22	10
Clean Room	17	16	3	11
Process Trigger/Timing Analysis	21	30	4	12
Failure Mode Effects Causality Analysis	26	21	9	13
Program Proving	31	42	5	14
Trace Assertion	40	48	7.5	15

¹ This was the ranking used to select the methods. However, a different method of ranking techniques was developed for the final report, and revised in the Volume 2 survey of V&V techniques.

² Based on the sums of rankings for the three metrics, ordered from lowest to highest.

for such approaches simply would not be available, particularly to staff the multiple experimental and control conditions which are needed for good experimental design.

Four methods remain: knowledge (semantic) checking, syntax checking, desk checking, and requirements tracing. All could be evaluated relatively easily, given the availability of automated tools for the first two. They all appear to be generally desirable for inclusion as part of a certification procedure².

Both knowledge checking and syntax checking address the correctness and accuracy of the knowledge base and would also appear to be highly desirable for a certification procedure. Checking of constraints and "meanings" will reveal invalid knowledge. Checking for violations of rule-writing style within a rule or testing for anomalies between rules (such as circularity or contradiction) identifies accuracy problems involved with the composition and interaction of knowledge elements. Both methods would contribute greatly to the certification procedure.

Desk checking, the careful human examination of a program for problems, is also a powerful and desirable technique, particularly when guided by precepts of what to look for and how.

Requirements tracing constitutes tracing forward from each unique requirement element to, where possible, design elements and then to the elements of the implementation. Requirements tracing is a mandatory procedure to investigate the completeness of the knowledge base with respect to the requirements that were established at the start of the project. It will help determine whether the individual requirement and design elements are satisfied (testing of other components and the system as a whole is also necessary for this determination). It will also expose "unintended functions", aspects of the system which may have been added in for various good (or not so good) reasons but which are nowhere motivated by a requirements or design element.³

Based on the above arguments, these four methods were chosen for experimental test.

Please note that this study is an investigation of the efficacy of the four selected individual static analysis tools used singly. In actual Knowledge Base Certification, multiple methods would be employed sequentially to test for faults. (e.g., cf to Guideline V&V Packages G, H, and I in Section 2, Volume 7 of this report.)

2.4 Determining the Experimental Form of the Four Selected Methods

Having selected four specific methods, there still remained the problem of deciding on the actual form of these methods for experimental use. An automated requirements tracing tool was available, but was not tailored for expert systems. The decision was made to simulate the output of such a requirements grouping and tracing tool. While an automated syntax checking tool, VERITE', was available, a fully-featured knowledge checking tool was not. Since use of an automated knowledge checking capability is so desirable as part of the certification package, the decision was made to simulate the output of a knowledge checking tool.

² These are not yet final decisions, since -- even with a favorable outcome from the present experiment -- the results of three tasks yet to be completed will also contribute to the V&V guidelines decisions.

³ It should be noted that requirements tracing depends on requirements being previously analyzed and grouped, and that these precursor activities can also be studied separately.

Desk Checking provides a convenient and appropriate control condition for each of the actual and simulated tools. With this procedure, the best possible hand procedures for requirements tracing, syntax checking, and knowledge checking were taught and evaluated. The detailed considerations determining the experimental form of the four techniques are given below.

2.4.1 Requirements Tracing

It has been estimated that a problem identified during testing costs 20 - 40 times more to correct than a problem detected during the preparation of requirements documents (Lewis, 1987). One of the most costly errors to correct is the omission of requirements from a completed system. Basili and Weiss found that thirty-one percent of the requirements errors detected on a project were incomplete (missing) requirements (Basili & Weiss, 1981). Such errors, discovered during testing, necessitate the update of many requirements and operational documents (e.g., requirements specification, design documents, and user's manual) and the development, integration, and testing of additional source code. Requirements tracing, a technique which detects requirement omissions, incomplete or incorrect requirement satisfaction, and unintended functions, greatly reduces cost, schedule, and technical risks to a software development effort (Hayes, 1990).

The requirements tracing method selected for this experiment is based on an existing internal tool called SuperTrace (described in Hayes, 1990). SuperTrace is written in SuperCard and runs on an Apple Macintosh II. SuperTrace provides a number of automated and interactive utilities which facilitate requirements tracing. This tool was developed by the Command Systems Division of SAIC for the U.S. Navy, and is being used by permission of the Navy. To facilitate the entry of requirements into SuperTrace, the SuperTrace Front End Processor (SFEP) was utilized. SFEP is written in TURBO Pascal and C++ and runs on an IBM PC or compatible. SFEP marks and extracts requirements from a document by searching for key words and phrases such as "shall" and "must." It writes the extracted requirements to a file which is then imported into SuperTrace.

2.4.1.1 Grouping of Requirements

The requirements tracing portion of the experiment was broken into two portions: requirements grouping and requirements tracing. Requirements grouping refers to the identification of requirements which belong to a common group or category. An example would be identifying all the requirements pertaining to "software" (or hardware, or knowledge). SuperTrace was used to group requirements based on various topics (e.g., "logic", "Heat Sink Control" (System B), and "RPV Flooding" (System A)). In SFEP, the user can specify categories for requirements (such as software, hardware, or hardware and software). SFEP will then assign these keywords or categories to the requirements as they are extracted (a synonym list can be specified also; for example, if the words "code", "source code", or "software" are found, assign the category "software"). Once the requirements and associated keywords are imported into SuperTrace, printouts can be made based on these categories (see Figure 2.4.1.1-1). These printouts were used by the experimental group. The control group manually grouped the requirements, reading the text of each requirement and deciding whether or not it was part of the category.

2.4.1.2 Tracing of Requirements

Requirements tracing involved the requirements, the design, and the knowledge base of each of the two systems. For both systems, the Functional Requirements Document was the source of the requirements. In the case of

E10098	4.3	STEAM COOLING
The purpose of this procedure is to cool the reactor core with steam sufficiently to maintain peak cladding temperature below 2000F.		
E10099	4.3	STEAM COOLING
This procedure is entered from KNPS 500.7 level restoration step iv.d-3-a at page 6 of 7 when following condition exists: ***** NOTE*****TAF=0 cm as indicated on the following fuel zone instrumentation: Panel 1C01-RHR B/C Section - Reactor Water Level Indicator (B21-R610) Panel 1C01-RHR A Section - Reactor Water Level Recorder (B21-R615)		
E10100	4.3	STEAM COOLING
RPV water level has decreased to TAF AND no system injection subsystem or alternate injection subsystem is lined up with at least one pump running.		
E10101	4.3	STEAM COOLING
A. Line up injection system injection subsystem or alternate injection subsystem with at least one pump is running.		
E10102	4.3	STEAM COOLING
IF while executing the following steps Emergency RPV depressurizations required OR any system injection subsystem or alternate injection subsystem is lined up for injection with at least one pump running ENTER 500.9 Emergency RPV Depressurization at step III.A page 5 of 7.		
E10103	4.3	STEAM COOLING
B. WHEN RPV water level cannot be determined OR drops to minimum Zero Injection RPV Water Level - 264cm on the full zone instrumentation B21-R610/B21-R615.-----1. Verify that suppression pool water level is above 145 cm (elevation of top of SRV discharge device) AND -----2. Open one SRV.		
E10104	4.3	STEAM COOLING
C. WHEN RPV pressure decreases to 49.3kg/cm2 (700 psig)(Minimum signal SRV Steam Cooling Pressure) THEN Enter KNPS 500.9 Emergency RPV Depressurization at step III-A page 5 of 7.		

Figure 2.4.1.1-1 Sample Grouped Requirements Printout

System A, the Emergency Operating Procedure messages (EOP messages) constituted the design information. For System B, the graphs (or trees) supplied the design information. For both systems, the rules (knowledge base) constituted the actual system (lowest level of the trace).

SuperTrace was used to trace the requirements to the design and the design to the rules for both systems. As a result of this process, a printout was made that contained the requirement text and information on the design and rules that satisfied the requirement for every requirement in both systems. This printout was then enhanced slightly to produce requirements traceability "maps" (see Figure 2.4.1.2-1). It should be noted that SuperTrace could easily be modified to generate this enhanced printout. These maps were broken into several parts: information pertaining to the requirement (e.g., ID, text, and section number); information on the traceability of the requirement (design, rules); the actual design data (EOP messages or graphs) pertaining to the requirement; and the appropriate rules for the requirement. These "maps" were used by the experimental group. The control group manually traced the requirements, using keywords and phrases in the requirement text to find the appropriate design information and rules.

2.4.2 Syntax Checking

The syntax-checking tool VERITE' was made available for experimental use by its developer, Dr. Alun Preece of Concordia University.⁴

VERITE' is a recent offshoot of a previous tool, COVER Version 3, based on Alun Preece's dissertation and tested in connection with commercial utility expert systems (cf. Preece, 1990). VERITE' is based on first-order logic and is designed to take as input the facts and rules of a knowledge base and produce as output a description of any anomalies that were detected in the knowledge base.

Examination of the rule knowledge bases by VERITE' required writing special rule-parsing front-ends for both expert systems which permitted the tool to identify and interpret all of the IF and THEN clauses which made up each rule. This is something that would normally be required the first time any automated tool was applied to a new rule syntax. This activity was somewhat complicated by the variety of legitimate IF clause forms that appear in the System A rules. These include: (1) what are essentially Boolean, true/false, clauses such as "(RCIC TURBINE IS RUNNING)"; (2) functions returning true/false values such as "(QUALITY(REACTOR POWER, BAD))"; and (3) full conditional statements such as "(BORON WEIGHT INJECTED > HOT SHUTDOWN BORON WEIGHT CONSTANT)".

Once all rules were pre-processed into recognizable formats, VERITE' searches for the following kinds of anomalies:

- Conflicting (contradictory) rules,
- Redundant rules,
- Rules which are subsumed by other rules,
- Cyclic inference chains (e.g., Rule A causes Rule B to fire, which causes Rule A to fire),
- Useless rules (rules which can never be fired),

⁴This tool is not yet available for public use.

ID:	E10128								
Type:	Specific								
Keywords:	control rod, SRV, RPV								
Focus:	<p>B. IF any control rod is not inserted to or beyond the maximum subcritical banked withdraw position 06 THEN</p> <p>-----3. Commence and slowly increase injection into the RPV with the following systems until at least 2 SRV's are open and RPV pressure is above the Minimum Alternate RPV Flooding Pressure.</p> <p>-----a. Condensate Pumps/feedwater system</p> <p>-----b. CRD</p> <p>-----c. LPCI</p>								
Context:	4.5 RPV FLOODING - Operator Actions								
Forward Links:	<table> <tr> <td>Emergency Operating Instruction:</td><td>RPV Flooding (C_8)</td></tr> <tr> <td>EOP Messages:</td><td>8 III.B.3</td></tr> <tr> <td>Rules:</td><td>C_8_B 2.1</td></tr> </table>	Emergency Operating Instruction:	RPV Flooding (C_8)	EOP Messages:	8 III.B.3	Rules:	C_8_B 2.1		
Emergency Operating Instruction:	RPV Flooding (C_8)								
EOP Messages:	8 III.B.3								
Rules:	C_8_B 2.1								
For requirements of type "specific", the relevant EOP Messages and their Rules:									
EOP Message 8 III.B.3	<p>SLOWLY inj until ≥ 2 SRV's open AND RPV pr > MAFP</p> <p>(MORE)</p> <p>Slowly increase injection into the RPV with the following systems until at least 2 SRV's are open and RPV pressure is above the Minimum Alternate RPV flooding pressure.</p> <p>a. Condensate Pumps/feedwater system</p> <p>b. CRD</p> <p>c. LPCI</p>								
Rule C_8_B 2.1	<table> <tr> <td>IF</td><td>(SRV NUMBER_OPEN < min FLOOD SRV's OPEN CONSTANT)</td></tr> <tr> <td></td><td>(RPV PRESSURE IS BELOW min ALTERNATE RPV FLOODING PRESSURE)</td></tr> <tr> <td>THEN</td><td>(SET 8 III.B.3)</td></tr> <tr> <td></td><td>(SET CAUTION 8) /* CORE DAMAGE</td></tr> </table>	IF	(SRV NUMBER_OPEN < min FLOOD SRV's OPEN CONSTANT)		(RPV PRESSURE IS BELOW min ALTERNATE RPV FLOODING PRESSURE)	THEN	(SET 8 III.B.3)		(SET CAUTION 8) /* CORE DAMAGE
IF	(SRV NUMBER_OPEN < min FLOOD SRV's OPEN CONSTANT)								
	(RPV PRESSURE IS BELOW min ALTERNATE RPV FLOODING PRESSURE)								
THEN	(SET 8 III.B.3)								
	(SET CAUTION 8) /* CORE DAMAGE								

Figure 2.4.1.2-1 Sample Requirements, Traceability Map

- Dead-end rules (rules with THEN Actions which never facilitate any other rule),
- Unsatisfiable conditions (a test for a variable-value which could never occur),
- Missing values (values of a variable that could be assigned, logically, but never actually occur in any rule),
- Missing rules (the situation in which some rule is needed to satisfy a requirement of the system, but no existing rules fulfill that function), and
- Fundamental syntax errors (e.g., missing definitions and clashes between variable types).

The output of VERITE' consisted of error reports for every anomaly or problem detected. The reports provided a short description of the problems found and the identifiers of the rule(s) involved in the problem. When only one rule was thought by VERITE' to be causing the problem, the specific IF or THEN clause was identified with an arrow pointing to the culprit clause. A complete listing was provided of the complete rules involved in the problem. Examples of VERITE' output for the System A rules are shown in Figure 2.4.2-1. Examples for the System B rules are shown in Figure 2.4.2-2.

As can be seen from the figures, VERITE' was often able to identify clauses containing misspellings or otherwise unknown terms and therefore restrict the locus of the report to a single rule. In many cases, however, an error in a rule will show up as an error-type involving multiple rules. For example, an error in an IF clause, say in Rule 34, could cause VERITE' to discover that two rules, say Rule 34 and Rule 196, are contradictory. However, VERITE' has no way of deciding whether Rule 34 or Rule 196 is the problem rule, nor what that problem might actually be. Thus, VERITE' reports on the symptoms of errors that actually occur within rules to the extent they involve anomalous interactions between rules. Since VERITE' is reporting on symptoms, the actual error description provided often is not of great use in pinpointing the specific problem.

For 11 of the 21 rule errors purposefully introduced into System A, VERITE' was able to identify either the single rule containing an error or a set of rules which included the error-containing one. For System B, within which there was no possibility of rule-chaining, VERITE' identified all 11 purposefully introduced rule errors. The empirical question is whether the experimental participants can utilize the VERITE' output effectively, and whether their performance is improved over participants using the desk-checking method.

2.4.3 Knowledge Checking

The primary knowledge checking capability that appeared so promising in the review of expert systems V&V (Volume 3) is an aspect of the experimental DARPA Expert Systems Validation Associate ((D)EVA) tool developed by Stachowitz and others (1987). However, this tool was not available for use in this experiment. The knowledge checking capability of an even more powerful tool was simulated, based on an encoding of specific, readily-obtained, engineering knowledge about the two expert systems into a database and assuming a very straight-forward automated capability for accessing this database in the evaluation of each of the systems' rules. Specifically, the knowledge checking processing was performed manually, and the results were formatted to provide clear informative output.

Two types of engineering information were developed as the basis for this knowledge checking tool. First, all of the plant variables used in any way in the expert systems were described in terms of the sub-systems they were associated with, the units of their measurement, their allowable range of values, and the reference setpoints or values to

<p>1. CYCLIC INFERENCE CHAIN</p> <p>RULE RPV_ENTRY_7</p> <p>IF (ROV ENTRY)</p> <p>THEN (SET REACTOR SCRAM IS REQUIRED) /*JUST IN CASE (ENTER COMPOSED_POINTS, CAUTIONS, GENERAL_RULES, RC/L, RPV_CONTROL)</p> <p>RULE RPV_ENTRY_4</p> <p>IF (REACTOR POWER IS ABOVE APRM DOWNSCALE TRIP SetPOINT) ;/*5% (REACTOR SCRAM IS REQUIRED)</p> <p>THEN (SET RPV ENTRY)</p>	<p>2. UNSATISFIABLE CONDITION in Rule PC/HG 2.3 (RAD RELEASE RATE IS BENEATH LCO RAD RELEASE RATE)</p> <p>RULE PC/HG_2.3</p> <p>IF (NOT(PC/P-8)) ;/*NOT VENTING (PC H2 CONC IS ABOVE min DETECTABLE H2 CONC) PC (NOT(H2 RECOMBINER max OPERATING LIMIT HAS BEEN EXCEEDED)) ;/* 0.5 (NOT(H2 DEFLAG LIMIT HAS BEN EXCEEDED)) ;/* A SAVE (RAD RELEASE RATE IS BENEATH LOC RAD RELEASE RATE) ;/* A SAVE</p> <p>==> THEN (SET 4 III.F.1a) ;/* VENT AND PURGE (SET CAUTION 6) ;/* DEFEAT ISOL INTERLOCKS</p>
<p>MESSAGES: 4 III.F.1.a</p> <p>SAMPLE the PC for hydrogen - use KNPS 807.2 (see MORE)</p> <p>The hydrogen monitor in the primary containment has failed or is not operational</p> <p>~</p> <p>Sample the primary containment for hydrogen using KNPS 807.2, then follow the written hydrogen control procedures.</p> <p>CAUTION-6 DEFEAT ISOLATION INTERLOCKS</p> <p>Defeating isolation interlocks may be required.</p>	
<p>3. SUBSUMED RULE PAIR</p> <p>RULE RPV_ENTRY_4</p> <p>IF (REACTOR POWER IS ABOVE APRM DOWNSCALE TRIP SetPOINT) ;/*5% (REACTOR SCRAM IS REQUIRED)</p> <p>THEN (SET RPV ENTRY)</p> <p>RULE RPV_ENTRY_6</p> <p>IF (REACTOR SCRAM IS REQUIRED)</p> <p>THEN (SET RPV ENTRY)</p>	

Figure 2.4.2-1 A Subset of VERITE' Output for SYSTEM A

4. UNUSED CONSEQUENT in Rule COMPOSED_POINTS_7
(SET STEAM LINE BREAK = INDICATED)

RULE COMPOSED_POINTS_7

IF

(MAIN STEAM LINE LEAK DETECTION HAS TRIPPED)

THEN

====>

(SET STEAM LINE BREAK = INDICATED)

;/* A SAVE VARIABLE

Figure 2.4.2-1 A Subset of VERITE' Output for SYSTEM A

1.	<p>UNSATISFIABLE CONDITION in Rule 060 rate of [Temp Hott Leg A] <=0</p> <p>RULE 060</p> <p>=====> if rate of {Temp Hott Leg A} <= 0 then HotLegTempDecreasingLoopA hasValue normal</p>
2.	<p>CONFLICTING RULES: Rule 222 and Rule 223</p> <p>RULE 222 if [RCS Forced Flow?] is yes then RCSFlow>Min hasVlaue normal</p> <p>RULE 223 if [RCS Forced Flow?] is maybe then RCSFlow?Min hasValue degraded</p> <p> ;rules for node ThA-TcA>10</p> <p> From permissible input: [RCS Forced Flow?] is yes [RCS Forced Flow?] is maybe Incompatible conclusions would be inferred: RCSFlow>Min hasValue normal RCSFlow>Min hasValue degraded</p>
3.	<p>UNUSED CONSEQUENT in Rule 239x2 ThA2-TcA2<50 hasValue somewhat_degraded</p> <p>=====> if [Temp Hot Leg A] - [Temp Cold Leg-A2] > 50 then ThA2-Tc2<50 hasValue somewhat_degraded</p> <p> ;rules for node DelatTCoreThotLoopA2</p>

Figure 2.4.2-2 A Subset of VERITE' Output for System B

which they could legitimately be compared. Descriptions and units were also provided for these references. Second, a list of all legitimate comparisons between variables and references were developed. These comparisons included the actual operators which the expert system designer had decided were appropriate (e.g., expressing the notion of a variable exceeding a reference by the text string "greater than" rather than using the equivalent symbol ">"). In addition, the implication of every particular test or comparison being TRUE (e.g., "ALL RODS ARE INSERTED AT OR BEYOND 06") was evaluated in terms of the impending consequences of that state, using the following set of descriptors: Normal, Improving, Degrading, Serious, and Catastrophic.

All of the above engineering information was obtained for the experiment from nuclear engineers familiar with the BWR (Boiling Water Reactor) emergency operating procedures and the PWR (Pressurized Water Reactor) system safety and availability states. In practice such information would be gathered in exactly the same way by consulting knowledgeable engineers. The information specific to the expert system, such as the preferred form of a comparison (text vs. symbol) for a particular variable-reference evaluation, was obtained by general inspection of the whole set of expert system rules. In practice, this information could be rapidly and easily obtained in a discussion with one of the expert system developers.

While the gathering of the engineering information took about two weeks of engineering personnel time to complete and enter in a spread-sheet, most of this was that of a junior engineer, and the amount of time is minor relative to the overall multi-year development efforts involved in these systems. To the extent that such knowledge can lead to the detection of difficult software problems, unidentifiable by other means, and can prevent operational failures, this small amount of time must be considered well cost-justified.

The utility of this engineering knowledge, when properly organized in a database, is that it can provide the basis, either for a human or an automated tool, to detect a wide variety of knowledge base errors, many quite subtle even for a knowledgeable nuclear engineer. The types of errors which are detectable and the basis for their detection are shown in Table 2.4.3-1.

The simulated knowledge-checking tool was introduced in the experiment as Meta-Check, a tool which checked the rules against the "meta-knowledge" about the nuclear plant. The output of the tool was presented in a form analogous to that of VERITE': an overall diagnostic message was given, the specific rule was listed, and the problematic IF clause(s) and/or THEN action(s) were indicated by an arrow. Examples of Meta-Check output are shown in Figure 2.4.3-1.

A major difference between the capability and output of VERITE' vs. Meta-Check is the latter's capability to fault-locate to a specific clause in one rule a problem that VERITE' would recognize only as a multiple-rule symptom. Given the engineering database, Meta-Check would be able to recognize 14 of the 21 seeded errors for System A, compared to 11 for VERITE' for matched errors; for System B, it would recognize all 11, as with VERITE', but would also be able to locate the offending clause(s) in specific rules.

Table 2.4.3-1 Errors detectable from the engineering knowledge compiled for the simulated META-CHECK tool.

No.	Type of Error	Basis for Detection
1	Variable (or Reference) not recognized	Not listed in Database
2	Reference value outside of variable range	Range of variable
3	Variable and Reference units don't agree	Variable and Reference Units
4	Variable and Reference have same units but are not comparable because they relate to different sub-systems	Not listed in Database as comparable
5	Comparison not recognized	Not listed in Database for this variable and reference combination
6	Contradiction in IF clauses	
7	A mistake in an IF clause causes clause to test for a <u>normal</u> , rather than abnormal, state of a plant variable, but a THEN Action specifies a course of action appropriate for the <u>abnormal</u> state of that plant variable; or vice versa. ^a	Two IF clauses are identical, except that one is negated Implication field of variable-reference comparison
8	Two THEN Actions are in conflict.	Either one is exactly a negation of the other, or the two Actions set values or specify operations which are known to be incompatible from other knowledge.

^a Such situation are not necessarily errors; however, they are at least anomalous and should be reported.

RULE PC/HG_2.3

```
IF
    (NOT(PC/P-8))                                /*NOT VENTING PC
    (PC H2 CONC IS ABOVE min DETECTABLE H2 CONC) /*0.5
    (NOT(H2 RECOMBINER max OPERATING LIMIT HAS BEEN EXCEEDED)) /*A SAVE
    (NOT(H2 DEFLAG LIMIT HAS BEEN EXCEEDED))      /*A SAVE
    (RAD RELEASE RATE IS BENEATH LCO RAD RELEASE RATE)
=> THEN
    (SET 4 III.F.1.a)                             /*VENT AND PURGE
    (SET CAUTION 6)                               /*DEFEAT ISOL INTERLOCKS
```

MESSAGES:**4III.F.1a**

VENT and PURGE the PC if rad release OK (see MORE, CAUT 6)

*** CAUTION 6 applies ***

If the rad release rate is expected to remain below 6×10^{-5} uci/cc, then vent and purge the primary containment to restore and maintain the H2 concentration below 0.5% as follows:

- Refer to sampling procedure KNPS 807.2
- RxBldg Normal Purge Exhaust System
- Rx Bldg Standby Gas Treatment System
- GN-HV-459, GN-HV-460 drywell vent valve (to PC)

SET CAUTION-6 DEFEAT ISOL INTLK

Defeating isolation interlocks may be required

////////////////////////////////////

POSSIBLE ERROR DETECTED:

COMPARISON "BENEATH" is not recognized for this variable. Check Variable/Reference Guideline.

RULE COMPOSED_POINTS_7

```
IF
    (MAIN STEAM LINE LEAK DETECTION HAS TRIPPED)
THEN
=> (SET STEAM LINE BREAK=INDICATED)          /*A SAVE VARIABLE
```

////////////////////////////////////

POSSIBLE ERROR DETECTED:

VARIABLE "s/pool" and REFERENCE "D/W DESIGN TEMPERATURE" are not known to be compatible. Check Variable/Reference Guideline.

RULE CAUTIONS_H.a

```
IF
    (SGTS IS RUNNING)
=> (DRYWELL TEMPERATURE NEAR OR ABOVE NO-COND EVAC TEMP)
THEN
    (SET CAUTION-H SGTS SUCTION)
```

MESSAGES:

SET CAUTION-H SGTS SUCTION

Do NOT evac area with DGTS unless temp<100(see MORE)

Do NOT evacuate an area with the SGTS unless area temperature is below 100 degC.

////////////////////////////////////POSSIBLE ERROR DETECTED:

VARIABLE "DRYWELL TEMPERATURE" and REFERENCE "NON-COND EVAC TEMP" are not known to be compatible. Check Variable/Reference Guideline.

Figure 2.4.3-1 Sample output from META-CHECK for System A

3 DESIGN OF BEHAVIORAL EXPERIMENT

3.1 Experimental Design

A major constraint in determining an appropriate experimental design was the practical difficulty in obtaining suitable participants for an extended period. Ideally, independent groups of participants would have been used for the three tool conditions (and the three controls), to form a completely between-subjects design⁵, for a total minimum number of subjects of 72 (albeit for a single day). Such a number was not feasible, and twenty participants were found to be the maximum number obtainable for the extended two-day testing period.

At least two different expert systems were needed to insure that performance/effects attributable to an interaction of the method with the system could be assessed. Because of the complexity of these systems, this variable had to be a between subjects type, with participants always working with the same system. Similarly, assignment of participants to either the tool or the control desk-checking version also needed to be a between-subjects variable, with participants always getting tools or always participating in control situations. With twenty participants, five were assigned to each of the four combinations of these two variables. Each of the four groups of participants then performed three sessions of testing, following an initial half day of training.

This experimental design is shown in Figure 3.1-1.⁶ Note that the specific methods are confounded,⁷ with successive sessions. Method 1 for Requirements Tracing actually involves two tasks, Requirements Grouping and Requirements Tracing, to form a total of four methods. However, for clarity of presentation in Figure 3.1-1, the two requirements tasks are treated as one. Were there a general learning or practice effect, then performance could improve for knowledge checking vs. the earlier methods due to these effects. Performance could also deteriorate over sessions due to cumulative fatigue or boredom. Despite the confounding, this is a very efficient design in terms of the minimal number of 20 participants, and experimental participants will always be compared to control participants who have had the same amount of prior testing. This means, in particular, that testing the significance of performance differences between expert systems or between the experimental and control conditions for a particular method can be interpreted in a straight-forward statistical manner.

The order of the methods was chosen to correspond to how V&V would actually best be performed for a system, with requirements activities followed by syntax checking followed by knowledge checking. This order is such that the previous method used by the participants would help them least on the next method, in comparison to other possible orders, thus reducing the "beneficial" transfer effects. For example, it is hard to imagine how having previously performed requirements tracing would not assist a participant very much in performing syntax checking.

⁵ The term "between-subjects design" means that different groups of subjects are used for the different experimental conditions, such as the expert system (A vs. B) and the experimental vs. control conditions of this experiment. In a "within-subjects design" the same group of subjects receive several experimental conditions. In this experiment, the four method conditions (requirements grouping, requirements tracing, syntax checking, and knowledge checking) constituted a within-subjects variable since all 20 subjects experienced all four of these conditions. Since both between-subjects and within-subjects conditions were used in this experiment it is designated as a "mixed" design.

⁶ That is, the same method was always used in the same session, as opposed to having some participants use method 1 during session 1, some use method 2 during session 1, etc. The result is that the effects due to session and those due to method cannot be separately evaluated.

⁷ Figure 3.1-1 and Table 3.1-1 provide details of special interest to persons with statistical design expertise.

Figure 3.1-1 Depiction of the Experimental Design.

Note: Depiction of the experimental design showing how participants (P1,P2,..., P20) were divided into four independent groups based on the combinations of two different expert system (A and B) and two methods conditions (Experimental --Exp'l --and Control). The four groups used three different V&V methods over the three sessions of the experiment.

Table 3.1-1 Planned analysis-of-variance model for the overall experiment¹

Sources of Variance	Symbol	Number of Levels	Degrees of Freedom	Appropriate F-ratio Error term
<i>Between-Subjects Sources</i>				
Expert Systems	E	2	1	S(EC)
Condition (Exp'l vs. Control)	C	2	1	S(EC)
Interaction between systems and conditions	E*C	4	1	S(EC)
Subjects nested within systems and conditions (Between-Subjects error)	S(EC)	n.a.	16	Within-in Subjects error
<i>Within-Subjects Sources</i>				
Methods ²	M	4	3	pooled error ³
Interaction between Methods and Systems	M*E	8	3	pooled error
Interaction between Methods and Conditions	M*C	8	3	pooled error
Interaction between Methods, Systems, and Conditions	M*E*C	16	3	pooled error
Subjects nested within systems and conditions by methods (Within-Subjects error)	S(EC)*M	n.a.	48	—

¹ In evaluating the effects of systems and conditions for each method, separate two-way Between-Subjects analyses of variance corresponding to the upper half of the table are intended.

² Both of the Requirements Tasks, Requirement Grouping and Requirement Tracing, are treated as separate methods; together with Syntax Checking and Knowledge Checking there are a total of four methods.

³ The Between-Subjects error term and the Within-Subjects error terms are planned to be pooled.

The details of the training and testing sessions are shown in the overall experimental schedule given in Figure 3.1-2. Each of the three testing sessions was limited to a total of 2.5 hours (.5 hrs. for initial/review training and 2 hrs. for testing). This was believed to be very near the maximum amount of focused activity tolerable by participants over the three session experiment without inducing serious fatigue effects. The technical details of the planned analysis of variance are shown in Table 3.1-1, treating the two requirements tasks (Requirements Grouping and Requirements Tracing) as separate methods, for a total of four methods.

The rule knowledge bases of the two expert systems were too large for participants to examine within a single session. Consequently, they were partitioned into three roughly equivalent parts, based on analyses provided by the VERITE' tool concerning three matching factors: (1) number of rules, (2) homogeneity of rules with respect to the sub-systems addressed, and (3) the complexity of the rule interactions. One partition was used for training purposes; the remaining two partitions were used for the Syntax Checking and Knowledge Checking testing. A subset of the requirements were used for Requirements Tracing, with the participants receiving the entire requirements listing, and the entire Knowledge Base, as part of their materials. Several requirements (not part of the subset used in the experiment) were used for training purposes.

3.2 Performance Measures

Participants' response sheets were designed to permit scoring for a wide variety of dependent variables. The 24 chosen measures are described in Table 3.2-1. Of these, the first six are the most important. These are the primary dependent variables. All the other variables are either components or derivations of these main variables.

3.3 Response Sheets

The response sheets captured detailed information about participants' judgments concerning problems they believed they had detected. The response sheet used for the Syntax Checking and Knowledge Checking methods (for System A) is shown as a representative example in Figure 3.3-1.

The top portion of the response sheet pertains to identification of the participant, method, and session. The first column, time started on a new problem, records the "start time" for the next problem the participant is searching for. The rule IDs are written in the next column. The confidence level in this rule, on a scale of 1-5, is recorded in column 4. The participants then selected a problem cause ID from the far right portion of the response sheet and recorded it in column 5. The participants recorded their confidence in this problem cause ID in column 6, and the time they finished working on the problem in column 7.

3.4 Selection and Assignment of Participants

Participants for this experiment were selected during a search process that actually began when SAIC was developing the proposal for this project. At that time, SAIC contacted a number of U.S. utilities with operating nuclear power plants requesting their participation in this project. Two utilities agreed and sent letters of commitment which were included in the proposal.

Once the date and participant needs for this experiment were decided, a search for additional participants commenced. The underlying strategy in obtaining participants was to include nuclear subject engineers as well as some software testing experts, with a significant representation from the nuclear industry. It was also important to have people with a wide range of experience level to encompass the spectrum of staff that might be involved in expert system V&V.

Figure 3.1-2 USNRC/EPRI Knowledge Based Certification Experiment Agenda

Schedule for Wednesday and Thursday on following page

Tuesday, September 22

11:30- 1200	Registration	
12:00 - 1:00	Lunch	
	General Instruction	(Everyone - Room A)
1:00 - 2:10	Overview of Experiment and General Comments Expert Systems: What they are, their components Overview of Testing of Conventional and Expert Systems	
	System Instruction	
2:10 - 2:30	System A Training System B Training	(Silver and Copper Groups - Room A) (Iron and Gold Groups - Room C)
	Testing and Method Instruction	
2:30 - 3:30	Requirements Training - General	(Everyone - Room A)
3:30 - 3:45	Break	(Everyone)
3:45 - 4:15	Requirements Tracing 1 Training Requirements Tracing 2 Training	(Silver and Iron Groups - Room A) (Copper and Gold Groups - Room C)
4:15 - 4:20	Fast Break	(Everyone)
4:20 - 4:40	Syntax Checking Training - General	(Everyone - Room A)
4:40 - 5:00	Syntax Checking 1 Training Syntax Checking 2 Training	(Silver and Iron Groups - Room A) (Copper and Gold Groups - Room C)
5:00 - 5:05	Fast Break	(Everyone)
5:05 - 5:30	Knowledge Checking 1 Training Knowledge Checking 2 Training	(Silver and Iron Groups - Room A) (Copper and Gold Groups - Room C)
6:00	Adjourn	
	Evening Review	
	Individual self-paced review of notes	

Figure 3.1-2 (Continued).

Wednesday, September 23

8:30 - 9:00	Breakfast	
9:00 - 12:00	Requirements Tracing 1 Requirements Tracing 2	(Silver and Iron Groups - Room A) (Copper and Gold Groups - Room C)
12:00 - 12:30	Business Break	(Everyone)
12:30 - 1:30	Lunch	(Everyone)
1:30 - 4:30	Syntax Checking 1 Syntax Checking 2	(Silver and Iron Groups - Room A) (Copper and Gold Groups - Room C)

Thursday, September 24

8:30 - 9:00	Breakfast	(Everyone)
9:00 - 12:00	Knowledge Checking 1 Knowledge Checking 2	(Silver and Iron Groups - Room A) (Copper and Gold Groups - Room C)
12:00 - 1:00	<i>Lunch</i>	(Everyone)
1:00 - 2:00	Comments and Debriefing	(Everyone - Room A)
2:00 - 2:15	Awards and Adieu	
2:30	<i>Adjourn - End of Experiment</i>	

Table 3.2-1 Description of the planned experimental dependent variables

No.	Dependent Variable	Description
1	TRSREP	Total number of responses
2	NCFP	Number of correctly found problems
3	PCFP	Percentage of correctly found problems
4	TIME	Total time for the exercise
5	MISS	Total number of problems missed (not found)
6	FA	Total number of false alarms (Participant responses that were not actual problems)
7	TFA	Total number of true false alarms
8	BFA	Total number of borderline false alarms (Responses that were not correct problems, but could not be totally dismissed either)
9	TFAL	Total number of true false alarms, low confidence
10	TFAH	Total number of true false alarms, high confidence
11	BFAL	Total number of borderline false alarms, low confidence
12	BFAH	Total number of borderline false alarms, high confidence
13	CONCP	Average confidence for correct problems found
14	CONFA	Average confidence for false alarms
15	CON	Overall average confidence
16	ATIME	Average time for all problems identified
17	ATIMEC	Average time for correct problems identified
18	ATIMEFA	Average time for false alarms identified
19	NHPF	Number of hard problems found

Table 3.2-1 (Continued).

20	NMPF	Number of medium problems found
21	NEPF	Number of easy problems found
22	THPF	Average time for hard problems identified
23	TMPF	Average time for medium problems identified
24	TEPF	Average time for easy problems identified

(Circle the Appropriate Answer)							
Name:		No:	Group: Silver Copper Iron Gold				Day: Tu We Th Time: AM PM
No.	Time Started on New Problem	ID(s) of the RULE(s) Containing the Suspected Problem	Confidence in Rule ID's	Problem Cause ID #	Confidence in Problem Cause ID	Time When Finished	Problem Cause IDs (Characterize type/Cause of problem)
1							1. Inconsistency or conflict among 2 or more rules
2							An Inconsistency within 1 rule
3							2. Among the IF clauses
4							3. Among the THEN
5							4. Between IF and THEN
6							Some clause or action probably missing or added
7							5. An IF clause missing or added
8							6. A THEN action missing or added
9							Problem in some part of a rule
10							7. Variable part of full IF
11							8. Comparison part of full IF
12							9. Reference part of full IF
13							10. Some aspect of abbrev. IF
14							11. Keyword part THEN
15							12. Other part THEN
16							13. OTHER
17							
18							
19							
20							
21							
22							
23							

Figure 3.3-1 Answer Sheet 2

To provide a reasonable chance of statistically detecting true experimental effects, the goal was a minimum of 16 participants (i.e., four in each group) with 20 participants being more desirable.

Discussions with the USNRC and EPRI as well as independent contacts with other nuclear power utilities and engineering companies resulted in the commitment of a total of 13 participants from three utilities (Baltimore Gas and Electric Company, Duke Power Company, and Yankee Atomic Electric Company), the USNRC Technical Training Center, the University of Maryland (graduate students), and EG&G Idaho, Inc. In addition, eight technical staff members from SAIC were participants, bringing the total number to 21. The extra participant over 20 was purposefully included to compensate for any last minute dropout of any of the other committed people.

Each participant filled out a two-page survey form (Figure 3.4-1) and, where available, also submitted their resume. These documents were used to characterize their experience and qualifications. Table 3.4-1 presents key qualification parameters for each of the 21 participants in this experiment. The most important parameters that were used in dividing the participants between the four groups were years of experience, expert system and software testing experience, and familiarity with the specific knowledge base subject matter (i.e., BWR emergency operating procedures and PWR systems). In addition, the composition of each group was designed to mix SAIC and non SAIC participants and separate individuals from the same organization (i.e., utility or USNRC) into a different group.

The four groups for this experiment were designated by one of four metal names: gold, copper, silver or iron. The silver and copper groups worked with Expert System A while the gold and iron groups worked with Expert System B. The silver and iron groups used new experimental techniques while the copper and gold groups were the control groups using current methods during the experiment. Each group consisted of five participants except the silver group which had six members (see Figure 3.4-2). This group was assigned the sixth member because it encompassed BWR systems and the experimental methods.⁸ As Table 3.4-1 indicates, only eight of the 21 participants had any experience in BWR systems.

With the 21 participants described in Table 3.4-1, the experiment consisted of: 38% SAIC staff, 38% nuclear utility staff, 10% USNRC staff, 10% graduate students, and 5% from a national laboratory. Moreover, the eight utility participants were evenly distributed as two in each of the four groups. Among the four groups, the average number of years of experience varied from 7.0 in the copper group to 11.0 in the silver group. Each group included two or three members with testing and/or expert system experience.

3.5 Error Seeding

Three sets of errors were developed for inserting into the three knowledge base partitions. These were derived on the basis of two principles: (1) the location of the error within a rule, and (2) the expected difficulty of detecting the error. This scheme was used only partially for System B, because of the reduced simplicity of these rules and the absence of rule-chaining; it was fully employed for generating the System A errors.

⁸ One of these six subjects was dropped from the analyses; cf Section 5.3.

Name:		Date:	
Organization:			
Job Title:			
Degree: (Please check highest degree obtained):			
☐ Ph.D.	☐ M.S.	☐ M.B.A.	☐ B.S.
☐ USN Nuclear School			
Field Your Highest Degree is in:			
Years of related experience since degree:			
Nuclear experience (Please check all that apply):			
☐ PWR Systems	☐ BWR Systems	☐ Emergency Operating Procedures	
☐ Operations	☐ Maintenance	☐ Design Engineering	
Expert Systems Experience (Please circle the value that best describes your experience):			
1	2	3	4
5			
No knowledge of expert systems	Book knowledge of expert systems	Participated in the development of at least one expert system	
Software Testing Experience (Please circle the value that best describes your experience):			
1	2	3	4
5			
No knowledge of testing	Have written and tested my own programs	Participated in the testing of a mid-to-large-scale system	
Knowledge of Nuclear Systems (Please enter the system name and circle the value that best describes your knowledge. See sample entry):			
	1	2	3
	4	5	
SARA	Book knowledge of the system	Hands-on experience with the system	Participated in the development of the system
	1	2	3
	4	5	
	Book knowledge of the system	Hands-on experience with the system	Participated in the development of the system
	1	2	3
	4	5	
	Book knowledge of the system	Hands-on experience with the system	Participated in the development of the system
	1	2	3
	4	5	
	Book knowledge of the system	Hands-on experience with the system	Participated in the development of the system

Figure 3.4-1 Participant Survey for Activity 6 - Knowledge Base Certification

Table 3.4-1 Experiment Participant Key Qualifications and Group Assignment

ID	Group	Organization	Highest Degree and Field	Yrs Exp.	Nuclear Experience	Test Exp.	ES Exp.
27	Copper	SAIC	B.S.; Nuclear Engineering	2	PWR, BWR, EOPs	No	No
73	Copper	EG&G Idaho	M.S.; Electrical Engineering	19	PWR, BWR, Design	Yes	Yes
44	Iron	Duke Power Company	B.S.; Electrical Engineering	9	PWR, Maintenance	No	Yes
99	Silver	SAIC	B.S.; Psychology/Statistics	2	BWR, EOPs	No	No
97	Gold	SAIC	B.S.; Nuclear Engineering	3	BWR, EOPs, Design	No	Yes
39	Silver	University of Maryland	M.S.; Electro Optics	2	PWR	Yes	Yes
82	Copper	SAIC	M.S.; Engineering Physics	2	PWR, BWR	Yes	Yes
23	Silver	USNRC	B.S.; Nuclear Technology	23	PWR, EOPs, Operations, Maintenance	No	No
56	Iron	SAIC	B.S.; Nuclear Engineering	2	PWR, BWR, EOPs, Design	Yes	Yes
46	Silver	SAIC	B.S.; Electrical Engineering	3	--	Yes	No
51	Copper	Balt. Gas and Electric	B.S.; Nuclear Science	10	PWR, EOPs, Operations	No	No
35	Copper	Duke Power Co.	B.S.; Chemical Engineering	2	PWR, EOPs, Operations	No	No
38	Gold	SAIC	M.S.; Mechanical Engineering	2	PWR, BWR, EOPs	No	No
43	Gold	Yankee Atomic	M.S.; Nuclear Engineering	12	PWR, EOPs	Yes	Yes
25	Iron	Balt. Gas and Electric	M.S.; Nuclear Engineering	12	PWR, Operations, Maintenance	Yes	Yes
65	Gold	University of Maryland	M.S.; Nuclear Engineering	7	PWR, EOPs, Operations, Maintenance	Yes	Yes
54	Gold	Balt. Gas & Electric	Shift Sup; SRO	20	PWR, EOPs, Operations	No	No
62	Silver	Yankee Atomic	B.S.; Mechanical Engineering	22	PWR, BWR, EOPs, Design	No	No
75	Silver	Balt. Gas & Electric	M.S.; Computer Science	14	PWR, EOPs, Operations, Maintenance	Yes	Yes
67	Iron	USNRC	B.S.; Mechanical Engineering	11	PWR, EOPs, Operations	No	No
88	Iron	SAIC	B.A. Mathematics	4	--	Yes	No

Notes: Shift Sup = Nuclear Power Plant Shift Supervisor Nucl = Nuclear BWR = Boiling Water Reactor
SRO = Senior Reactor Operator PWR = Pressurized Water Reactor EOPs = Emergency Operating Procedures

Figure 3.4-2 Experimental Teams

	System A	System B
Experimental	<p>Silver 39 Univ. of Maryland</p> <p>Silver 46 SAIC</p> <p>Silver 62 Yankee Atomic</p> <p>Silver 23 USNRC</p> <p>Silver 75 BG&E</p> <p>Silver 99 SAIC</p>	<p>Iron 25 BG&E</p> <p>Iron 56 SAIC</p> <p>Iron 88 SAIC</p> <p>Iron 67 USNRC</p> <p>Iron 44 Duke Power</p>
Control	<p>Cooper 51 BG&E</p> <p>Cooper 73 EG&G Idaho</p> <p>Cooper 27 SAIC</p> <p>Cooper 82 SAIC</p> <p>Cooper 35 Duke Power</p>	<p>Gold 66 Univ. of Maryland</p> <p>Gold 97 SAIC</p> <p>Gold 43 Yankee Atomic</p> <p>Gold 38 SAIC</p> <p>Gold 54 BG&E</p>

Ten different error locations were specified, six involving the IF clauses and four involving the THEN actions as shown in Figure 3.5-1.⁹ The six IF-clause locations for errors were: (1) in the variable, or left-hand-side, of a full IF clause (e.g., in "RPV PRESSURE" for the clause "(RPV PRESSURE > PC PRESSURE)"); (2) in the comparison operator for a full IF clause (e.g., ">"); (3) in the reference, or right-hand-side, of a full IF clause (e.g., "PC PRESSURE" in the above rule); (4) in the negation of a reduced (Boolean) IF clause; (5) in the presence of an additional IF clause added to the rule; and (6) in the absence of an IF clause deleted from a rule. The four THEN-action locations for errors were: (1) in the keyword of a THEN action (e.g., "SET"); (2) in some other part of a THEN action; (3) in the presence of an additional THEN action added to the rule; and (4) in the absence of a THEN action deleted from a rule.

One error was designed for each of the four deleted or added IF-clause and THEN-action locations, and the remaining errors were distributed roughly equally among the remaining six possibilities.

Seven levels of expected detectability ranging from "should be very obvious to anyone" to "would require a nuclear engineer intimately familiar with this expert system" were used. These are described in Table 3.5-1. Three to four errors were chosen for each level of detectability. A total of 20 and 21 errors were seeded into the System A knowledge base partitions for the syntax checking and knowledge checking methods, respectively. A total of 11 errors were seeded into each of the System B syntax checking and knowledge checking methods.

The ten types of errors were intended to cover the whole space of possible errors that could be made (i.e., incorrect modifications of all the possible parts of the rules, rules and additions, and deletions to the two parts of rules). Were data available on the empirical frequency of errors found in these locations, that frequency profile would have been followed in selecting the mix of errors to be seeded. As it is, the choice of roughly equal frequencies of error locations, distributed equally over judged detectability levels, insures that the performance results will not be biased by the unequal selection of one type of error over another.

To simplify analysis of the participant responses, these seven levels were condensed into three categories: hard, medium, and easy, with hard being the most difficult to detect, and easy being the simplest to detect. The levels of detectability are categorized as follows:

D1 -	Easy or Medium
D2 -	Medium or Hard
D3 -	Easy or Medium
D4 -	Medium or Hard
D5.1 -	Medium or Hard
D5.2, D5.3 -	Medium
D5.4 -	Easy or Medium
D5.5 -	Easy
D5.6 -	Medium or Hard
D6 -	Hard
D7 -	Hard

⁹ Although IF-THEN constructs are used in many conventional imperative programming languages, such statements are only superficially comparable to the IF-clause THEN-action rules in expert systems since the transfer-of-control properties are so different in the two kinds of systems.

IF-clause error locations

- 1 (ISV) Left-hand-side of a full IF clause:

IF (RPV PRESSURE > PC PRESSURE) THEN
(SET 13 III.B.3)
- 2 (ISC) comparison operator of a full IF clause:

IF (RPV PRESSURE>PC PRESSURE) THEN (SET 13 IIIB.3)
- 3 (ISR) Right-hand-side of a full clause:

IF (RPV PRESSURE>PC PRESSURE) THEN (SET 13 III.B.3)
- 4 (IN) Negation of a reduced IF clause:

IF NOT (RPV EMERGENCY DEPRESSURIZATION REQUIRED)
THEN (SET 13 III.B.2)
- 5 (IC+) Additional IF clause added to the rule:
IF (RPV PRESSURE > PC PRESSURE)
(HPCS IS RUNNING) THEN (*added*) (SET 13 III.C.7)
- 6 (IC-) IF clause deleted from a rule:

IF (RPV PRESSURE > PC PRESSURE)
(HPCS IS RUNNING)THEN (*deleted*) SET 13 III.C.7

Figure 3.5-1 Error Locations

THEN-action error locations:

- 1 (TAK) in THEN-action keyword:

IF (RPV PRESSURE > PC PRESSURE) THEN
SENT 13 III.C.7

- 2 (TAV) in some other part of the THEN-action

IF (RPV PRESSURE > PC PRESSURE) THEN
(SET 13 111.Z.2)

- 3 (TA+) Additional THEN-action added to the rule:

IF (RPV PRESSURE > PC PRESSURE) THEN
(SET 13 III.B.2)
(*added*) (SET 12 III.C.1)

- 4 (TA-) THEN-action deleted from a rule:

IF (RPV PRESSURE > PC PRESSURE) THEN
(SET 13 III.B.2)
(*deleted*) (SET 12 III.C.1)

Figure 3.5-1 (Continued)

Table 3.5-1 Characterization of Possible Errors

Detectability	
D1 =	possible to detect directly, logically, looking just at one clause or action, no additional information required -- obvious mis-spelling, obvious mistake in variable/reference name/operator, or obvious missing element
D2 =	possible to detect directly, logically, by looking at another clause in the same (IF or THEN) part of rule
D3 =	may be possible to see a problem directly, logically, by comparing clause(s) in IF part of rule to clause(s) in THEN part, because THEN action is inconsistent in some way with some IF clause
D4 =	not possible to detect by direct inspection but possible to see the problem by looking at other rules; meta-knowledge is not necessary:
D4.1 =	detectable by looking at a rule-chain of two rules
D4.2 =	detectable by looking at similar surrounding rules which establish that problem is an exception to the spelling, syntax, etc. of these other rules
D5 =	not possible to detect directly or via examination of rule-chains or other rules, but detectable from supplied meta-knowledge about variables, their operators and their references:
D5.1 =	variable and reference have different units
D5.2 =	variable and reference have same units, but not associated with each other
D5.3 =	THEN action is subtly inconsistent with implications of IF clause(s) (e.g., operator is reversed) (Less obvious than D3)
D5.4 =	reference is outside of allowable range
D5.5 =	unallowable operator
D5.6 =	variable or reference name is apparently reasonable but in fact invalid (i.e., not on list of allowable variable and reference names)
D6 =	not possible to detect directly, requires a rule-chain of length 3 or more and knowledge of precursor states in these rules

Table 3.5-1 Characterization of Possible Errors

D7 =	not possible to detect directly, not detectable by examining rule-chains of any length or any rule contexts, and NOT DETECTABLE from any supplied meta-knowledge. Requires high familiarity with application and/or software implementation to detect.
------	--

This categorization is depicted in Figure 3.5-2. For detectability levels that fall in two categories, such as D1, the context dictates the category. For example, some D1 errors will be very obvious due to the same word being spelled correctly and incorrectly in a rule. Other D1 errors could be more subtle.

The errors seeded for syntax checking and knowledge checking into both Systems A and B are shown in Appendix A. The level of detectability, location, and difficulty category are provided for each error.

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

4 KNOWLEDGE BASE CERTIFICATION EXPERIMENT

The actual knowledge base experiment took place from September 22 through September 24, 1992 at the SAIC offices in McLean, Virginia. Twenty-one subjects participated in the experiment as discussed in Section 3.5. The experiment was broken into 16 major parts:

Afternoon of September 22:

- Introduction and Overview
- General Instruction and Training
- Training on System A and System B
- General Training on Requirements Tracing
- Method Specific Training on Requirements Tracing
- General Training on Syntax Checking
- Method Specific Training on Syntax Checking
- General Training on Knowledge Checking
- Method Specific Training on Knowledge Checking

Morning of September 23:

- Refresher Method Specific Training on Requirements Tracing
- Actual Requirements Tracing Experiment

Afternoon of September 23:

- Refresher Method Specific Training on Syntax Checking
- Actual Syntax Checking Experiment

Morning of September 24:

- Refresher Method Specific Training on Knowledge Checking
- Actual Knowledge Checking Experiment

Afternoon of September 24:

- Debriefing

In the interest of space conservation, the hundreds of pages of experimental materials are not included here. Samples of all experimental materials were presented in Sections 2.4.1-2.4.3.

Each of these major parts of the experiment are described in detail in the following sections.

4.1 Introduction and Overview

The experimenters were qualified to instruct and direct the participants, with a balance of nuclear, expert systems, and verification and validation expertise. The experiment was begun with a definition of verification, validation, and expert systems. The role of V&V guidelines in the nuclear industry was briefly discussed. The experiment was described in very general terms. A schedule for the experiment was presented. Finally, the participants were asked not to discuss the experiment among themselves. This reminder was repeated frequently during the experiment. Also, participants were informed that the best performers would receive a special award; this competition promoted confidentiality.

4.2 General Instruction and Training

The experimenters provided some general instruction and training on expert systems and Verification and Validation (V&V).

4.2.1 Expert Systems Training

The expert systems training was very general in nature. A definition of expert systems was followed by a discussion of knowledge and knowledge engineering. Advantages of expert systems were discussed. Expert system limitations were detailed. The components of expert systems were then described. In particular, the concept of a knowledge base and inference engine was presented.

4.2.2 Verification and Validation Training

The Verification and Validation training consisted of the definition of terms, an examination of why V&V should be performed, and qualifications for the Independent Verification and Validation (IV&V) team. An overview of testing then followed. The concepts of static and dynamic testing were introduced. The terms "anomaly" and "invalidity" were defined. V&V guidelines for knowledge base systems were covered. For example, the importance of maintaining a requirements document for a knowledge base system was stressed. Other V&V guidelines covered were requirements tracing, committing to V&V, and utilizing Barry Boehm's seven principles for software development (Boehm, 1976).

4.3 Systems A and B Training

During the first day (September 22, 1992) of the experiment, SAIC provided basic training to the participants on the two expert systems, denoted A and B, which were being used as a source of knowledge base rules for the experiment. Groups were separated according to their assigned system, with the copper and silver groups in one conference room receiving System A training while the gold and iron groups were in another room undergoing System B training. The training was designed to provide a fundamental understanding of the purpose and function of each expert system with the emphasis on an explanation of actual sample rules taken from their respective knowledge bases.

For System A, the developer and current field application was identified. System A was developed to follow Boiling Water Reactor (BWR) Emergency Operating Procedures (EOPs) during an abnormal event. Participants were informed that this system is designed to continually monitor the plant state and compare it to symptoms in its EOPs

with the end result of providing, but not performing, recommended operator actions. The fact that a different version of this system is on-line and real time, but does not replace written EOPs, was also discussed with this group. Some key design features that were included in this training were: (1) ability to simultaneously follow multiple plant symptoms, plant states, and EOPs, (2) the linkage of each rule to a message statement, (3) user retrieval of the basis for a message being generated, (4) the "wait" feature for operator judgment calls, and (5) the exclusion of operator input or presumed actions. The history and difference between event-based and symptom-based EOPs was also presented to the group. Finally, the System A training ended with a detailed presentation and analysis of two actual rules from the System A knowledge base.

The System B training presented the origins and current application site for this expert system. System B currently models key systems in a Pressurized Water Reactor (PWR) to determine the status of six Critical Safety Functions. These Critical Safety Functions are: reactivity control, reactor coolant system (RCS) inventory control, RCS pressure control, RCS transport control, RCS integrity control, and heat sink control. The subject of success trees which are used in System B was discussed, including an example taken directly from this system for heat sink control. The success trees assess the availability of combinations of equipment which are necessary to perform a critical safety function. In addition, support systems are modelled to determine the relationship between their operability and that of the systems which are directly responsible for affecting critical safety functions.

The design features of System B include the following: (1) monitoring the plant critical safety function status using thermodynamic conditions and success paths, (2) determining conflicts between status evaluations, (3) determining the effect of support system failures on the front line or key safety systems, and (4) alerting the user to plant problems and changes in status. System B is based on technical specifications and design limits, but not on operating limits, and much of the system model is generic and not plant specific. Finally, the group was advised that the system currently only models hot shutdown conditions. This modeling restriction has no effect on the conduct or results of the experiment.

At the conclusion of the simultaneous System A and B training, the respective participant groups were asked to keep the actual names of A and B confidential. Also, they were instructed not to draw any conclusions about the quality of these systems because they would be looking at subsets of the knowledge base rules for earlier versions of these systems containing errors that had been purposefully seeded.

4.4 Requirements Tracing

4.4.1 General Requirements Tracing Training

Following the system specific training on the afternoon of September 22, all participants were trained on requirements tracing. This training covered requirements grouping (manual method only) and requirements tracing (manual method only) as described in Section 2.3.1. The participants were instructed on the classes of errors to look for during the experiment: requirement omissions, incomplete or incorrect requirement satisfaction, and unintended functions (see Section 2.4.1). Examples from both System A and System B were used. Participants worked through a total of four examples: grouping requirements for System A; grouping requirements for System B; tracing of a requirement for System A; and tracing of a requirement for System B. The participants were also instructed on how to

fill out their response sheets. It was emphasized that the materials being used did not represent the original materials for the two expert systems. The training lasted approximately 1 hour and 15 minutes.

4.4.2 Experimental (Method 1) Requirements Tracing

4.4.2.1 Experimental (Method 1) Requirements Tracing Training

Members of the Silver and Iron groups were briefed on a new method for grouping requirements (SuperTrace printouts) and a new requirements tracing technique known as requirements traceability "maps." The SuperTrace printouts, described in Section 2.3.1.1, list all requirements from a Functional Requirements Document that belong to a particular group or category (such as "software," and "RPV Flooding"). The printouts were explained to the participants, and then two examples were worked through, one for each system.

The maps, described in Section 2.3.1.2, present all the information needed to trace a requirement on a single sheet of paper. They are, in effect, a self-contained requirements tracing document. The maps were explained to the participants, who were taught how to look for the three types of errors discussed above. The Silver and Iron participants then worked through two examples using the maps, one for each system.

4.4.2.2 Experimental (Method 1) Requirements Tracing Materials

Each participant was given a Requirements Tracing notebook (for either System A or B). During the training, each section of the notebook was explained:

- Table of Contents - lists the contents of the notebook
- Functional Requirements Document (FRD) - the actual requirements document for the system
- Functional Requirements Printout - an itemized listing of the requirement ID, requirement text, section number, section title, and keywords for each requirement in the FRD (generated by SuperTrace)
- Parameter Trees (Graphs) - a graphical depiction of the tree hierarchy with rules annotated on the lowest levels of the trees. This was provided to the Iron group.
- Emergency Operating Procedure (EOP) messages - a listing of the EOP message text for all the pertinent messages. This was provided to the Silver group.
- Rules - a listing of the rule names and the text of the rules
- Requirements Traceability Maps - described in Section 2.3.1.2
- "Grouped" Functional Requirements - printouts similar to the functional requirements printout, but printed for various categories (such as "logic," and "Computer Human Interface")
- Acronym List - a list of applicable acronyms and their definitions

In the front pocket of the notebooks were:

- Requirements Tracing Strategy Sheet - a "cook book" that lists the steps to be performed during the experiment
- Response Sheets - for grouping and tracing

4.4.2.3 Experimental (Method 1) Requirements Tracing Experiment

Grouping of Requirements. The instructor had all participants retrieve the appropriate response sheet, and fill out all the general information (e.g., subject name and subject number) in unison. The participants were then told to "start". Participants were given 45 minutes to group requirements. When the time period had ended, all response sheets were collected. There were several questions asked during the experiment. These questions ranged from "am I doing this right?" to "is this the right answer?" No information was given out concerning whether a correct problem had been found. Information previously presented during training was reiterated to the question askers. On occasion, the monitor would take a few seconds to repeat this training information to the whole group.

Tracing of Requirements. The instructor had all participants retrieve the appropriate response sheet, and fill out all the general information (e.g., subject name and subject number) in unison. The participants were then told to "start". Participants were given 1 hour and 30 minutes to trace requirements. When the time period had ended, all response sheets were collected. There were several questions asked during the experiment similar to those in the grouping task. No information was given out concerning whether a correct problem had been found. Information previously presented during training was reiterated to the question askers. On occasion, the monitor would take a few seconds to repeat this training information to the whole group.

4.4.3 Control (Method 2) Requirements Tracing

4.4.3.1 Control (Method 2) Requirements Tracing Training

Members of the Copper and Gold groups were briefed on the state-of-the-art methods for grouping requirements and for tracing requirements. For grouping requirements, the participants were given the original Functional Requirements Document and were told to identify important words and phrases in each requirement. Then, given a topic to "group" by, the participants were taught to look through the document attempting to find requirements whose key words and phrases dealt with the topic of interest. After this was explained, two examples were worked through, one for each system.

For tracing requirements, the participants were given the original Functional Requirements Document, the parameter trees (graphs) or EOP messages (depending on the system), and the rules. The participants were taught to identify key words or phrases in the requirement text, and to note the section title for the requirement. The section title was used to locate the parameter tree or EOP message. The keywords or phrases were used to help locate the node (box with a name in it) on the parameter tree and the rules that traced to the requirement. The participants were reminded about the three types of errors to look for. The Copper and Gold participants then worked through two examples, one for each system.

4.4.3.2 Control (Method 2) Requirements Tracing Materials

Each participant was given a Requirements Tracing notebook (for either System A or B). During the training, each section of the notebook was explained:

Table of Contents - lists the contents of the notebook

Functional Requirements Document (FRD) - the actual requirements document for the system
Parameter Trees (Graphs) - a graphical depiction of the tree hierarchy. This was provided to the Gold group.
Emergency Operating Procedure (EOP) messages - a listing of the EOP message number and EOP message text for all the pertinent messages. This was provided to the Copper group.
Rules - a listing of the rule names and the text of the rules
Acronym List - a list of applicable acronyms and their definitions

In the front pocket of the notebooks were:

Requirements Tracing Strategy Sheet - a "reminder sheet" that lists the steps to be performed during the experiment
Response Sheets - for grouping and tracing

4.4.3.3 Control (Method 2) Requirements Tracing Experiment

Grouping of Requirements. The instructor had all participants retrieve the appropriate response sheet, and fill out all the general information (e.g., subject name and subject number) in unison. The participants were then told to "start". Participants were given 45 minutes to group requirements. When the time period had ended, all response sheets were collected. There were several questions asked during the experiment. These questions ranged from "am I doing this right?" to "is this the right answer?" No information was given out concerning whether a correct problem had been found. Information previously presented during training was reiterated to the question askers. On occasion, the monitor would take a few seconds to repeat this training information to the whole group.

Tracing of Requirements. The instructor had all participants retrieve the appropriate response sheet, and fill out all the general information (subject name, subject number, etc.) in unison. The participants were then told to "start". Participants were given 1 hour and 30 minutes to trace requirements. When the time period had ended, all response sheets were collected. There were several questions asked during the experiment. These were of the same kind, and handled in the same way, as those in the grouping task.

4.5 Syntax Checking

4.5.1 General Syntax Checking Training

Following the general training on requirements tracing on the afternoon of September 22, all participants were presented general information on syntax checking. This training covered syntax checking concepts which are described in Section 2.3.2. The participants were reminded about the parts of a rule (IF "variable" part, "comparison" part, "reference" part, THEN "action."). The participants were instructed on the types of errors to look for during the experiment. The major types of errors (e.g., cyclic inference chain, error internal to the clause or action, illegal action, conflicting rules, subsumed rule pair, unfirable rule, and error in the THEN clause) are described in Section 3.6, Error Seeding. Examples from both System A and System B were presented for each error type. Participants were shown a sample rule from one of the systems, and brainstormed on the types of errors that could "slip" into the rule post-compilation. Participants were informed that errors had been introduced for the experiment. The information to be

written on the response sheets was reviewed, and a sample response sheet was shown. The training lasted approximately 35 minutes.

4.5.2 Experimental (Method 1) Syntax Checking

4.5.2.1 Experimental (Method 1) Syntax Checking Training

Members of the Silver and Iron groups were briefed on an automated tool for syntax checking called VERITE'. First, the participants were taught that VERITE' reports on the effects of errors, and that it should be used only as a guide to search for errors. VERITE' will report that an error is believed to exist in one or several rules. The participants were taught to examine the rules mentioned, and to see if any kind of problem existed there (not necessarily the problem cited by VERITE'). The types of errors detected by VERITE' were discussed (e.g., cyclic inference chain, unsatisfiable condition, illegal action, conflicting rules, subsumed rule pair, unfirable rule, and unused consequent). These error types are also discussed in Section 3.6, Error Seeding. Participants were shown a sample rule from one of the systems, and brainstormed on the types of errors that could "slip" into the rule post-compilation. Participants were informed that errors had been introduced for the experiment. The information to be written on the response sheets was reviewed, and a sample response sheet was shown. The materials to be used were then discussed, followed by the participants working through one practice problem.

4.5.2.2 Experimental (Method 1) Syntax Checking Materials

Each participant was given a Syntax Checking folder (for either System A or B). During the training, the contents of the folder were explained:

- Copy of the System Rules - a listing of the rule names and the text of the rules, separated by bold lines
- Copy of the VERITE' Output - a listing of the type of problem detected and the rule ID(s), the clause(s) containing the error, and the rule(s) involved in the problem
- Acronym List - a list of applicable acronyms and their definitions
- Response Sheets

4.5.2.3 Experimental (Method 1) Syntax Checking Experiment

The instructor had all participants retrieve the response sheet, and fill out all the general information (e.g., subject name and subject number) in unison. The participants were then told to "start". Participants were given 2 hours to identify syntax errors in the rules. When the time period had ended, all response sheets were collected. There were several questions asked during the experiment. These were of the same kind, and handled in the same way, as those in Section 4.4.3.3).

4.5.3 Control (Method 2) Syntax Checking

4.5.3.1 Control (Method 2) Syntax Checking Training

Members of the Copper and Gold groups were briefed on the state-of-the-art methods for checking the syntax of a knowledge base. This training covered syntax checking concepts which are described in Section 2.3.2. The participants were instructed on the types of errors to look for during the experiment. The major types of errors (e.g., cyclic inference chain, error internal to the clause or action, illegal action, conflicting rules, subsumed rule pair, unfirable rule, and error in the THEN clause) are described in Section 3.6, Error Seeding. Examples from both System A and System B were presented for each error type. Participants were shown a sample rule from one of the systems, and brainstormed on the types of errors that could "slip" into the rule post-compilation. Participants were informed that errors had been introduced for the experiment. The information to be written on the response sheets was reviewed, and a sample response sheet was shown. The materials to be used were then discussed.

4.5.3.2 Control (Method 2) Syntax Checking Materials

Each participant was given a Syntax Checking folder (for either System A or B). During the training, the contents of the folder were explained:

- Copy of the System Rules - a listing of the rule names and the text of the rules, separated by bold lines
- Acronym List - a list of applicable acronyms and their definitions
- Response Sheets

4.5.3.3 Control (Method 2) Syntax Checking Experiment

The instructor had all participants retrieve the response sheet, and fill out all the general information (e.g., subject name and subject number) in unison. The participants were then told to "start". Participants were given 2 hours to identify syntax errors in the rules. When the time period had ended, all response sheets were collected. There were several questions asked during the experiment. These were of the same kind, and handled in the same way, as those in the requirements grouping task (Section 4.4.3.3).

4.6 Knowledge Checking

4.6.1 General Knowledge Checking Training

Since the afternoon session on September 22 was highly interactive in nature and ran late, the general knowledge checking training was deferred to September 23 as part of the method specific training. This did not effect the experiment since knowledge checking and syntax checking are tightly coupled (at the general level). (In fact, data analysis results indicate that the participants found more problems correctly using knowledge checking than syntax checking.) The materials were still given to the participants, however. These charts covered knowledge checking concepts which are described in Section 2.3.3. First, the charts defined knowledge checking. The charts discussed the types of errors to look for during the experiment. The major types of errors (e.g., misspelled or completely unknown, out of context, conflicting with another piece of knowledge, inaccurate knowledge, and inconsistent knowledge) are described in Section 3.6, Error Seeding. Strategies for finding these types of problems in single clauses or actions, within single rules, or between two or more rules were included in the briefing charts. For example, the following

strategies were suggested for locating errors in single clauses or actions: 1) look for misspellings; 2) look for variable and reference terms that haven't been seen before, and therefore may be incorrect; 3) look for mistakes in the comparison such as incorrect choice of operator, etc.

4.6.2 Experimental (Method 1) Knowledge Checking

4.6.2.1 Experimental (Method 1) Knowledge Checking Training

Members of the Silver and Iron groups were briefed on a new tool for knowledge checking called Meta-Check. First, the participants were taught that Meta-Check uses meta-knowledge (knowledge about knowledge) to find errors. Some examples of meta-knowledge were given (e.g., names of things in the application, reasonable comparisons, allowable states of things, allowable manners of expressing states, relationships among system states, states that can and cannot occur together, ranges of variables, units of variables, and implication of a particular state or comparison between states). Meta-Check's messages and their meaning were explained. An example from System A was shown. Participants were instructed that Meta-Check may not have found all the knowledge errors, and that they should examine all the other rules very closely. To assist in finding these other errors, the variable and reference description table was explained. The materials to be used were then discussed.

4.6.2.2 Experimental (Method 1) Knowledge Checking Materials

Each participant was given a Knowledge Checking folder (for either System A or B). During the training, the contents of the folder were explained:

- Copy of the System Rules - a listing of the rule names and the text of the rules, separated by bold lines
- Meta-Check Listing - listing of potential problems
- Variable and Reference Description Table
- Response Sheets

4.6.2.3 Experimental (Method 1) Knowledge Checking Experiment

The instructor had all participants retrieve the response sheet, and fill out all the general information (e.g., subject name and subject number) in unison. The participants were then told to "start". Participants were given 2 hours to identify knowledge errors in the rules. When the time period had ended, all response sheets were collected. There were several questions asked during the experiment. These were of the same kind, and were handled in the same way, as those in the requirements grouping task (Section 4.4.3.3.).

4.6.3 Control (Method 2) Knowledge Checking

4.6.3.1 Control (Method 2) Knowledge Checking Training

Members of the Copper and Gold groups were briefed on meta-knowledge (knowledge about knowledge) and how to use it to find errors. Some examples of meta-knowledge were given (e.g., names of things in the application, reasonable comparisons, allowable states of things, allowable manners of expressing states, relationships among system

states, states that can and cannot occur together, ranges of variables, units of variables, and implication of a particular state or comparison between states). Examples from both systems were used. Next, a "laundry" list of the types of problems that can be discovered using meta-knowledge was covered. For example: a keyword is not correct and a variable is not correct. Errors between parts of a rule or between different rules were covered. The participants were given some hints on finding errors in the rules. The materials to be used were then discussed.

4.6.3.2 Control (Method 2) Knowledge Checking Materials

Each participant was given a Knowledge Checking folder (for either System A or B). During the training, the contents of the folder were explained:

Copy of the System Rules - a listing of the rule names and the text of the rules, separated by bold lines
Response Sheets

4.6.3.3 Control (Method 2) Knowledge Checking Experiment

The instructor had all participants retrieve the response sheet, and fill out all the general information (e.g., subject name and subject number) in unison. The participants were then told to "start". Participants were given 2 hours to identify knowledge errors in the rules. When the time period had ended, all response sheets were collected. There were several questions asked during the experiment. These were of the same kind, and handled in the same way, as those in the requirements grouping task (Section 4.4.3.3).

4.7 Debriefing

At the completion of this experiment on the afternoon of September 24, 1992, the participants all gathered in one of the two conference rooms used for the three days and were given a debriefing by the SAIC project staff. The participants were informed that they had been divided up into four groups consisting of an experimental and control group for each of the two nuclear expert systems. They were also told that the control groups were given the current state-of-the-art methodology for requirements tracing, requirements grouping, finding syntax errors, and knowledge checking whereas the experimental groups were provided with new innovative aids and tools that were designed to assist them in uncovering knowledge base errors. SAIC revealed that several seeded errors were not discovered by any of the experiment participants (see Section 5.9).

Although detailed grading of the experiment results had not been completed, an evaluation of the number of seeded errors found and the time to find them was made for each participant. This assessment resulted in the award of a textbook on nuclear industry expert systems to the best participant in the control and experimental groups. The participants were asked to provide any written comments on the experiment to SAIC prior to their departure.

As part of this debriefing, both the USNRC and EPRI project managers discussed the importance of this experiment and project to the use and regulation of expert systems in the nuclear industry. They also emphasized that results of this project are expected to influence conventional software V&V as well as important ongoing projects, such as digital feedwater control systems. The USNRC project manager stressed that the participants should not consider errors discovered during this experiment to in any way relate to the quality of the actual expert systems which were used

for this task. These errors were seeded by SAIC and earlier versions of the expert systems' knowledge bases were used in this experiment. Finally, participants were reminded that these expert systems should be referred to as Systems A and B even though they may be aware of their true identity.

After the debriefing, participants were provided with copies of three papers on the subject of expert system V&V and handed in written comments. These comments will be discussed in Section 5.10 of this report.

5 EXPERIMENTAL RESULTS AND DISCUSSION

This section provides the results of analyzing the data collected in the three experimental sessions. The basic approach was to apply the analysis-of-variance model described in Table 3.1-1 to each of the 24 dependent variables described in Table 3.2-1. These analyses were supplemented, as necessary, by more detailed analyses to permit understanding of a particular finding.

Some essential statistical concepts are first described, followed by a description of the very conservative approach taken in all analyses. Then a short sub-section provides a summary of the main findings. The rest of this section goes into details of the findings and discusses the pattern of results across the 24 performance measures studied. The final conclusions and recommendations are given in the next section (Section 6).

5.1 Three Essential Concepts

Every attempt has been made to subordinate the technical details of these analyses in favor of the relevance of the findings. Nevertheless, three concepts are essential to an appropriate understanding of the results:

- A **factor, variable, or treatment** refers to one of the primary experimental manipulations, of which there were three:
 - 1) assigning participants to expert systems, System A or B,
 - 2) assigning participants to one of the two conditions, Experimental (receiving the special tool) or Control (using no special tool but the best existing manual desk-checking approach), and
 - 3) giving participants a particular V&V method, two having to do with Requirements, one relating to Syntax Checking, and one relating to Knowledge Checking.
- **Interaction** refers to the possibility that performance for a particular factor may depend on another factor. For example, it may be the case that requirements tracing with the special tool was better than using the "hand" or "manual" approach, but the difference between these two conditions was greater for System A than for System B. Were this the case, it could be said that "there was an interaction between the condition and expert system factors".
- **Significance** refers to whether or not an observed difference in performance can be attributed to the manipulations of the experiment, or is something that could be expected to occur on a chance basis due to the random sampling effects or other effects of random variables. Most of the statistical analyses performed were concerned with the question of the significance of a particular factor or interaction and computed an estimate of the probability that an observed difference occurred by chance. By convention, if the computed probability of observed effects was less than 0.05 of having occurred by chance,

expressed as $p < .05$, the performance difference is said to be significant, meaning that it is to be interpreted as arising not from random events but from the experimental manipulation.¹⁰

5.2 Conservatism of Analyses

The statistical analysis and interpretation of data from behavioral experiments is a complex activity, and there are many types of errors which can be made. These include violation of underlying assumptions of the chosen statistical technique (e.g., non-normal distribution of scores in analysis-of-variance tests) and not controlling for the possibility of spurious findings when using many dependent variables (e.g., not controlling for the experiment-wise error rates). Without going into these, it is sufficient to say that the most conservative analysis, test, or interpretation was always chosen (e.g., selecting the most conservative degrees of freedom using two-tailed tests at all times).

5.3 Overview of Findings

All 21 participants completed the experiment. Since unequal numbers of subjects greatly complicate the analyses, one of the six participants in the System A (Experimental Condition) group was dropped prior to data analysis. The person chosen to be dropped, per normal custom, was the one who deviated most from the scores of the other five participants for the 24 dependent variables (this person was also the only one who had continual difficulty in understanding the methods).

As discussed above in Section 5.1, the experiment performed is concerned with three primary factors: system, condition, and method. The data collected was analyzed by applying the analysis-of-variance (ANOVA) model to the 24 dependent variables described in Table 3.2-1. The factors were looked at individually, as well as in combination with each other. Of interest in this section are the significant factors and interactions.

Table 5.3-1 presents the 24 dependent variables (rows), the factors (columns), and whether or not the combinations have statistical significance (cells). For example, the probability of the differences in the **number of**

¹⁰ The performance effect evaluated by the analysis of variance (ANOVA) approach primarily used in the data analyses concerned estimates of the variability of scores within particular conditions, the *variances*. The ANOVA method applied to this experiment involved partitioning the overall variability of the 20 participants' scores (for each dependent variable) into the various components shown in Table 3.1-1. The performance effect tested is a ratio of two components: the numerator is the variance-estimate for some experimentally manipulated variable, such as Expert Systems, and the denominator is an appropriate variance-estimate for experimental "error", the amount of variability due to chance factors, such as the Between-Subjects Error. Under the assumption that the two expert systems are really not that different from each other and should not have different effects on performance, the so-called null hypothesis, the variance due to expert systems is accountable by the same random error factors that determine the error variance estimate. Thus, the expected value of this ratio, under the null hypothesis, is 1.0. To the extent that the null hypothesis is false and there really is a performance difference due to the two different expert systems, the ratio of the two variances, called the F-statistic, will be larger than 1. Values of the F-statistic, computed for various numbers of experimental conditions and number of participants (actually, computed for the *degrees of freedom* associated with these numbers), under the null hypothesis assumption, compose what is called the F-distribution. If the experimental F exceeds the tabled value, then the particular effect, Expert Systems in this example, is said to be significant. The null hypothesis of there being no performance effect due to Expert Systems can be rejected. There are different, increasing values of the tabled F corresponding to whether the threshold probability of rejecting the null hypothesis is set at 0.05, at 0.01, or even at 0.001 or smaller. See Winer (1971) for an extended treatment of the Analysis of Variance technique.

Table 5.3-1 Significance of findings for the 24 dependent variables^a

Factors Dependent Variables	Method	Method* Condition	Sys* Meth	Sys* Meth* Cond	Condition	System	Sys* Cond
Total number of responses	.0001	—	.0470	.0002	—	—	—
Number of correctly found problems	.0001	.0001	.0001	—	.0001	.0001	.0109
Percentage of correctly found problems	.0003	.0001	.0001	—	.0001	.0001	—
Total time for the exercise	.0001	—	.0006	.0128	.0257	.0039	—
Total number of problems missed (not found)	.0001	.0001	.0001	—	.0001	.0001	.0053
Total number of false alarms	.0058	.0335	.0003	.0003	.0109	—	—
Total number of true false alarms	.0052	.0099	.0001	.0002	.0058	—	—
Total number of borderline false alarms	.0001	.0187	—	—	.0468	—	—
Total number of true false alarms, low confidence	—	—	—	—	—	—	—
Total number of true false alarms, high confidence	.0010	—	.0005	.0050	.0199	—	—
Total number of borderline false alarms, low confidence	.0044	—	.0044	—	—	.0058	—
Total number of borderline false alarms, high confidence	.0001	—	—	—	—	—	—
Average confidence for correct problems found	.0148	—	—	—	—	.0140	.0310
Average confidence for false alarms	—	—	—	—	—	—	—
Overall average confidence	.0001	—	—	—	—	—	—
Average time for all problems identified	.0001	—	—	—	.0458	—	—
Average time for correct problems identified	.0001	—	.0001	.0024	.0001	—	—
Average time for false alarms identified	.0001	—	.0006	—	—	—	—
Number of hard problems found	.0001	.0434	.0005	—	.0109	.0016	—
Number of medium problems found	.0024	—	.0011	—	.0161	.0161	—
Number of easy problems found	.0001	.0010	—	—	.0002	.0035	.0002
Average time for hard problems identified	—	.0405	—	—	—	.0001	.0085
Average time for medium problems identified	.0241	—	—	—	.0354	.0501	—
Average time for easy problems identified	.0042	.0182	—	—	.0094	.0020	.0009

^a Values are probabilities from ANOVA analysis that the effects could be accountable solely by random factors.

NOTE: "—" Means no statistical significance (p .05)

Legend for Column Headings: Method — main effect of Method; Method * Condition — interaction effect between levels of Method and levels of Condition; Method * System — interaction effect between levels of Method and levels of System; Method * System * Condition — interaction effect between levels of Method, levels of System, and levels of Condition; Condition — main effect of Condition; System — main effect of System; System * Condition — interaction effect between levels of System and levels of Condition.

total responses per method being due to chance is .0001, or 1 in 10,000. In other words, this difference is very statistically significant and is interpreted as being attributable to real differences among the methods. On the other hand, the differences in **average confidence for system and method** (SYS*METH) could be attributable to random events and are not significant. A value is not considered to be statistically significant if the probability of the result being due to chance is greater than .05; a dash ("-") is entered in the cell whenever the probability is greater than .05.

To examine the hypothesis that the methods would assist inexperienced nuclear engineers to perform V&V, several statistical tests (t-tests) were run by grouping the participants' data differently. For one test, the data for the four participants with the most testing experience was compared to the data for the other 16 participants. A second test looked at six participants with testing experience compared to the other 14 participants. The final t-test looked at the 10 participants with the most nuclear experience compared to the rest of the participants. Table 5.3-2 presents the 24 dependent variables (rows), the three t-tests (columns), and whether or not the combinations have statistical significance. Note that none of the cells are highlighted, indicating that results were as expected.

The main dependent variables of interest are number of correctly found problems, percentage of correctly found problems, total time, number of misses, and number of false alarms as discussed in Section 3.2. A false alarm is a participant's response which is not truly a problem. That is, the participant sounded a "false alarm" when no problem existed. The other less important dependent variables will be brought in to explain/support the findings. Appendix B presents all the experimental means (averages) in tabular format. The remainder of this section will be dedicated to explaining the findings summarized in the aforementioned tables.

5.4 Effects of System and Condition

As mentioned in Section 5.3, the experimental results and analysis for system and condition will be discussed below, including explanation of any unexpected results. Figure 5.4-1 presents a graphical depiction of representative experimental results for system and condition, which are discussed below.

5.4.1 Condition Effects

The over-riding concern for this experiment was the effect of condition. How did the groups of participants with a tool perform as compared to the groups without a tool?

Experimental Group Outperformed Control Group. It was hypothesized that the group of participants using a tool (experimental group) would outperform the participants with no tool (control group).¹¹ This was indeed the case. Averaging by condition, the experimental group found almost twice as many correct problems (15.87 as compared to 8.55),¹² a significantly higher percentage of correct problems (73.2% as compared to 45.6%), missed half as many problems (5.5 as compared to 12.9), had almost half as many false alarms (7.6 as compared to 14.7), and performed the

¹¹ While various hypotheses were formulated, the conservative two-tail tests of significance were used to evaluate experimental findings.

¹² All values represent means of the participants' scores, not actual scores.

Table 5.3-2 Significance of findings based on participant experience¹

Participant Groups	Group of 4 (testing) vs. 6	Group of 6 (testing & nuclear) vs. 4	Group of 10 (high nuclear experience) vs. 10
<i>Dependent Variables</i>			
Total number of responses	--	--	--
Number of correctly found problems	--	--	--
Percentage of correctly found problems	--	--	--
Total time for the exercise	--	--	--
Total number of problems missed (not found)	--	--	--
Total number of false alarms	--	--	--
Total number of true false alarms	--	--	--
Total number of borderline false alarms	--	--	--
Total number of true false alarms, low confidence	--	.0265	--
Total number of true false alarms, high confidence	--	--	--
Total number of borderline false alarms, low confidence	--	--	--
Total number of borderline false alarms, high confidence	--	--	--
Average confidence for correct problems found	.0039	.0384	--
Average confidence for false alarms	.0372	.0038	--
Overall average confidence	.0023	.0293	--
Average time for all problems identified	.0126	.0265	--

Table 5.3-2 (Continued).

Participant Groups	Group of 4 (testing) vs. 6	Group of 6 (testing & nuclear) vs. 4	Group of 10 (high nuclear experience) vs. 10
<i>Dependent Variables</i>			
Average time for false alarms identified	--	--	--
Number of hard problems found	--	--	--
Number of medium problems found	--	--	--
Number of easy problems found	--	--	--
Average time for hard problems identified	--	--	--
Average time for medium problems identified	--	--	--
Average time for easy problems identified	--	--	--

¹ Values are probabilities from ANOVA analysis Note: No statistical significance (p<.05)

Figure 5.4-1 Representative Significant Results for System and Condition

Method

Percent Correctly Found Problems

Total Number of False Alarms

Number Correctly Found Problems

Percent Correctly Found Problems

Number Problems Missed

Number Correctly Found Problems

Figure 5.4-1 (Continued)

Figure 5.4-1 (Continued)

exercise in significantly less time (3378 seconds as compared to 4119 seconds) than the control group. At first glance, it might appear that a difference of 741 seconds (12.35 minutes) is not of practical significance in performing these tasks. However, this represents an 18% reduction in the amount of time to perform the task! A difference of 12 minutes for a subset of the rules of an expert system (as used in this experiment) could translate into a savings of many hours when performing V&V on all the rules of a large expert system! This represents a major experimental effect.¹³

5.4.2 System Effects

System B Group Outperformed System A Group. It was hypothesized that the group of participants working with System B would outperform the group working with System A, since System B is less complex in nature (e.g., the rules do not interact with each other). This was indeed the case. Averaging by system, the System B group had a significantly higher percentage of correct problems found (71.7% as compared to 47.2%), missed only one-third as many problems (4.5 as compared to 13.9), found more hard problems (2.2 as compared to 1.4), found more easy problems (3.8 as compared to 2.96), and performed the exercise in significantly less time (3241.5 seconds as compared to 4255.5 seconds, a 23.8% time reduction) than the System A group.

While the System A group found more correct problems (14.4 vs. 10.02), there were more System A problems to be found than for System B (due to the greater number seeded -- 20 for syntax checking, and 21 for knowledge checking, vs. 11 and 11 -- and due to latent errors described in Section 5.5.1). Therefore, percentage of correctly found problems better permits comparison of performance success for Systems A and B. The System A group also found more medium problems than the System B group at 4.36 as compared to 3.1. This is likely a result of System A finding more correct problems than System B. However, the ability to find medium problems is not quite as important as the ability to find hard problems. As mentioned above, the System B group found more hard problems than the System A group. Once again, this result has to be interpreted in light of the differences between Systems A and B: "hard" System B problems were not really as hard, in some absolute sense, as hard System A problems.

One way of summarizing these findings is to say that System B was an easier, simpler system to use than System A for the four methods tested in the experiment. There is very little practical significance of this finding. However, the system differences are important to this experiment because they show that the two chosen expert systems were, in behavioral terms, quite different from each other. They therefore, provide a basis for interpreting the general utility of the methods -- e.g., if a tool facilitates performance only for System A and not for System B, one explanation is that the tool has its effect only for the more complex systems.

5.4.3 System and Condition Effects

System A Group with Tools Significantly Outperformed System A Group without Tools. The experimental group for System A significantly outperformed the System A control group for all six of the statistically significant dependent variables: the System A experimental group found almost twice as many correct problems (19.25 as compared to 9.55), missed half as many problems (8.85 as compared to 18.95), found twice as many easy problems

¹³ The experimental hypothesis, that tools will lead to improved task performance, was supported by these results.

(4.13 as compared to 1.8), had higher confidence (on a scale of 1 to 5) in the correct problems they found (4.57 as compared to 4.12), took much less time to find hard problems (2536 seconds as compared to 4840 seconds), and took only one-third as much time to find easy problems (1371 seconds as compared to 3906 seconds) as the System A control group.

System B Group with Tools Outperformed System B Group without Tools. The experimental group for System B significantly outperformed the System B control group. However, the magnitude of the differences were not as great. The System B experimental group found significantly more correct problems (12.5 as compared to 7.55), missed only one-third as many problems (2.15 as compared to 6.85), and took less time to find hard problems (970.5 seconds as compared to 1094 seconds) than the System B control group. However, the control group for System B had higher confidence in their correct problems (4.78 as compared to 4.62), and took much less time to find easy problems (1106 seconds as compared to 1517 seconds, a 27% time reduction) than the System B experimental group. It is hypothesized that the control group had higher confidence in their correct responses since they had identified the problems themselves. The experimental group had expressed some skepticism at how easy it was to find the errors and that they felt like they were "cheating". This could also explain the reduced time for the control group to find easy problems. When the experimental group found easy problems, they did not quite believe it, so they checked further.

Mixed Results for System A Group with Tools Compared to System B Group with Tools. The System A experimental group found significantly more correct problems (19.25 as compared to 12.5) than the System B experimental group. The System B group missed significantly fewer problems, however, at 2.15 as compared to 8.85. The System A experimental group found slightly more easy problems (4.13 as compared to 3.8). The experimental group for System B had slightly higher confidence in their correct problems (4.62 as compared to 4.57). The System B group took significantly less time to find hard problems at 970.5 seconds as compared to 2536.66 seconds for the System A experimental group. However, the System A group with the tool took less time to find easy problems (1371.5 seconds as compared to 1517.2 seconds) than the System B tool group. These results are not conclusive, but in general they indicate that the tools facilitated performance for System A to a greater degree than System B.

The results for system and condition can be summarized as follows: the System A participants with the tools dramatically outperformed the System A participants without tools; the System B participants with the tools outperformed the System B participants without tools; and there was a trend towards better performance for the System A participants with the tools as compared to the System B participants with the tools. This is due to the greater complexity of System A. The tools helped more with the more complicated of the two systems.

It should be remembered that the results of this section concern the aggregate findings for all tool vs. no-tool conditions for the four methods. If tools differ in effectiveness across the four methods -- as is shown below -- then this can help explain the variability in the tool vs. no-tool comparisons.

5.4.4 Method

Methods one and two, grouping of requirements and requirements tracing, are related but cannot be compared. They each have different objectives and it would not make sense to compare the results of these two methods. However, methods three and four, syntax checking and knowledge checking, are well suited for comparison. Although method

three involves syntactical checks as compared to semantics checks for method four, both methods basically target the same error groups (see Section 2.4.3).

Method Four Superior to Method Three. The participants using the method four knowledge checking tool found significantly more correct problems (10.95 as compared to 8.95), found a higher percentage of correct problems (69.13% as compared to 58.61%), missed many fewer problems (5.8 as compared to 8.15), had fewer false alarms (12.15 as compared to 15.35), and took less time to complete the exercise (3981 seconds as compared to 4509 seconds, an 11.7% time reduction) than the participants using method three. This improvement in performance represents a major experimental finding. It may well be due to the fact that the knowledge checking tool, Meta-Check, was much more precise in pinpointing the exact location of the error and describing the problem.

It should be noted that had there been a general practice effect carrying over between the Method 3 and Method 4 Sessions (that is, if participants had performed better on knowledge checking due to learning that took place during syntax checking), it should effect both the control and experimental groups. As can be seen from Figure 5.4-1, however, the control group (no tool) performed only slightly (and insignificantly) better at knowledge checking than syntax checking (12% improvement in number of correctly found problems, 9% increase in % correctly found problems, for example). This does not support a learning hypothesis. On the other hand, the improvement for the experimental group (with tool) was significant, with a 31% improvement in number of problems found, and 25% improvement in % correctly found problems (to list a few) when going from syntax to knowledge checking.

Since the experimental group had to learn how to use a completely different tool, this could have been a source of negative transfer, and performance could well have suffered. The facts are that the tool condition significantly improved from the VERITE' tool to the Meta-Check tool, while the no-tool condition -- which essentially did the same thing in both sessions -- did not improve. The clearest interpretation of this pattern of findings is twofold: 1) there was not a learning effect between the two sessions, and 2) the improvement in the tools group's performance is due to Meta-Check being a more effective tool to use than VERITE'.

5.5 Requirements Grouping and Tracing Findings

As mentioned in Section 5.3, the experimental results and analysis for Requirements Grouping and Requirements Tracing will be discussed below including explanation of any unexpected results. Figures 5.5-1 and 5.5-2 present a graphical depiction of representative experimental results for requirements grouping and requirements tracing which are discussed below.

5.5.1 Requirements Grouping Findings

The effects of condition are of primary interest. How did the group with a tool (experimental group, condition 1) perform as compared to the group with no tool (control group, condition 2)?

Participants with Tool Outperformed Participants without Tool. Averaging over both systems, the participants using the tool (experimental group) had significantly more mean correct responses (32.0 compared to 11.3) and significantly fewer false alarms (5.3 compared to 20.6) than the participants without the tool (control group). The participants using the tool had a significantly higher mean percentage of correct problems of 93.47% as compared to

Figure 5.5-1 Representative Significant Results for Requirements Grouping

Figure 5.5-2 Representative Significant Results for Requirements Tracing

30.66% for the participants without the tool. Similarly, the participants using the tool missed a mere 1.5 problems as compared to the mean of 22.2 problems missed by the participants with no tool. This suggests that the tool made them much more effective in grouping requirements.

That the total time for the exercise was not statistically significant is due to the fact that almost all the participants, with or without the tool, used the entire amount of time allotted. It is hypothesized that since this was the first method used during the 3 day experiment, participants were particularly serious and dedicated, and spent time double-checking for errors. Had considerably more time been allocated for this exercise then it is likely that the experimental group would have stopped work sooner than the control group.

Method Assisted System A Participants More Than System B Participants. The participants grouping requirements for System A had significantly more correct responses (30.9 as compared to 12.4), and a much higher percentage of correct responses (70.22 as compared to 53.91) than the participants working on System B. The System A participants also had significantly fewer false alarms (6.3 as compared to 19.6). This seems to indicate that the tool helped the System A participants more than the System B participants. However, the System B participants took marginally less time to complete the task (1776 seconds as compared to 1920 seconds, a 7.5% time reduction), and missed slightly fewer problems (10.6 as compared to 13.1).

One reason for the higher number of correct problems for System A participants deals with total number of problems existing in each system. Although the same number of errors (problems) were seeded for both systems, there appeared to be more latent errors in System A. Recall that an early version of System A was used for the experiment. This version had not previously been V&Ved. When the participants' sheets were scored, each response was evaluated for validity. If a response was found to be a true problem even though it was not seeded by SAIC, a so-called "latent error", it was added to the master list of problems that should be found. Since more of these were identified for System A than for System B, there were many more problems to be found for System A than B, thus explaining the number of correct responses being higher. The latent errors were determined to be of the same types as those seeded. No further analysis of these latent errors was performed. Since the seeded errors were more numerous than the latent and, since the latent were the same types as the seeded, it is concluded that the tools were quite effective in supporting error detection.

The fact that there were more problems to be found also explains why it took the System A group longer to complete the task. That is, the System A group had more work to do than the System B group, thus taking longer. The System A group had more total responses (37.2 as compared to 32.4) and far fewer true false alarms (2.3 as compared to 16.7) than the participants for System B. This implies that perhaps the System A participants had a better understanding of the task that needed to be accomplished than their counterparts for System B.

Systems A and B Experimental Groups Outperformed Control Groups. The participants with the tool for System A found significantly more correct problems (44.0 as compared to 17.8) and a much higher percentage of correct problems (20.0 as compared to 4.8) than the participants with no tool. This implies that the tool made them more effective in grouping requirements. Similarly, the participants with the tool for System B found significantly more correct problems (20.0 as compared to 4.8) and a much higher percentage of correct problems (86.95 as compared to 20.87) than the participants with no tool. The System B participants with the tool also had significantly false alarms, with only 4.0 as compared to 35.2.

As discussed above, the total time for the task was not statistically significant. Also, the number missed was not statistically significant.

The System A participants with the tool had more false alarms at 6.6 as compared to 6.0 for the System A group with no tool. However, not only was this difference not significant, but this should be discounted in any event since most of these were not true false alarms (the group with the tool had 1.0 true false alarms as compared to 3.6 for the non-tool group). Therefore, most of these false alarms were borderline and should be treated as an indication of the tool's effectiveness. Borderline false alarms are problems identified by the participants which were not seeded by SAIC, but when evaluated were not obviously correct problems and could not be dismissed as non-problems either.

System A Experimental Group Outperformed System B Experimental Group. The participants with the tool for System A found significantly more correct problems (44.0 as compared to 20.0) and a higher percentage of correct problems (100.0 as compared to 86.95) than the participants with the tool for System B. This implies that the tool was more helpful for System A (as discussed above). Just as the System A participants with the tool had more false alarms than the non-tool group, they also had more false alarms than the System B participants with the tool (6.6 as compared to 4.0).

The results for Requirements Grouping can be summarized as follows: the participants with keyword printouts performed grouping of requirements better than the participants without the tool; the System A participants with the tool performed better than the System A participants without the tool; the System B participants with the tool performed better than the System B participants without the tool; and the System A participants with the tool outperformed the System B participants with the tool.

5.5.2 Requirements Tracing Findings

The effects of conditions are of primary interest. How did the group with a tool (experimental group, condition 1) perform as compared to the group with no tool (control group, condition 2)?

Participants with Tool Marginally Outperformed Participants without Tool. Averaging over both systems, the participants using the tool found slightly more correct problems (7.8 as compared to 6.8) and had a mean percentage of correct problems of 50.47% as compared to 45.4% for the participants without the tool. Similarly, the participants using the tool missed only 10.7 problems as compared to the mean of 11.3 problems missed by the participants with no tool. This suggests that the tool made them slightly more effective in tracing requirements. It is interesting to note that the total time difference for the exercise was not statistically significant. This is due to the fact that almost all the participants, with or without the tool, used the entire amount of time allotted. It is hypothesized that since this was the first method used during the 3 day experiment, participants were particularly serious and dedicated, and spent time double-checking for errors. Also, it is highly likely that if more time had been allocated the effects of total time for the exercise would probably have been apparent.

The participants with the tool had slightly more total false alarms than the non-tool group (4.9 as compared to 3.3). It should be noted that the experimental group (participants with the tool) had lower confidence in these false alarms. The average confidence per false alarm for the experimental group was 4.11, as compared to their confidence in

correct problems of 4.52. In contrast, the control group (participants with no tool) had a confidence of 4.5 in their false alarms with a confidence of only 4.29 in correct problems.

Results Based on System. The participants who were tracing requirements for System A had significantly more correct responses (8.8 as compared to 5.8) than the participants working on System B. The System A participants also had far fewer false alarms (2.7 as compared to 5.5). The System A participants had far fewer true false alarms (2.1 as compared to 5.5). They also took marginally less time to complete the task at 4536 seconds as compared to 4776 seconds. This seems to indicate that the method helped the System A participants more than the System B participants. However, the System B participants had a significantly higher mean percentage correct at 64.44% as compared to 31.42% for System A. Also, the System B group missed far fewer problems at only 2.8 (as compared to 19.2). It was anticipated that the System B participants would perform better on this task due to System B being a less complex system than System A.

One reason for the higher percentage of correct problems for System A participants has to do with the total number of problems existing in each system. Although the same number of errors (problems) were seeded for both systems, there were more latent errors in System A (7 vs. 2). These latent errors were of the same types as those seeded. When the participants' sheets were scored, each response was evaluated for validity. If a response was found to be a true problem, even though it was not seeded by SAIC, it was added to the master list of problems that should be found. Since so many of these were identified for System A, there were many more problems to be found for System A than B (28 for System A, 9 for System B). Although the System A participants found more correct problems than the System B group, the percentage was lower.

The fact that there were more problems to be found also explains why the System A group missed more problems during requirements tracing. That is, with such a larger number of problems to find, it only makes sense that the System A group had a higher number missed.

Results Based on System and Condition. The participants with the tool for System A found a higher percentage of correct problems (34.28 as compared to 28.57) and missed fewer problems (18.4 as compared to 20.0) than the participants with no tool. This implies that the tool made them more effective in tracing requirements. Similarly, the participants with the tool for System B found a slightly higher percentage of correct problems (66.66 as compared to 62.22) than participants with no tool. As discussed above, the total time for the task was not statistically significant. Also, the number of correct problems found was not statistically significant.

The System A participants with the tool had marginally more false alarms at 2.8 as compared to 2.6 for the control group (no tool). However, this should be discounted since most of these were borderline false alarms (the experimental group had 1.0 borderline false alarms vs. 0.2 for the control group). Therefore, most of these false alarms were borderline and should be treated as an indication of the tool's effectiveness. The experimental condition for System B also had more false alarms than the control condition (7.0 vs. 4.0). It should be noted that the experimental group (participants with the tool) lacked confidence in these false alarms. The average confidence per false alarm for the experimental group was 4.13 as compared to their confidence in correct problems of 4.69. In contrast, the control condition (participants with no tool) had a confidence of 4.62 in their false alarms with a confidence of only 4.5 in correct problems! That is, the control group indicated that they were very confident (4.62 on a scale of 5) in the answers

they got wrong -- in fact, more confident than they were in the answers they got right (4.5 on a scale of 5)! The System B group also missed slightly more problems than the control group at 3.0 as compared to 2.6.

The participants with the tool for System B found a significantly higher percentage of correct problems (66.66% as compared to 34.28%) and missed fewer problems (3.0 as compared to 18.4) than the System A participants with the tool. This implies that the tool was more helpful for System B. This is consistent with the observation that, while System B is simpler than System A, the requirements tracing activity was somewhat more complicated. The System B participants with the tool did have more total false alarms (7.0 as compared to 2.8) and true false alarms (7.0 as compared to 1.8) than the System A tool group.

The results for requirements tracing can be summarized as follows: the experimental group (with tool) outperformed the control group (no tool); the results based on system were not conclusive with the System A group performing better on some dependent variables and the System B group performing better on others; and similarly the results were not conclusive based on system and condition.

5.6 Syntax Checking Findings

Figure 5.6-1 presents a graphical depiction of representative experimental results for syntax checking which is discussed below.

Participants with Tool Outperformed Participants without Tool. The participants using the tool found more correct problems (10.3 as compared to 7.6) and had a significantly higher mean percentage of correct problems of 66.3% as compared to 50.92% for the participants without the tool. Similarly, the participants using the tool missed only 6.7 problems as compared to the mean of 9.6 problems missed by the participants with no tool. The experimental group also had slightly fewer false alarms at 14.1 as compared to 16.6 for the control group. The participants with the tool found 3.2 hard problems as compared to 1.8 for the group with no tool. Similarly, the tool group found a mean of 4.2 problems as compared to 2.6 for the group without a tool. This suggests that the tool made the participants more effective in checking the syntax of the rulebase.

The total time for the exercise was not statistically significant. Going back to the raw data for the participants, it is apparent that time to complete the task varied greatly within the groups. For example, in the control group one participant finished in only 1920 seconds, while another took 8460 seconds. Similarly, in the experimental group one participant finished in 1740 seconds and another took 7200 seconds. It is hypothesized that this lack of statistical significance is attributable to individual's attention to detail. That is, given a group of individuals, some will pay more attention to detail in performing a task than others, independent of the method being used or the condition (tool versus no tool).

Method Assisted System B Participants More than System A Participants. The participants checking the syntax of rules for System B had more correct responses (10.3 as compared to 7.6) and a significantly higher percentage of correct problems found (79.23% as compared to 38%) than the participants working on System A. The System B participants had far fewer problems missed at 3.2 as compared to 13.1 for System A. They also took much less time to complete the task at 3732 seconds as compared to 5286 seconds. Further, the System B participants found more hard syntax problems than the System A group (3.0 as compared to 2.0), and more medium syntax problems (3.1 as

Figure 5.6-1 Representative Significant Results for Syntax Checking

compared to 2.1). This seems to indicate that the method helped the System B participants more than the System A participants, likely due to System B being less complex than System A.

However, the System B participants had a higher number of false alarms (16.3 as compared to 14.4), and a higher number of true false alarms (16.3 as compared to 13.6). This is puzzling, as it was anticipated that the System B participants would perform better on this task due to System B being a less complex system than System A. The System B participants had a mean of 26.6 responses as compared to 22.0 for the System A group, so that the higher false alarm rate is associated with a higher overall response rate. Interestingly, it was the System B group which used the syntax-checking tool that had the higher false alarms and total responses, but this pattern disappeared for the knowledge-checking tool. Perhaps the sometimes confusing output from the symptom-based syntax-checking tool led these participants to suspect other, similar, rule characteristics as being erroneous, and they didn't trust the tool to have detected them. However, no comments or questions by these participants shed any light on this finding.

Mixed Results Based on System and Condition. The participants with the tool for System A found many more correct problems (9.6 as compared to 5.6) and a much higher percentage of correct problems (48.0 as compared to 23.0) and missed far fewer problems (10.8 as compared to 15.4) than the System A participants with no tool. They also took much less time to complete the task at 3600 seconds as compared to 6972 seconds. This implies that the tool made the System A group more effective in checking for syntax errors. Similarly, the participants with the tool for System B found more correct problems (11.0 as compared to 9.6) and a higher percentage of correct problems (81.54 as compared to 70.76) and missed fewer problems (2.6 as compared to 3.8) than the System B participants with no tool. Also, total false alarms, hard problems, and medium problems were not statistically significant.

The participants with the tool for System B found slightly more correct problems (11.0 as compared to 9.6), a significantly higher percentage of problems (81.54% as compared to 48%), and missed far fewer problems (2.6 as compared to 10.8) than the System A tool participants. This implies that the tool was more helpful for System B. The System B participants with the tool did take slightly longer to complete the task (3840.0 seconds as compared to 3600.0 seconds) than the System A tool group.

The results for syntax checking can be summarized as follows: the experimental group outperformed the control group; the System B participants outperformed the System A participants; and the results for system and condition were not conclusive, with the System A and B experimental groups performing better on some dependent variables while the control groups performed better on others.

5.7 Knowledge Checking Findings

As mentioned in Section 5.3, the experimental results and analysis for knowledge checking will be discussed below, including explanation of any unexpected results. Figure 5.7-1 presents a graphical depiction of representative experimental results for knowledge checking which is discussed below.

Participants with Tool Significantly Outperformed Participants without Tool. Averaging by condition (tool versus no tool), the participants using the tool found significantly more correct problems (13.4 as compared to 8.5) and had a significantly higher mean percentage of correct problems of 82.85% as compared to 55.42% for the participants without the tool. Similarly, the participants using the tool missed a mere 3.1 problems as compared to the mean of 8.5 problems missed by the participants with no tool. The experimental group also had fewer false alarms at only 6.0 as

Figure 5.7-1 Representative Significant Results for Knowledge Checking

compared to 18.3 for the control group. The participants with the tool found 2.9 hard problems as compared to 2.3 for the group with no tool. The tool group also found more easy problems than the non-tool group (5.4 as compared to 3.4). The average time to find a hard problem was much lower for the participants with the tool (1847 seconds as compared to 2331.6, a 20.7% time reduction) than for the participants with no tool. Also, the participants with the tool took much less time to find easy problems (712.8 seconds as compared to 1824.1 seconds) than the non-tool participants. All this data strongly demonstrates that the tool made the participants much more effective in performing knowledge checking for the rulebase.

The total time for the exercise was not statistically significant. Going back to the raw data for the participants, it is apparent that time to complete the task varied greatly within the groups. For example, in the control group, one participant finished in only 1380 seconds, while another took 7500 seconds. Similarly in the experimental group, one participant finished in 1200 seconds and another took 6000 seconds. As for syntax checking, it appears that both the control and the experimental groups contained individuals who varied greatly in their attention to detail, causing large within-group variability that could mask other effects.

Method Assisted System B Participants Significantly More than System A Participants. Examining the averages for system, the participants performing knowledge checking for System B had more correct responses (11.6 as compared to 10.3) and a much higher percentage of correct problems found (89.22% as compared to 49.04%) than the participants working on System A. The System B participants had significantly fewer problems missed at 1.4 as compared to 10.2 for System A. They also took much less time to complete the task at 2682 seconds as compared to 5280 seconds. Further, the System B participants found more hard problems than the System A group (3.6 as compared to 1.6). This convincingly demonstrates that the method helped the System B participants more than the System A participants.

However, the System A participants found slightly more medium problems than the System B participants (4.6 as compared to 3.3). This can be explained by looking at the number of correct responses. The System B participants found only 1.3 more correct problems than the System A participants. The System B group found 2.0 more hard problems than the System A group. If the System A participants found significantly more easy problems than the System B participants, then it might follow that System B participants found more medium problems. However, the number of easy problems found was not statistically significant. Therefore, it makes sense that the System A group found more medium problems. Note that this dependent variable (number of medium problems found) is not really important. Of real interest is the ability to find hard problems. As mentioned above, the System B group did find significantly more hard problems than the System A group.

Both System A and B Groups with Tool Outperformed Non-Tool Groups. The participants with the tool for System A found significantly more correct problems (13.8 as compared to 6.8) and a much higher percentage of correct problems (65.7 as compared to 35.38) and missed far fewer problems (6.2 as compared to 14.2) than the participants with no tool. They also took less time to complete the task at 4944 seconds as compared to 5616 seconds. The participants with the tool for System A also had far fewer false alarms at 8.0 than the control group at 32.8. The experimental group for System A also found more hard problems (1.8 as compared to 1.4) and twice as many medium problems (6.2 as compared to 3.0) as the control group did. This implies that the System A group with the tool was more effective in knowledge checking than the group with no tool. Similarly, the participants with the tool for System B found more correct problems (13.0 as compared to 10.2) and a much higher percentage of correct problems (100 as

compared to 78.44) and missed fewer problems (0.0 as compared to 2.8) than the participants with no tool. The experimental group for System B also found more hard problems (4.0 as compared to 3.2) and more medium problems (4.0 as compared to 2.6) as the control group did. This implies that the System B group with the tool was more effective in knowledge checking than the group with no tool.

The System B experimental group took longer to complete the task than the control group (2928 seconds as compared to 2436 seconds). This should be viewed in light of the fact that the experimental group for System B had a lower average time per correct response (225.23 as compared to 297.7). It is hypothesized that it took the experimental group slightly longer than the control group because they had more responses (17 as compared to 14).

Meta-Check Helped the System B Group More than the System A Group. The System B experimental group found a significantly higher percentage of correct problems (100% as compared to 65.7%), took significantly less time (2928 seconds as compared to 4944 seconds), and missed significantly fewer problems than the System A experimental group (0.0 as compared to 6.2). They had half as many false alarms at 4.0 as compared to 8.0. The System B experimental group found many more hard problems (4.0 as compared to 1.8), but found fewer medium problems (4.0 as compared to 6.2). The experimental group for System A found slightly more correct problems than the System B experimental group (13.8 as compared to 13.0). There ended up being a total of 21 possible problems to be found for System A as compared to only 13 for System B. Therefore, the number of correctly found problems should be discounted, and the percentage of correctly found problems should be weighed more heavily.

The results for knowledge checking can be summarized as follows: the participants with Meta-Check performed knowledge checking better than the participants without the tool; the participants performing knowledge checking for System B (regardless of tool/no tool) outperformed the System A participants; the System A participants with Meta-Check performed better than the System A participants without the tool; the System B participants with Meta-Check performed better than the System B participants without the tool; and the System B participants with Meta-Check outperformed the System A participants with Meta-Check.

The presence of latent errors, errors present in the knowledge base but not deliberately seeded, (7 such for System A, 2 for System B) did not appear to bias the results. Participants performed equally for latent and seeded errors. This is to be expected, since the latent errors were exactly of the same classification types as those seeded, except none were of the "hard" variety.

5.8 Findings Based on Participants' Experience

The effects of the participants' backgrounds and experience on their results is of interest. Particularly, how did the tool assist participants with a large amount of testing experience but not much nuclear engineering experience? To answer this question, the participants' results were subsequently analyzed using student t-tests. Three sets of tests were run. The first set of t-tests examined a group of four subjects who all had testing experience, but little or no nuclear engineering experience. All four participants came from the experimental (tool) condition. They were compared to the remaining members of the experimental group (6 participants). Next, two members of the experimental group who had testing experience, but also had nuclear engineering experience were added to the group, increasing it from 4 to 6. These 6 participants' scores were compared to the remaining 4 members of the experimental group. Finally, the 10 participants with the least amount of nuclear engineering experience were compared to the 10 having the most

experience. This was based on years of nuclear engineering experience. The "low experience" group had between 0 and 4 years of nuclear experience for an average of 2.4 years of experience. The "high experience" group had between 7 and 22 years of experience for an average of 13.6 years of experience. In view of the small number of participants involved, all of the following results should be viewed with caution.

5.8.1 Testing Experience with No Nuclear Experience

The first t-test run was to compare the four participants with testing experience to the remaining 6 members of the experimental group. The four participants with high testing experience were:

Subject 46 Silver
Subject 39 Silver
Subject 56 Iron
Subject 88 Iron

The t-tests were run on all 24 dependent variables described in Table 3.2-1. The major dependent variables of interest (e.g., number of correctly found problems, percentage of correctly found problems, and number missed) were not statistically significant for this student t-test. Therefore, testing experience per se is not a predictor of the ability to profit from the tools.

5.8.2 Testing Experience with Some Nuclear Experience

The second t-test run was to add two participants with testing experience and some nuclear experience to the original four participants and compare these 6 to the remaining 4 members of the experimental group. The six participants with testing experience and some nuclear experience were:

Subject 46 Silver
Subject 39 Silver
Subject 56 Iron
Subject 88 Iron
Subject 25 Iron
Subject 75 Silver

The t-tests were run on all 24 dependent variables described in Table 3.2-1. Again, the major dependent variables of interest (e.g., number of correctly found problems, percentage of correctly found problems, and number missed) were not statistically significant for this t-test.

5.8.3 Importance of Significant Nuclear Experience

The third t-test run was to determine if participants with a strong background in nuclear engineering performed better on the tasks than those with less nuclear experience, based on years of nuclear engineering experience. The "low experience" group had between 0 and 4 years of nuclear experience for an average of 2.4 years of experience. The "high

experience" group had between 7 and 22 years of experience for an average of 13.6 years of experience. The ten participants with 7-22 years of nuclear engineering experience were:

Subject 25 Iron
Subject 75 Silver
Subject 62 Silver
Subject 54 Gold
Subject 73 Copper
Subject 43 Gold
Subject 67 Iron
Subject 51 Copper
Subject 44 Iron
Subject 65 Gold

Note that the 10 participants are evenly distributed between experimental and control conditions (5 from each). Although this was not intentional (only nuclear engineering experience was used to group participants), it makes the results more defensible. Had all the participants with lower nuclear engineering experience been in the experimental group, one could argue that the tools helped them outperform the other participants.

The t-tests were run on all 24 dependent variables described in Table 3.2-1. There was no statistical significance found for any of the 24 dependent variables. This is a major finding of the experiment. It suggests that verification and validation activities for nuclear expert systems might be performed just as well by engineers with 0 - 4 years of experience as by an engineer with 22 years of experience. This discovery should be capitalized on by the nuclear industry to lower the costs of performing V&V on expert systems.

The results based on participant experience can be summarized as follows: the participants with testing experience but no nuclear experience did not significantly outperform the remainder of the experimental group; the participants with testing experience and some nuclear experience did not significantly outperform the remainder of the experimental group. Finally, the participants with 7 - 22 years of nuclear experience did not outperform the participants with only 0 - 4 years of experience.

5.9 Undetected Errors

Not all the errors seeded by the experimenters were detected. As one would expect, the undetected errors were in the more complex of the two systems, System A, and were for methods 3 and 4. These errors were categorized as "hard to detect" problems. For method 3, syntax checking, three errors went unnoticed. The first error involved substituting "RPV Water Level" for "S/Pool Water Level". This error, though not detected by VERITE, would have been detected by Meta-Check. There were no obvious clues from the dimensions or wording of the variable or reference, but a nuclear engineer very familiar with BWRs could detect this error since "RPV Water Level" is never compared to "suppression pool load limit". Similarly, the second error could have been caught by a highly experienced nuclear engineer. There were no clues to this error in previous rules but it would have been discovered by Meta-Check. This second error substituted "NPSH Limit" (Net Positive Suction Head) for "TAF" (Top of Active Fuel). However, the third error, which involved the removal of an IF clause "RCIC is not running", would not have been caught by Meta-Check and would require an expert on EOPs to detect.

For method 4, knowledge checking, no errors went undetected. However, two of the hard problems found were not correctly categorized (cause ID) by any of the participants. That is, the participants wrote down the correct rule ID, but wrote down a cause ID that was unequivocally wrong. This shows that the participant did not really catch these hard errors. The first error involved deleting the second THEN clause "set caution 6". This would not have been detected by Meta-Check. There were no obvious clues from the dimensions or wording of the variable or reference. Only an expert on EOPs could catch this error. The same holds true for the second error which deleted the 3rd IF clause "H2 Recombiner Max Operating Limit Has Been Exceeded." As discussed in Section 2.2, hundreds of dynamic test cases may have to be developed and run in order to find these errors. So it seems practical that instead, a new static tool or method should be devised to detect these types of errors. Even given an unlimited amount of time to analyze the rules, it is highly unlikely that these types of errors would have been detected without such a new tool.

5.10 Participant Comments

Participants in the experiment were afforded two opportunities to provide written comments. At the end of the requirements tracing phase of the experiment participants were given forms which they had the option to fill out with their comments on the grouped requirements and requirements tracing methods. Also, at the end of the experiment, the participants were encouraged to write and submit comments about any facets of the experiment. This section summarized those participant comments that addressed the methods used for knowledge base certification. A total of 22 separate comment sheets were provided by the participants after the requirements tracing phase and at the end of this experiment. The written comments varied in length from two sentences to six pages. In addition, verbal comments are also incorporated in this section.

The experimental group consistently expressed an appreciation of the requirements grouping and tracing aids as a means of greatly simplifying these tasks. They liked the way that the aid was organized in that all the materials needed for requirement tracing were presented on a single sheet of paper. In fact, several of the Requirements Tracing tool subjects said they had a hard time coming to believe that the difficult tracing activity they learned in the training session was made such an easy task by the SuperTrace tool. Such incredulity might explain some of the marginal effects of this tool! In contrast, the control group repeatedly complained about the tediousness of the standard practice regarding requirements tracing which involves a complicated series of steps in looking up information from different documents. A number of participants described the requirements tracing process and training as "very intense", "challenging", and demanding considerable concentration. A common thread that was apparent in many comments on all phases of the experiment was that the participant wanted to know more about the overall purpose of the experiment in a "big picture" sense rather than the details and mechanics of the process.

The technical engineering background of most of the participants turned out to be a mixed blessing during the experiment. Many comments were made regarding the technical accuracy of nomenclature used in the knowledge bases. Suggestions were put forth as to how the rules could be rewritten to better perform their function. Some participants spent time looking too deeply into the knowledge base and basis rather than following their instructions for requirements tracing, syntax checking, or knowledge checking.

Comments were all favorable to the organization and conduct of the experiment, but varied as to the depth of training. Some participants felt that more time should have been spent on training the group to perform the experiment, with more problems being worked out together. Others believed that too much training and "hand holding" had occurred thus detracting from the time allowed for the actual experiment. It was evident from the nature of verbal remarks made

during and after the experiment that individuals in the experiment exhibited a wide range of comfort with the subject matter and tasks of this experiment. Considering the diverse background of the group, this is a reasonable reaction. Most written and verbal comments indicated that many applicants learned a great deal about expert systems and verification and validation.

6 CONCLUSIONS AND RECOMMENDATIONS

This section provides an overall summary of the knowledge base certification experiment with all the important results and insights from the statistical analysis. First, the effects of condition (i.e., performance of the experimental group versus the control group) are discussed. Then, the effects of the two expert systems are addressed followed by a summary of the effect of using each of the four methods or tools that were developed for this experiment. Finally, the key conclusions from this task and their relevance to the project are presented in this section.

6.1 Condition Effects

The performance of the experimental groups (using tools) exceeded that of the control group by almost a factor of two in most important measures such as number of errors detected, number of errors missed, and number of false alarms. In addition, the experimental group completed its tasks in 20% less time than the control group. Thus, we have the most important finding of the experiment: effective static-testing tools can greatly improve the quality and productivity of V&V.

6.2 System Effects

Due to its relatively lower complexity, the System B groups outperformed the System A group in detecting all types of errors and using less time to detect them. For different parameters, the System B groups' performance was anywhere from 20% to 300% better than that of System A. Systems will differ greatly in complexity, and performance will be more effective with the simpler systems, tools or no tools.

6.3 Comparison of Tool-to-Tool Effects

The only meaningful comparison between tools or methods is that of syntax checking vs. knowledge checking because these two methods are searching for the same types of errors in the knowledge base. For key performance measures, the knowledge checking tool enhanced the participants' ability to correctly detect errors anywhere from 12% to 40% over that of the syntax checking tool. Both tools improved the performance of the experimental group over that of the control group.

The knowledge checking tool, with its pinpointing of error location and type, is clearly a more efficient technique.

6.4 Individual Method Experimental-to-Control Group Effects

For requirements grouping, the experimental group with the tool outperformed the control group in key parameters by a factor of 3 to 15. This large difference underscores the usefulness of the requirements grouping tool.

In the case of requirements tracing for both expert systems, the tool resulted in a small benefit (on the order of-about 10%) to the experimental group as compared to the control group. More dramatic increases in performance were calculated for an individual system for certain parameters, but the overall effect for both systems was not as pronounced as observed for requirements grouping.

The syntax tracing tool experimental groups exhibited a 15 - 80% improvement in key performance parameters for both systems as compared to the control group. This included such measures as total errors detected, false alarms, number of hard errors detected, and number of errors missed.

The knowledge checking tool showed a significant improvement in performance in comparing the experimental and control groups. Performance parameters improved anywhere from 20% for the average time to find a hard problem to 300% for the number of false alarms. Both the total number of defects detected and the time to find easy problems improved by greater than 50%.

6.5 Participants' Experience Effects

The testing and/or expert system experience of the participants was not found to have any statistically significant effect on their performance in detecting knowledge base defects for all four methods. The number of years of nuclear engineering subject matter experience, anywhere from 0 to 22 years, also had no statistically significant impact on the participants' ability to detect defects. This second result indicates that less experienced staff could perform as well as more senior staff in certifying a knowledge base. It should be noted, however, that 95% of the participants had a college degree in a technical area. The other participant was also thoroughly trained and experienced in the area.

6.6 Conclusions

Based on a statistical analysis of the knowledge base certification experiment using new tools and traditional methods for requirements grouping, requirements tracing, syntax checking and knowledge checking, the following conclusions are made:

- 1) The use of tools or aids for use in static knowledge base certification results in a significant improvement in detecting all types of defects (including hard-to-find problems), avoiding false alarms, and completing this effort in less time.
- 2) Of the four tools developed and used in this KB certification experiment, knowledge checking and requirements grouping showed the greatest improvement in performance.
- 3) The most dramatically effective single tool was that for knowledge checking, Meta-Check.
- 4) The syntax and knowledge checking tools led to the detection of the majority of errors in the rule bases (66% for System A, 83% for System B). Static testing methods, therefore, offer a highly effective -- and lower cost -- initial alternative to the traditionally-preferred dynamic testing methods which involve actual execution of system rules with "live" data and experienced professionals as observers judge performance accuracy ("oracles"). Note that in practice multiple methods would be applied sequentially (such as in the Guideline Packaegs G, H, and I in Section 2, Volume 7.) With multiple methods, if one method does not detect a fault then there is a chance that another method will.

- 5) Testing experience and extensive (7 to 22 years) subject matter experience did not improve the performance of the knowledge base certification participants as compared with less (2 to 4 years) subject matter experience. (It should be noted, however, that all of the participants had either a college degree in a technical field or a high level of training and experience.) This discovery should be capitalized on by the nuclear industry to lower the costs of performing V&V on expert systems.
- 6) Even with the tools and aids provided, some hard "seeded" errors were still not detected by anyone. Other methods or tools may be warranted for KB certification. However, over 85%, on average, of the rule base errors were judged detectable by the syntax and knowledge checking tools. Even a little increase in the present one-hour tool training could plausibly be expected to increase error detection significantly.
- 7) The method selection technique developed in Activity 1 was quite applicable in selecting Activity 6 methods to test.
- 8) The experimental methodology used in this experiment was highly effective and could be used for other behavioral studies of V&V performance.

7 REFERENCES

- Basili, V., and D. Weiss, *Evaluation of a Software Requirements Document by Analysis of Change Data*, Presented at the Fifth International Conference on Software Engineering, Computer Society Press of IEEE, Washington, D.C., IEEE Service Center, Piscataway, New Jersey, 08854, pp. 314-312.
- 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.
- Bernard, J.A., and T. Washio, *Expert System Applications Within the Nuclear Industry*, American Nuclear Society, La Grange Park, Illinois, 1989.
- Boehm, B.W., *Software Engineering*, IEEE Transactions on Computers, Vol. 25, No. 12, pp. 1126-1241, 1976.
- Groundwater, E.H., L.A. Miller, and S.M. Mirsky, *Guidelines for Verification and Validation of Expert Systems, Task 2: Survey and Document Expert System V&V Methodologies*. Draft, Science Applications International Corporation, 1710 Goodridge Drive, McLean, Virginia 22102, September 6, 1991.
- Hayes, J.H. Risk Reduction Through Requirements Tracing, *Proceedings of the Software Quality Conference*, May 15-18, Science Applications International Corporation, 1710 Goodridge Drive, McLean, Virginia 22102, 1990.
- Hayes, J.E., E.H. Groundwater, L.A. Miller, S. Kreyer, and S.M. Mirsky, *Guidelines for Verification and Validation for Expert Systems, Task 3, Test Applicability and Limitations of Conventional Verification Methods*, Technical Report, Science Applications International Corporation, 1710 Goodridge Drive, McLean, Virginia 22102, December 1992.
- Humphrey, W.S., *Managing the Software Process*, Addison Wesley Publishing Company, Reading, Massachusetts, 1990.
- Lewis, R.O., *Software Verification and Validation*, In Software Quality Management, Petrocelli Books, New York, New York, pp. 240, 1987.
- McGarry, F., *The Software Life Cycle -- Models and Management of Software Development*, Presentation to the U.S. Nuclear Regulatory Commission's Advisory Committee on Reactor Safeguards, Subcommittee meeting on Software Development and Design Concepts, Bethesda, Maryland, August 20-21, 1992.
- Miller, L.A., *Programming By Non-programmers*, International Journal of Man Machine Studies, 6, pp. 237-260, Science Applications International Corporation, 1710 Goodridge Drive, McLean, Virginia 22102, 1974.
- NUREG/CR-6018, EPRI TR-102106, Miller, L.A., E. Groundwater, and S.M. Mirsky, *Survey and Assessment of Conventional Software Verification and Validation Methods*, April 1993.
- Preece, A.D., *Towards a Methodology for Evaluating Expert Systems*, Expert Systems, Vol. 7, No. 4, pp. 215-223, November 1990.

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.

Winer, B.J., *Statistical Principles in Experimental Design*, Second Edition, McGraw-Hill, New York, New York, 1971.

APPENDIX A: SEEDED ERRORS

APPENDIX A
Table A1. System A - Syntax Checking Errors

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
C_9_1.3.1	IF (ADS HAS BEEN INITIATED) (SRV NUMBER_OPEN > 0) (SRV NUMBER_OPEN < (min SRVs OPEN CONSTANT)) (RPV PRESSURE >= PC PRESSURE + (min DEPRESSURIZATION CONSTANT)) THEN (SET 9 111.C.1) ;/*AUGMENT WITH OTHER SYSTEM (SET MORE DEPRESSURIZATION IS REQUIRED) (SET CAUTION 1) ;/*HIGH COOLDOWN (SET CAUTION 6) ;/*DEFEAT INTERLOCKS (SET CAUTION 2)	IF (ADS HAS BEEN INITIATED) (SRV NUMBER_OPEN > 0) (SRV NUMBER_OPEN < (min SRVs OPEN CONSTANT)) (RPV PRESSURE >= PC PRESSURE + (min DEPRESSURIZATION CONSTANT)) THEN (SET 9 111.C.1) ;/*AUGMENT WITH OTHER SYSTEM (SET MORE DEPRESSURIZATION IS REQUIRED) (SET CAUTION 1) ;/*HIGH COOLDOWN (SET CAUTION 6) ;/*DEFEAT INTERLOCKS (SET CAUTION 2)	D7 / TA-	H
C_7_5	IF (RPV WATER_LEVEL IS INCREASING) (RPV PRESSURE IS BELOW THE max LPCS INJECTION PRESSURE) ;/*20.35 (RPV PRESSURE IS ABOVE RCIC ISOLATION PRESSURE) ;/*5.74 kg/cm ² (RPV PRESSURE IS INCREASING) (RCIC is not RUNNING) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS REQUIRED)	IF (RPV WATER_LEVEL IS INCREASING) (RPV PRESSURE IS BELOW THE max LPCS INJECTION PRESSURE) ;/*20.35 (RPV PRESSURE IS ABOVE RCIC ISOLATION PRESSURE) ;/*5.74 kg/cm ² (RPV PRESSURE IS INCREASING) (RCIC is not RUNNING) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS REQUIRED)	D7 / IC-	H

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; *~~italics/strike thru~~* = DELETE(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A1. System A - Syntax Checking Errors
(continued - Page 2)

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
C_12_6	IF (HPCS IS RUNNING) (HPCS minimum FLOW VALVE IS SHUT) (HPCS TEST RETURN VALVE IS SHUT) (HPCS VALVES ARE not LINED UP TO CST) (RPV PRESSURE < HPCS RATED FLOW PRESSURE CONSTANT) (HPCS DISCHARGE PRESSURE > (RPV PRESSURE + HPCS min INJECTION PRESSURE CONSTANT) (HPCS FLOW > HPCS minimum FLOW CONSTANT) (FEEDWATER CHECK VALVES ARE SHUT) (RPV WATER_LEVEL IS ABOVE TAF) (RPV WATER_LEVEL IS not DECREASING) (WAIT(DELAY_TO_ESTABLISH_CORE_SPRAY CONSTANT)) THEN (SET EXIT C-12)	IF (HPCS IS RUNNING) (HPCS minimum FLOW VALVE IS SHUT) (HPCS TEST RETURN VALVE IS SHUT) (HPCS VALVES ARE not LINED UP TO CST) (RPV PRESSURE < HPCS RATED FLOW PRESSURE CONSTANT) (HPCS DISCHARGE PRESSURE > (RPV PRESSURE + HPCS min INJECTION PRESSURE CONSTANT) (HPCS FLOW > HPCS minimum FLOW CONSTANT) (FEEDWATER CHECK VALVES ARE SHUT) (RPV WATER_LEVEL IS ABOVE <u>NPSH</u> <u>LIMIT</u>) (RPV WATER_LEVEL IS not DECREASING) (WAIT(DELAY_TO_ESTABLISH_CORE_SPRAY CONSTANT)) THEN (SET EXIT C-12)	D5.1 / ISR	M
RPV_ Control_ 13.a	IF (ALTERNATE SHUTDOWN COOLING IS REQUIRED) THEN (SET C-13 ENTRY);/*ALT SHTDN CLG (ENTER C_13)	IF (ALTERNATE SHUTDOWN COOLING IS REQUIRED) THEN (SET C-13 ENTRY);/*ALT SHTDN CLG (SET REACTOR IS SHUTDOWN) (ENTER C_13)	D4 / TA+	H

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; *italics/strike-thru* = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A1. System A - Syntax Checking Errors
(continued - Page 3)

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
Composed_Points_2.c	IF (REACTOR MODE SWITCH IS IN SHUTDOWN POSITION) (SBLC SYSTEM IS not INJECTING) (NOT REACTOR IS SHUTDOWN) (WAIT(DELAY_TO_ALLOW_ROD_INSERT CONSTANT)) (ASK_USER(IS REACTOR SHUTDOWN?)) THEN (SET REACTOR IS SHUTDOWN)	IF (REACTOR MODE SWITCH IS IN SHUTDOWN POSITION) (SBLC SYSTEM IS not INJECTING) (NOT (REACTOR IS SHUTDOWN)) (WAIT(DELAY_TO_ALLOW_ROD_INSERT CONSTANT)) (ASK_USER(IS REACTOR SHUTDOWN?)) THEN (SET REACTOR IS SHUTDOWN)	D3 / IN	E
C_12_1.2	IF (NOT(RPV EMERGENCY DEPRESSURIZATION OCCURRED)) (S/POOL WATER LEVEL IS BELOW THE min SRV DISCHARGE LEVEL) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS REQUIRED)	IF (NOT(RPV EMERGENCY DEPRESSURIZATION OCCURRED)) (S/POOL WATER LEVEL IS BELOW THE min SRV DISCHARGE LEVEL) (PC PRESSURE IS ABOVE PRESS SUPPRESSION PRESSURE) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS REQUIRED)	D6 / IC+	H
RPV_Control_10.b	IF (IN C-10) ;/* A SAVE VARIABLE (NOT(EXIT C-10)) ;/* A SAVE VARIABLE THEN (SET C-10 ENTRY) (ENTER C_10)	IF (IN C-10) ;/* A SAVE VARIABLE (NOT(EXIT C-10)) ;/* A SAVE VARIABLE THEN (SET QUALITY(RPV WATER_LEVEL,BAD)) (SET C-10 ENTRY) (ENTER C_10)	D6 / TA+	H

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; ~~italics/strike thru~~ = DELETE

(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A1. System A - Syntax Checking Errors
(continued - Page 4)

Rule umber	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
RC/P_2.2	IF (NOT(REACTOR IS SHUTDOWN)) (RPV PRESSURE < OPERATING PRESSURE CONSTANT) ;/* 72.6 THEN (SET 3 111.C.2b) ;/* control below 72.6 with TBPV or TG	IF (NOT(REACTOR IS SHUTDOWN)) (RPV PRESSURE <u>is LESS THAN</u> OPERATING PRESSURE CONSTANT) ;/* 72.6 THEN (SET 3 111.C.2b) ;/* control below 72.6 with TBPV or TG	D5.5 / ISC	E
Composed_ Points_11	IF (FURTHER COOLDOWN IS REQUIRED) (RPV PRESSURE IS BELOW SHUTDOWN COOLING INTERLOCK PRESSURE) (RPV WATER_LEVEL IS ABOVE LOW LEVEL SCRAM SetPOINT) (RHR SHUTDOWN COOLING IS not RUNNING) (RPV TEMPERATURE > COLD_SHUTDOWN_TEMPERATURE CONSTANT) (ALL RODS ARE INSERTED AT OR BEYOND 06) (WAIT(DELAY_TO_ACHIEVE_COLD_SHU TDOWN CONSTANT)) THEN (SET ALTERNATE SHUTDOWN COOLING IS REQUIRED)	IF (FURTHER COOLDOWN IS REQUIRED) (RPV PRESSURE IS BELOW SHUTDOWN COOLING INTERLOCK PRESSURE) (RPV WATER_LEVEL IS ABOVE LOW LEVEL SCRAM SetPOINT) (RHR SHUTDOWN COOLING IS not RUNNING) (RPV TEMPERATURE > COLD_SHUTDOWN_TEMPERATURE CONSTANT) (ALL RODS ARE INSERTED AT OR BEYOND <u>09</u>) (WAIT(DELAY_TO_ACHIEVE_COLD_SHUTDO WN CONSTANT)) THEN (SET ALTERNATE SHUTDOWN COOLING IS REQUIRED)	D5.4 / ISR	M

LEGEND: bold = INSERT NEW; **bold & underline** = WORD CHANGE; *~~italics/strike-thru~~* = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A1. System A - Syntax Checking Errors
(continued - Page 5)

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
RC/P_5	IF (REACTOR IS SHUTDOWN) (RPV TEMPERATURE > COLD_SHUTDOWN_ TEMPERATURE CONSTANT) ;/* CLG REQ'D (RHR SHUTDOWN COOLING IS not RUNNING) (RPV WATER_LEVEL IS ABOVE SHUTDOWN COOLING INTERLOCK LEVEL) (RPV PRESSURE IS BELOW SHUTDOWN COOLING INTERLOCK PRESSURE) (WAIT(DELAY_TO_INIT_SHUTDOWN_COOLING CONSTANT)) (ASK_USER(IS FURTHER COOLDOWN REQUIRED?)) THEN (SET FURTHER COOLDOWN IS REQUIRED) (SET 3 111.C.3.5) ;/*PROCEED TO COLD SD WITH OTHER SYSTEMS	IF (REACTOR IS SHUTDOWN) (RPV TEMPERATURE ≤ COLD_SHUTDOWN_ TEMPERATURE CONSTANT) ;/* CLG REQ'D (RHR SHUTDOWN COOLING IS not RUNNING) (RPV WATER_LEVEL IS ABOVE SHUTDOWN COOLING INTERLOCK LEVEL) (RPV PRESSURE IS BELOW SHUTDOWN COOLING INTERLOCK PRESSURE) (WAIT(DELAY_TO_INIT_SHUTDOWN_COOLING CONSTANT)) (ASK_USER(IS FURTHER COOLDOWN REQUIRED?)) THEN (SET FURTHER COOLDOWN IS REQUIRED) (SET 3 111.C.3.5) ;/*PROCEED TO COLD SD WITH OTHER SYSTEMS	D5.3 / ISC	M
C_8_B_2. 3	IF (SRV NUMBER_OPEN ≥ min FLOOD SRVs OPEN CONSTANT) (RPV PRESSURE IS BELOW min ALT. RPV FLOODING PRESSURE) (RPV PRESSURE HAS BEEN BELOW min	IF (SRV NUMBER_OPEN ≥ min FLOOD SRVs OPEN CONSTANT) (RPV PRESSURE IS BELOW min D/W DEFLAG PRESSURE LIMIT) (RPV PRESSURE HAS BEEN BELOW min	D5.2 / ISR	M

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; *italics/strike-thru* = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A1. System A - Syntax Checking Errors
(continued - Page 5)

	ALTERNATE FLOOD PRESSURE) THEN (SET 8 111.C.1) (SET CAUTION 8) ;/*CORE DAMAGE	ALTERNATE FLOOD PRESSURE) THEN (SET 8 111.C.1) (SET CAUTION 8) ;/*CORE DAMAGE		
--	---	---	--	--

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; *~~italics/strike thru~~* = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A1. System A - Syntax Checking Errors
(continued - Page 6)

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
C_12_4	IF (HPCS IS RUNNING) (HPCS minimum FLOW VALVE IS SHUT) (HPCS TEST RETURN VALVE IS SHUT) (HPCS VALVES ARE not LINED UP TO CST) (RPV PRESSURE < HPCS RATED FLOW PRESSURE CONSTANT) (HPCS DISCHARGE PRESSURE > (RPV PRESSURE + HPCS min INJECTION PRESSURE)) (HPCS FLOW > HPCS minimum FLOW CONSTANT) (FEEDWATER CHECK VALVES ARE OPEN) THEN (SET 12 111.D);/*STOP ALL INJ	IF (HPCS IS RUNNING) (HPCS minimum FLOW VALVE IS SHUT) (HPCS TEST RETURN VALVE IS SHUT) (HPCS VALVES ARE not LINED UP TO CST) (RPV PRESSURE < HPCS RATED FLOW PRESSURE CONSTANT) (HPCS DISCHARGE PRESSURE > (RPV PRESSURE + <u>TAF</u>)) (HPCS FLOW > HPCS minimum FLOW CONSTANT) (FEEDWATER CHECK VALVES ARE OPEN) THEN (SET 12 111.D);/*STOP ALL INJ	D5.1 / ISV	M
SPL_11	IF (S/POOL WATER LEVEL IS ABOVE S/POOL LOAD LIMIT) (WAIT(DELAY_TO_RESTORE_SPLVL_BELOW_LOAD LIMIT CONSTANT)) THEN (SET 4 111.E.3.a.3) (SET REACTOR SCRAM IS REQUIRED) (SET CAUTION 1) (SET CAUTION 2)	IF (<u>RPV</u> WATER LEVEL IS ABOVE S/POOL LOAD LIMIT) (WAIT(DELAY_TO_RESTORE_SPLVL_BELOW_LOAD LIMIT CONSTANT)) THEN (SET 4 111.E.3.a.3) (SET REACTOR SCRAM IS REQUIRED) (SET CAUTION 1) (SET CAUTION 2)	D5.2 / ISV	M
C_12_1	IF (C_12) THEN (SET IN C-12)	IF (C_12) THEN (SET IN <u>C-13</u>)	D3 / TAV	M

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; *italics/strike-thru* = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A1. System A - Syntax Checking Errors
(continued - Page 7)

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
Composed_Points_10. g	IF (RAD RELEASE RATE IS NEAR GENERAL EMERGENCY RAD LIMIT) (NOT(RPV EMERGENCY DEPRESSURIZATION IS REQUIRED)) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS ANTICIPATED)	IF (RPV EMERGENCY DEPRESSURIZATION IS ANTICIPATED) (RAD RELEASE RATE IS NEAR GENERAL EMERGENCY RAD LIMIT) (NOT(RPV EMERGENCY DEPRESSURIZATION IS REQUIRED)) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS ANTICIPATED)	D2 / IC+	H
Cautions_8	IF (CAUTION 8) THEN (SET CAUTION-8 CORE DAMAGE MAY RESULT)	IF (CAUTION 8) THEN (SET <u>CAUTION-9</u> CORE DAMAGE MAY RESULT)	D2 / TAV	M
RC/P_N1.1	IF (RPV EMERGENCY DEPRESSURIZATION IS ANTICIPATED) (NOT(BORON INJECTION IS REQUIRED)) (MSIVs ARE OPEN) (RPV PRESSURE IS ABOVE SHUTDOWN COOLING INTERLOCK PRESSURE) ;/*9.34 kg/cm ² (REACTOR IS SHUTDOWN WITHOUT BORON) THEN (SET 3 111.C-N1) (SET CAUTION 1) ;/*HIGH COOLDOWN	IF (RPV EMERGENCY DEPRESSURIZATION IS ANTICIPATED) (NOT(BORON <u>INJECTION</u> IS REQUIRED)) (MSIVs ARE OPEN) (RPV PRESSURE IS ABOVE SHUTDOWN COOLING INTERLOCK PRESSURE) ;/*9.34 kg/cm ² (REACTOR IS SHUTDOWN WITHOUT BORON) THEN (SET 3 111.C-N1) (SET CAUTION 1) ;/*HIGH COOLDOWN	D1 / ISV	E

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; ~~*italics/strike-thru*~~ = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A1. System A - Syntax Checking Errors
(continued - Page 8)

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
PC/HG_2.12.1	IF (PC H2 CONC IS NEAR PC DEFLAG PRESSURE LIMIT) (NOT(RPV EMERGENCY DEPRESSURIZATION IS REQUIRED)) THEN (SET RVP EMERGENCY DEPRESSURIZATION IS ANTICIPATED)	IF (PC H2 CONC IS <u>NEARLY</u> PC DEFLAG PRESSURE LIMIT) (NOT(RPV EMERGENCY DEPRESSURIZATION IS REQUIRED)) THEN (SET RVP EMERGENCY DEPRESSURIZATION IS ANTICIPATED)	D1 / ISC	E
PC/P_3	IF (PC PRESSURE IS NEAR OR ABOVE PRESS SUPPRESSION PRESSURE) (NOT(PRV EMERGENCY DEPRESSURIZATION IS REQUIRED)) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS ANTICIPATED)	IF (PC PRESSURE IS NEAR OR ABOVE PRESS SUPPRESSION PRESSURE) (NOT(PRV EMERGENCY DEPRESSURIZATION IS REQUIRED)) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS ANTICIPATED)	D1 / ISR	M
SP/L_8	IF (S/POOL WATER_LEVEL IS NEAR OR ABOVE max S/POOL LEVEL LCO) (S/POOL WATER_LEVEL IS BELOW S/POOL LOAD LIMIT) (SPMS IS not INITIATED) THEN (SET 4 111.E.3.a)	IF (S/POOL WATER_LEVEL IS NEAR OR ABOVE max S/POOL LEVEL LCO) (S/POOL WATER_LEVEL IS BELOW S/POOL LOAD LIMIT) (SPMS IS not INITIATED) THEN (<u>GET</u> 4 111.E.3.a)	D1 / TAK	E

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; ~~italics/strike-thru~~ = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A2. System A - Knowledge Checking Errors

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
DW/HG_ 1.3	IF (NOT(PC/P-8));/*NOT VENTING PC (DRYWELL H2 CONC IS ABOVE min DETECTABLE H2 CONC);/* 0.5 (NOT(H2 RECOMBINER max OPERATING LIMIT HAS BEEN EXCEEDED));/* A SAVE (NOT(H2 DEFLAG LIMIT HAS BEEN EXCEEDED));/* A SAVE (RAD RELEASE RATE IS BELOW LCO RAD RELEASE RATE) THEN (SET CAUTION 6) (SET 4 111.F.1.a);/* VENT AND PURGE	IF (NOT(PC/P-8));/*NOT VENTING PC (DRYWELL H2 CONC IS ABOVE min DETECTABLE H2 CONC);/* 0.5 (NOT(H2 RECOMBINER max OPERATING LIMIT HAS BEEN EXCEEDED));/* A SAVE (NOT(H2 DEFLAG LIMIT HAS BEEN EXCEEDED));/* A SAVE (RAD RELEASE RATE IS BELOW LCO RAD RELEASE RATE) THEN (SET CAUTION 6) (SET 4 111.F.1.a);/* VENT AND PURGE	D7 / TA-	H
PC/HG_ 2.9	IF (NOT(PC/P-8));/*NOT VENTING PC (PC H2 CONC IS NEAR max RECOMBINER OPERATING LIMIT);/*6% (NOT(H2 RECOMBINER DEFLAG LIMIT HAS BEEN EXCEEDED));/*2mR/lr THEN (SET 4 111.F.4.a);/*VENT AND PURGE (SET CAUTION 6)	IF (NOT(PC/P-8));/*NOT VENTING PC (PC H2 CONC IS NEAR max RECOMBINER OPERATING LIMIT);/*6% (NOT(H2 RECOMBINER DEFLAG LIMIT HAS BEEN EXCEEDED));/*2mR/lr THEN (SET 4 111.F.4.a);/*VENT AND PURGE (SET CAUTION 6);/*DEFEAT ISOL INTERLOCKS	D7 / IC-	H

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; *~~italics/strike thru~~* = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A2. System A - Knowledge Checking Errors

RPV_ CONTROL_ 8b	/*DEFEAT ISOL INTERLOCKS IF (RPV FLOODING IS REQUIRED) (NOT(REACTOR IS SHUTDOWN WITHOUT BORON)) THEN (SET C-8 ENTRY) (ENTER C_8_B)	IF (PC H2 CONC IS NEAR PC DEFLAG TEMP LIMIT) (RPV FLOODING IS REQUIRED) (NOT(REACTOR IS SHUTDOWN WITHOUT BORON)) THEN (SET C-8 ENTRY) (ENTER C_8_B)	D6 / IC+	H
------------------------	--	--	----------	---

LEGEND: bold = INSERT NEW; bold & underline = WORD CHANGE; ~~xxxx~~ = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A2. System A - Knowledge Checking Errors
(continued - Page 2)

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
C_8_C_1_1.1	IF (SRV NUMBER_OPEN < min SRVs OPEN CONSTANT) (NOT(STOP ALL INJECTION)) THEN (SET 8 111.C.1)	IF (SRV NUMBER_OPEN < min SRVs OPEN CONSTANT) (NOT(STOP ALL INJECTION)) THEN (SET 8 111.C.1) (SET 13 111.B.3)	D6 / TA+	H
PC/HG_2.3	IF (NOT(PC-8)) ;/* NOT VENTING PC (PC H2 CONC IS ABOVE min DETECTABLE H2 CONC) ;/*0.5 (NOT(H2 RECOMBINER max OPERATING LIMIT HAS BEEN EXCEEDED)) ;/* A SAVE (NOT(H2 DEFLAG LIMIT HAS BEEN EXCEEDED)) ;/* A SAVE (RAD RELEASE RATE IS BELOW LCO RAD RELEASE RATE) THEN (SET 4 111.F.1.a) ;/* VENT AND PURGE (SET CAUTION 6) ;/* DEFEAT ISOL INTERLOCKS	IF (NOT(PC-8)) ;/* NOT VENTING PC (PC H2 CONC IS ABOVE min DETECTABLE H2 CONC) ;/*0.5 (NOT(H2 RECOMBINER max OPERATING LIMIT HAS BEEN EXCEEDED)) ;/* A SAVE (NOT(H2 DEFLAG LIMIT HAS BEEN EXCEEDED)) ;/* A SAVE (RAD RELEASE RATE IS BENEATH LCO RAD RELEASE RATE) THEN (SET 4 111.F.1.a) ;/* VENT AND PURGE (SET CAUTION 6) ;/*DEFEAT ISOL INTERLOCKS	D5.5 / ISC	E
COMPOSED_POINTS_7	IF (MAIN STEAM LINE LEAK DETECTION HAS TRIPPED) THEN (SET STEAM LINE BREAK IS INDICATED) ;/* A SAVE VARIABLE	IF (MAIN STEAM LINE LEAK DETECTION HAS TRIPPED) THEN (SET STEAM LINE BREAK = INDICATED) ;/* A SAVE VARIABLE	D5.5 / TAK	E

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; *~~italics/strike-thru~~* = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A2. System A - Knowledge Checking Errors
(continued - Page 2)

COMPOSED _POINTS_ 14.e	IF (RPV FLOODING IS ANTICIPATED) THEN (SET REACTOR SCRAM IS REQUIRED)	IF (RPV FLOODING IS ANTICIPATED) THEN (SET PC TEMPERATURE IS NEAR OR ABOVE PC SAT. TEMP LIMIT) (SET REACTOR SCRAM IS REQUIRED)	D4 / TA+	H
------------------------------	---	--	----------	---

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; *~~italics/strike-thru~~* = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A2. System A - Knowledge Checking Errors
(continued - Page 3)

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
GENERAL_RULES_9	IF (S/POOL WATER_LEVEL IS NEAR THE min SRV DISCHARGE LEVEL) (SRV NUMBER_OPEN > 0) THEN (SET SRVS OFF) ;/* DONT USE SRVS FOR PRESSURE CONTROL	IF (S/POOL WATER_LEVEL IS NEAR THE min SRV DISCHARGE LEVEL) (SRV NUMBER_OPEN = 0) THEN (SET SRVS OFF) ;/* DONT USE SRVS FOR PRESSURE CONTROL	D5.3 / ISC	M
DW/T_4	IF (DRYWELL TEMPERATURE IS ABOVE THE D/W DESIGN TEMPERATURE) (WAIT(DELAY_TO_REDUCE_DW_TEMP CONSTANT)) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS REQUIRED)	IF (<u>S/POOL</u> TEMPERATURE IS ABOVE THE D/W DESIGN TEMPERATURE) (WAIT(DELAY_TO_REDUCE_DW_TEMP CONSTANT)) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS REQUIRED)	D5.2 / ISV	M
CAUTIONS_H.a	IF (SGTS IS RUNNING) (ANY PC AREA TEMP NEAR OR ABOVE NON-COND EVAC TEMP) THEN (SET CAUTION-H SGTS SUCTION)	IF (SGTS IS RUNNING) (<u>DRYWELL TEMPERATURE</u> NEAR OR ABOVE NON-COND EVAC TEMP) THEN (SET CAUTION-H SGTS SUCTION)	D5.2 / ISR	M
PC/HG_2.5	IF (PC H2 CONC IS ABOVE min RECOMBINER OPERATING LIMIT) ;/* 1% (PC H2 CONC IS BELOW max RECOMBINER	IF (PC <u>PRESSURE</u> IS ABOVE min RECOMBINER OPERATING LIMIT) ;/* 1% (PC H2 CONC IS BELOW max RECOMBINER	D5.1 / ISV	M

LEGEND: bold = INSERT NEW; bold & underline = WORD CHANGE; ~~italics/strike-thru~~ = DELETE

(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A2. System A - Knowledge Checking Errors
(continued - Page 3)

	OPERATING LIMIT) ;/* 6% (PC H2 CONC IS BELOW PC DEFLAG PRESSURE LIMIT) (PC H2 CONC IS BELOW PC DEFLAG TEMP LIMIT) (H2 RECOMBINER IS not OPERATING) THEN (SET 4 111.F.2.a) ;/* OPER RECOMB	OPERATING LIMIT) ;/* 6% (PC H2 CONC IS BELOW PC DEFLAG PRESSURE LIMIT) (PC H2 CONC IS BELOW PC DEFLAG TEMP LIMIT) (H2 RECOMBINER IS not OPERATING) THEN (SET 4 111.F.2.a) ;/* OPER RECOMB		
--	--	--	--	--

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; ~~*italics/strike thru*~~ = DELETE
 (1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A2. System A - Knowledge Checking Errors
(continued - Page 4)

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
RPV_Entry_2	IF (RPV HIGH PRESSURE SCRAM SETPOINT) ;/*74.76 kg/cm ² THEN (SET RPV ENTRY)	IF (RPV <u>PRESSURE IS ABOVE CST</u>) ;/*74.76 kg/cm ² THEN (SET RPV ENTRY)	D5.1 / ISR	M
RPV_Control_8.d	IF (IN C-8-C) ;/* A SAVE VARIABLE - FLOOD OCCURRED (NOT(EXIT C-8-C)) ;/*A SAVE VARIABLE THEN (SET C-8 ENTRY) (ENTER C_8_C)	IF (IN C-8-C) ;/* A SAVE VARIABLE - FLOOD OCCURRED (NOT(EXIT C-8-C)) ;/*A SAVE VARIABLE THEN (SET C-8 ENTRY) (<u>SET IN C-8-C</u>) (ENTER C_8_C)	D4 / TA+	M
SP/T_9	IF (S/POOL TEMPERATURE IS NEAR OR ABOVE S/POOL HCTL) (RPV PRESSURE IS NEAR OR ABOVE THE S/POOL HCTL) (NOT(RPV EMERGENCY DEPRESSURIZATION IS REQUIRED)) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS ANTICIPATED)	IF (S/POOL TEMPERATURE IS NEAR OR ABOVE S/POOL HCTL) (RPV PRESSURE IS NEAR OR ABOVE THE S/POOL HCTL) (NOT(RPV EMERGENCY DEPRESSURIZATION IS REQUIRED)) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS ANTICIPATED)	D3 / IN	M
C_13_2.2	IF (RHR IS not RUNNING IN S/POOL COOLING MODE) (WAIT(DELAY TO INIT S/POOL C	IF (RHR IS not RUNNING IN S/POOL COOLING MODE) (WAIT(DELAY TO INIT S/POOL COOLING	D3 / TAV	M

LEGEND: bold = INSERT NEW; **bold & underline** = WORD CHANGE; ~~italics/strike thru~~ = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A2. System A - Knowledge Checking Errors
(continued - Page 4)

	OOLING CONSTANT)) (ASK_USER(IS ALTERNATE S/POOL COOLING IN SERVICE?)) THEN (SET S/POOL COOLING IS IN PROGRESS)	CONSTANT)) (ASK_USER(IS ALTERNATE S/POOL COOLING IN SERVICE?)) THEN (SET S/POOL COOLING IS not IN PROGRESS)		
--	---	--	--	--

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; ~~*italics/strike thru*~~ = DELETE
 (1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A2. System A - Knowledge Checking Errors
(continued - Page 5)

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
RC/L_1.9	IF (RPV WATER_LEVEL CANNOT BE DETERMINED TO BE ABOVE LEVEL 1) (LPCI-B IS not RUNNING) (RHR-B IS not RUNNING IN S/POOL COOLING MODE) (RHR-B IS not RUNNING IN CONTAINMENT SPARY MODE) (RHR-B PUMP IS RUNNING) (ASK_USER(RHR-B IS RUNNING?)) THEN (SET 3 111.B.1.b.4.a) ;/* RETURN RHR-B TO LPCI MODE)	IF (RPV WATER_LEVEL CANNOT BE DETERMINED TO BE ABOVE LEVEL 1) (RHR-B PUMP IS not RUNNING) (LPCI-B IS not RUNNING) (RHR-B IS not RUNNING IN S/POOL COOLING MODE) (RHR-B IS not RUNNING IN CONTAINMENT SPARY MODE) (RHR-B PUMP IS RUNNING) (ASK_USER(RHR-B IS RUNNING?)) THEN (SET 3 111.B.1.b.4.a) ;/* RETURN RHR-B TO LPCI MODE)	D2 / IC+	H
Cautions_3	IF (CAUTION 3) THEN (SET CAUTION-3 BYPASS MSIV INTLK)	IF (CAUTION 3) THEN (SET CAUTION- <u>4</u> BYPASS MSIV INTLK)	D2 / TAV	M
PC_Entry_8.a	IF (SRV NUMBER_OPEN > 0) (WAIT(DELAY_TO_INIT_RX_SCRAM_ON_STUCKOPEN_SRV CONSTANT)) (ANY SRV IS STUCK OPEN) THEN (SET SRV STUCK OPEN PAST TIME LIMIT) (SET REACTOR SCRAM IS REQUIRED)	IF (SRV NUMBER_OPEN > 0) (WAIT(DELAY_TO_INIT_RX_SCRAM_ON_STUCKOPEN_SRV CONSTANT)) (ANY <u>SVR</u> IS STUCK OPEN) THEN (SET SRV STUCK OPEN PAST TIME LIMIT) (SET REACTOR SCRAM IS REQUIRED)	D1 / ISV	E

LEGEND: bold = INSERT NEW; **bold & underline** = WORD CHANGE; *~~italics/strike-thru~~* = DELETE

(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A2. System A - Knowledge Checking Errors
(continued - Page 6)

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
DW/HG_1. 7	IF (DRYWELL H2 CONC IS NEAR OR ABOVE FLAME PROPAGATION LIMIT) (PC PRESSURE IS BELOW PC PRESSURE LIMIT) (DRYWELL H2 CONC IS BELOW D/W DEFLAG PRESSURE LIMIT) (DRYWELL H2 CONC IS BELOW D/W DEFLAG TEMP LIMIT) (PC H2 CONC IS BELOW PC DEFLAG PRESSURE LIMIT) (PC H2 CONC IS BELOW PC DEFLAG TEMP LIMIT) (H2 MIXING SYSTEM IS NOT OPERATING) THEN (SET 4111.F.2.b) ;/*OPER H2 MIXING SYS	IF (DRYWELL H2 CONC IS NEAR OR ABOVE FLAME PROPAGATION LIMIT) (PC PRESSURE IS BELOW PC PRESSURE LIMIT) (DRYWELL H2 CONC IS BELOW D/W DEFLAG PRESSURE LIMIT) (DRYWELL H2 CONC IS BELOW D/W DEFLAG TEMP LIMIT) (PC H2 CONC IS BELOW PC DEFLAG PRESSURE LIMIT) (PC H2 CONC IS BELOW PC DEFLAG TEMP LIMIT) (H2 MIXING SYSTEM <u>ISN'T</u> OPERATING) THEN (SET 4111.F.2.b) ;/*OPER H2 MIXING SYS	D1 / ISC	E
DW/T_3	IF (DRYWELL TEMPERATURE IS NEAR OR ABOVE THE D/W DESIGN TEMP) (NOT(RPV EMERGENCY DEPRESSURIZATION IS REQUIRED)) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS ANTICIPATED)	IF (DRYWELL TEMPERATURE IS NEAR OR ABOVE THE D/W DESIGN TEMP) (NOT(RPV EMERGENCY DEPRESSURIZATION IS <u>REQUISITE</u>)) THEN (SET RPV EMERGENCY DEPRESSURIZATION IS ANTICIPATED)	D1 / TAK	E

LEGEND: bold = INSERT NEW; **bold & underline** = WORD CHANGE; *~~italics/strike thru~~* = DELETE

(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A2. System A - Knowledge Checking Errors
(continued - Page 6)

Composed_ Points_6	IF (MAIN STEAM LINE RADIATION MONITORS ARE TRIPPED) THEN (SET GROSS FUEL FAILURE IS INDICATED) ;/* A SAVE VARIABLE	IF (MAIN STEAM LINE RADIATION MONITORS ARE TRIPPED) THEN (SETT GROSS FUEL FAILURE IS INDICATED) ;/* A SAVE VARIABLE	D1 / TAK	E
-----------------------	---	--	----------	---

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; *~~italics/strike thru~~* = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A3. System B - Syntax Checking Errors

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
131	if [NI Power/Power Range] <= 5 then Power Range Trip has Value normal	if [NI Power/Power-Range] <= 5 then Power Range Trip has Value normal	D1 / ISV	E
175	if [SG-C Press] >= [min SG Code Safety Press] and rate of [SG-C Press] < 0 then SGC < Code Safety.....	if [SG-C Pressure] >= [min SG Code Safety Press] and rate of [SG-C Pressure] < 0 then SGC < Code Safety.....	D5.6 / ISV	M
166	if [SG-C Level] > [min SG Level: Forced Flow] then SGC > min has Value normal	if [SG-C Press] > [min SG Level: Forced Flow] then SGC > min has Value normal	D5.1 / ISV	H
014	if NEGATIVE rate of [SG-A Saturation Temp] <= 100	if NEGATIVE rate of [SG-A Saturation Temp] <= 100	D5.6 / ISV	H
111	if [Boron concen increased for each stuck rod?] is no then.....	if [Boron concen increased for each stuck rod?] = no then.....	D4.2 / ISC	M
133	if [NI Power/Source Range] >= [max Source Range NI Indication] or [NI Power/Source Range] <= [min Source range NI Indication] then.....	if [NI Power/Source Range] >= [max Source Range NI Indication] or [NI Power/Source Range] <= [min Source range NI Indication] then.....	D5.3 / ISR	M
127	if rate of [NI Power/Intermed Range] >= 0 then.....	if rate of [NI Power/Intermed Range] >= [max Source Range NI Indication] then.....	D4.2 / ISR	H
120	if [No. of Control Rods did not insert?] > 5 then.....	if [No. of Control Rods did not insert?] > <u>-5</u> then.....	D5.4 / ISR	E
023	if [SG-B Press] >= [min SG Code Safety Press] then.....	if [SG-B Press] >= [<u>min SG Level: Natural Circ</u>] then.....	D5.1 / ISR	H
122	if [No. of Control Rods did not insert?] <= 1 then.....	if [No. of Control Rods did not insert?] <= <u>1.5</u> then.....	D5.4 / ISR	M
262	if rate of [SG-A Level] > 0 and [SG-A Level] <= [min SG Level: Natural Circ]	if rate of [SG-A Level] > 0 and [SG-A Level] <= [min SG Level: Natural Circ]	D1 / TAK	E

LEGEND: bold = INSERT NEW; **bold & underline** = WORD CHANGE; ~~italics/strike thru~~ = DELETE

(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A3. System B - Syntax Checking Errors

	then SGA > min Nat Circ has Value improving	then SGA > min Nat Circ has Value slightly improving		
217	if [SG-C Level] >= [max SG Level] then SGC < max has value degraded	if [SG-C Level] >= [max SG Level] then SGC < max has value degraded d	D1 / TAK	E
188	if NEGATIVE rate of [SG-C Saturation Temp] > 100 then SGCS at Temp Cooldown has Value degraded	if NEGATIVE rate of [SG-C Saturation Temp] > 100 then SGCS at Temp Cooldown has Value <u>normal</u>	D3 / TAK	M

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; ~~*italics/strike-thru*~~ = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A4. System B - Knowledge Checking Errors

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
060	if rate of [Temp Hot Leg A] <= 0 then.....	if rate of [Temp <u>Hott</u> Leg A] <= 0 then.....	D1 / ISV	E
045	if [RCS Press] < [RCS Press Safety Limit] then.....	if [RCS <u>Pressure</u>] < [RCS Press Safety Limit] then.....	D2 / ISV	M
068	if Absolute Value of ([Core Exit Temp-Max] - [Temp Hot Leg A]) <= 10 then.....	if Absolute Value of ([Core Exit Temp-Max] - [Temp <u>Cold</u> Leg A]) <= 10 then.....	D4.2 / ISV	H
069x2	if Absolute Value of ([Core Exit Temp-Max] - [Temp Hot Leg A]) > 10 then.....	if Absolute Value of ([Core Exit Temp-Max] - [Temp Hot Leg A]) > 10 then.....	D5.6 / ISV	H
146	if rate of [Temp Hot Leg C] > 0 then.....	if rate of [Temp Hot Leg C] <u>exceeds</u> 0 then.....	D4.2 / ISC	M
086	if [RCS Press] < [max TSNDTT Press] and [RCS Press] > [max TSNDTT Press] - [NDTT Press Band] and...	if [RCS Press] < [max TSNDTT Press] and [RCS Press] > [max TSNDTT Press] - [NDTT Press <u>Bands</u>] and...	D2 / ISR	M
067	if rate of [Temp Cold Leg B] > 0 then.....	if rate of [Temp Cold Leg B] > <u>[Temp Cold Leg B]</u> <u>±</u> <u>10</u> then.....	D4.2 / ISR	H
76	if [Temp Hot Leg A] - [Temp Cold Leg A] > 10 then.....	if [Temp Hot Leg A] - [Temp Cold Leg A] > <u>-20</u> then.....	D5.4 / ISR	E
060x2	if rate of [Temp Hot Leg A] <= 0 then.....	if rate of [Temp Hot Leg A] <= <u>Absolute Value of</u> <u>([Core Exit Temp - Max] - [Temp Hot Leg</u> <u>Bl])</u> then.....	D5.1 / ISR	H
223	if [RCS Forced Flow?] is no then.....	if [RCS Forced Flow?] is <u>maybe</u> then.....	D5.4 / ISR	E
239x2	if [Temp Hot Leg A] - [Temp Cold Leg - A2] > 50 then ThA2 - TcA2 < 50 has Value degraded	if [Temp Hot Leg A] - [Temp Cold Leg - A2] > 50 then ThA2 - TcA2 < 50 has Value <u>somewhat</u> degraded	D1 / TAK	E

LEGEND: **bold** = INSERT NEW; **bold & underline** = WORD CHANGE; ~~italics/strike thru~~ = DELETE

(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX A
Table A4. System B - Knowledge Checking Errors
(continued - Page 2)

Rule Number	Original Rule	Changed Rule	Detectability Level/Error Location (1)	Difficulty Category (2)
070x2	if Absolute Value of ([Core Exit Temp - max] - [Temp Hot Leg B]) <= 10 then Delta Tcore Thot Loop B has Value normal	if Absolute Value of ([Core Exit Temp - max] - [Temp Hot Leg B]) <= 10 then Delta Tcore Thot Loop B has Value <u>normaal</u>	D1 / TAK	E
139	if [Temp Hot Leg C] - [Temp Cold Leg C] <= 50 then Thc - Tcc < 50 has Value normal	if [Temp Hot Leg C] - [Temp Cold Leg C] <= 50 then Thc - Tcc < 50 has Value <u>dgrading</u>	D3 / TAK	M

111

LEGEND: bold = INSERT NEW; **bold & underline** = WORD CHANGE; *~~italics/strike-thru~~* = DELETE
(1) Defined in Figure 3.5-1 and Table 3.5-1; (2) Defined in Figure 3.5-2

APPENDIX B: EXPERIMENTAL MEANS (AVERAGES)

APPENDIX B
Table B1. Method Means
(if Method is significant)

Number	Dependent Variable	Method 1 (Grouping)	Method 2 (Tracing)	Method 3 (Syntax)	Method 4 (Knowledge)
1	Total Number Responses	34.80	11.90	24.30	23.10
2	Number Correctly Found Problems	21.65	7.30	8.95	10.95
3	Percent Correctly Found Problems	62.07	47.93	58.61	69.13
4	Total Time (seconds)	1848.00	4656.00	4509.00	3981.00
5	Number Problems Missed	11.85	11.00	8.15	5.80
6	Total Number of False Alarms	12.95	4.10	15.35	12.15
7	Number True False Alarms	9.50	3.80	14.95	12.15
8	Number of Borderline False Alarms	3.45	0.30	0.40	0.00
10	Number True False Alarms - High Confidence	5.75	3.20	14.30	12.15
11	Number Borderline False Alarms - Low Confidence	0.05	0.00	0.40	0.00
12	Number Borderline False Alarms - High Confidence	3.08	0.30	0.00	0.00
13	Average Confidence for Correct Problems	4.64	4.40	4.31	4.74
15	Average Confidence for all Responses	4.63	4.28	3.86	4.30
16	Average Time for all Responses (seconds)	66.94	442.90	211.60	187.01
17	Average Time per Correct Response (seconds)	275.57	799.67	606.71	439.15
18	Average Time per False Alarm (seconds)	248.91	1800.10	459.59	794.65
19	Number Hard Problems Found	N/A	0.25	2.50	2.60
20	Number Medium Problems Found	N/A	4.65	2.60	3.95
21	Number Easy Problems Found	N/A	2.35	3.40	4.40
23	Average Time per Medium Problem (seconds)	N/A	1252.00	2029.70	1177.70
24	Average Time per Easy Problem (seconds)	N/A	2669.20	1988.40	1268.40

APPENDIX B
Table B2. Method and Condition Means
(if significant)

Number	Dependent Variable	Method 1		Method 2		Method 3		Method 4	
		Condition 1	Condition 2	Condition 1	Condition 2	Condition 1	Condition 2	Condition 1	Condition 2
2	Number Correctly Found Problems	32.00	11.30	7.80	6.80	10.30	7.60	13.40	8.50
3	Percent Correctly Found Problems	93.47	30.66	50.47	45.39	66.30	50.92	82.85	55.42
5	Number Problems Missed	1.50	22.20	10.70	11.30	6.70	9.60	3.10	8.50
6	Total Number False Alarms	5.30	20.60	4.90	3.30	14.10	16.60	6.00	18.30
7	Number True False Alarms	0.50	18.50	4.40	3.20	--	--	--	--
8	Number Borderline False Alarms	--	--	--	--	0.20	0.60	--	--
10	Number True False Alarms - High Confidence	0.00	11.50	--	--	--	--	--	--
19	Number Hard Problems Found	N/A	N/A	0.20	0.30	3.20	1.80	2.90	2.30
21	Number Easy Problems Found	N/A	N/A	--	--	4.20	2.60	5.40	3.40
22	Average Time per Hard Problem (seconds)	N/A	N/A	--	--	1345.50	3505.00	1847.00	2331.60
24	Average Time per Easy Problem (seconds)	N/A	N/A	--	--	885.30	3091.50	712.80	1824.10

APPENDIX B
Table B3. System and Method Means
(if significant)

Number	Dependent Variable	System 1				System 2			
		Method 1	Method 2	Method 3	Method 4	Method 1	Method 2	Method 3	Method 4
1	Total Number of Responses	37.20	11.90	22.00	30.70	32.40	11.90	26.60	15.50
2	Number Correctly Found Problems	30.90	8.80	7.60	10.30	12.40	5.80	10.30	11.60
3	Percent Correctly Found Problems	70.22	31.42	38.00	49.04	53.91	64.44	79.23	89.23
4	Total Time (seconds)	1920.00	4536.00	5286.00	5280.00	1776.00	4776.00	3732.00	2682.0
5	Number Problems Missed	13.10	19.20	13.10	10.20	10.60	2.80	3.20	1.40
6	Total Number False Alarms	6.30	2.70	14.40	20.40	19.60	5.50	16.30	3.90
7	Number True False Alarms	2.30	2.10	13.60	20.40	16.70	5.50	16.30	3.90
10	Number True False Alarms - High	1.50	1.60	12.50	20.40	10.00	4.80	16.10	3.90
11	Number Borderline False Alarms - Low	0.10	0.00	0.80	0.00	0.00	0.00	0.00	0.00
17	Average Time per Correct Problem	91.66	668.31	840.45	616.82	459.47	931.02	372.97	261.48
18	Average Time per False Alarm (seconds)	341.48	2488.50	401.93	450.00	156.34	1111.70	517.25	1139.2
19	Number Hard Problems Found	N/A	0.50	2.00	1.60	N/A	0.00	3.00	3.60
20	Number Medium Problems Found	N/A	6.40	2.10	4.60	N/A	2.90	3.10	3.30

APPENDIX B
Table B4. Method, System, and Condition Means
(if significant)

Number	Dependent Variable	Method 1				Method 2				Method 3				Method 4			
		System 1		System 2		System 1		System 2		System 1		System 2		System 1		System 2	
		Cond 1	Cond 2	Cond 1	Cond 2	Cond 1	Cond 2	Cond 1	Cond 2	Cond 1	Cond 2	Cond 1	Cond 2	Cond 1	Cond 2	Cond 1	Cond 2
1	Total Number Responses	50.60	23.80	24.80	40.00	13.00	10.80	13.20	10.60	18.60	25.40	30.20	23.00	21.80	39.60	17.00	14.00
4	Total Time (seconds)	1512.00	2328.00	1212.00	2340.00	4596.00	4476.00	4392.00	5160.00	3600.00	6972.00	3840.00	3624.00	4944.00	5616.00	2928.00	2436.00
6	Total Number False Alarms	6.60	6.00	4.00	35.20	2.80	2.60	7.00	4.00	9.00	19.80	19.20	13.40	8.00	32.80	4.00	3.80
7	Number True False Alarms	1.00	3.60	0.00	33.40	18.0	2.40	7.00	4.00	8.60	18.60	19.20	13.40	8.00	32.80	4.00	3.80
10	Number True False Alarms - High Confidence	0.00	3.00	0.00	20.00	1.20	2.00	6.00	3.60	7.00	18.00	18.80	13.40	8.00	32.80	4.00	3.80
17	Average Time per Correct Response (seconds)	34.36	148.97	59.41	859.54	559.70	776.90	798.65	1063.00	365.30	1315.60	358.60	387.30	358.05	875.60	225.20	297.70

APPENDIX B
Table B5. System Means
(if significant)

Number	Dependent Variable	System 1	System 2
2	Number Correctly Found Problems	14.40	10.02
3	Percent Correctly Found Problems	47.17	71.70
4	Total Time (seconds)	4255.50	3241.50
5	Number Problems Missed	13.90	4.50
11	Number Borderline False Alarms- Low Confidence	0.23	0.00
13	Average Confidence for Correct Problems	4.34	4.70
19	Number Hard Problems Found	1.36	2.20
20	Number Medium Problems Found	4.36	3.10
21	Number Easy Problems Found	2.96	3.80
22	Average Time Per Hard Problem (seconds)	3583.60	1032.20
23	Average Time per Medium Problem (seconds)	1773.65	1186.40
24	Average Time per Easy Problem (seconds)	2639.10	1311.60

APPENDIX B
Table B6. Condition Means
(if significant)

Number	Dependent Variable	Condition 1	Condition 2
2	Number Correctly Found Problems	15.87	8.55
3	Percent Correctly Found Problems	73.20	45.60
4	Total Time (seconds)	3378.00	4119.00
5	Number Problems Missed	5.50	12.90
6	Total Number of False Alarms	7.57	14.70
7	Number True False Alarms	6.20	14.00
8	Number of Borderline False Alarms	1.37	0.70
10	Number True False Alarms - High Confidence	5.62	12.07
16	Average Time for all Responses (seconds)	196.60	257.60
17	Average Time per Correct Response (seconds)	344.90	715.60
19	Number Hard Problems Found	2.10	1.46
20	Number Medium Problems Found	4.36	3.10
21	Number Easy Problems Found	3.96	2.80
23	Average Time per Medium Problems (seconds)	1222.50	1777.00
24	Average Time per Easy Problem (seconds)	1444.30	2506.30

APPENDIX B
Table B7. Means by System and Condition
(if significant)

Number	Dependent Variable	System 1 Condition 1	System 1 Condition 2	System 2 Condition 1	System 2 Condition 2
2	Number Correctly Found Problems	19.25	9.55	12.50	7.55
5	Number Problems Missed	8.85	18.95	2.15	6.85
13	Average Confidence for Correct Problems	4.57	4.12	4.62	4.78
21	Number Easy Problems Found	4.13	1.80	3.80	3.80
22	Average Time per Hard Problem (seconds)	2536.66	4840.00	970.50	1094.00
24	Average Time per Easy Problem (seconds)	1371.53	3906.66	1517.20	1106.00

APPENDIX B
Table B8. Analysis of System by Method Interaction
(if significant)

Number	Dependent Variable	1st Method Significant				2nd Method Significant				3rd Method Significant				4th Method Significant			
		System 1		System 2		System 1		System 2		System 1		System 2		System 1		System 2	
		Cond 1	Cond 2	Cond 1	Cond 2	Cond 1	Cond 2	Cond 1	Cond 2	Cond 1	Cond 2	Cond 1	Cond 2	Cond 1	Cond 2	Cond 1	Cond 2
1	Total Number Responses	<u>M4</u> 21.80	39.60	17.00	14.00												
2	Number Correctly Found Problems	<u>M1</u> 44.00	17.80	20.00	4.80	<u>M3</u> 9.60	5.60	11.00	9.60	<u>M4</u> 13.80	6.80	13.00	10.20				
3	Percent Correctly Found Problems	<u>M1</u> 100.00	40.45	86.95	20.87	<u>M2</u> 34.28	28.57	66.66	<u>M3</u> 62.22	<u>M3</u> 48.00	23.00	81.54	70.76	<u>M4</u> 65.70	35.38	100.00	78.44
4	Total Time (seconds)	<u>M3</u> 3600.00	6972.00	3840.00	3624.00	<u>M4</u> 4944.00	5616.00	2928.00	2436.00								
5	Number Problems Misses	<u>M2</u> 18.40	20.00	3.00	2.60	<u>M3</u> 10.80	15.40	2.60	3.80	<u>M4</u> 6.20	14.20	0.00	2.80				
6	Total Number False Alarms	<u>M1</u> 6.60	6.00	4.00	35.20	<u>M2</u> 2.80	2.60	7.00	4.00	<u>M4</u> 8.00	32.80	4.00	3.80				
7	Number True False Alarms	<u>M1</u> 1.00	3.60	0.00	33.40	<u>M2</u> 1.80	2.40	7.00	4.00								
10	Number True False Alarms - High Confidence	<u>M2</u> 0.00	3.00	0.00	20.00	<u>M2</u> 1.20	2.00	6.00	3.60								
11	Number Borderline False Alarms - Low Confidence	<u>M3</u> 0.40	1.20	0.00	0.00												
17	Average Time per Correct Response (seconds)	<u>M3</u> 365.30	1315.60	358.60	387.30	<u>M4</u> 358.05	875.60	225.23	297.70								
18	Average Time per False Alarm (seconds)	<u>M2</u> 2271.00	2706.00	804.40	1419.00	<u>M4</u> 661.47	238.59	1028.57	1250.00								
19	Number Hard Problems Found	<u>M2</u> 0.40	0.60	0.00	0.00	<u>M4</u> 1.80	1.40	4.00	3.20								
20	Number Medium Problems Found	<u>M2</u> 7.00	5.80	3.60	2.20	<u>M4</u> 6.20	3.00	4.00	2.60								

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. 4

2. TITLE AND SUBTITLE

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

Evaluation of Knowledge Base Certification Methods

3. DATE REPORT PUBLISHED

MONTH YEAR

March 1995

4. FIN OR GRANT NUMBER

L1530

5. AUTHOR(S)

L.A. Miller, J.E. Hayes, 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 fourth 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. Here are presented the results of the Knowledge Base Certification activity that was concerned with developing and testing various static analysis methods for assuring the quality of knowledge bases. The testing procedure used was that of a behavioral experiment involving evaluation of four different V&V methods. The study used two real nuclear expert systems: a boiling water reactor emergency operating procedures tracking system, and a pressurized water reactor safety assessment system. The twenty participants were from three nuclear utilities, the USNRC Technical Training Center, the University of Maryland, EG&G Idaho, and SAIC. The major conclusion was that the use of tools in static knowledge base certification results in significant improvement in detecting all types of defects, avoiding false alarms, and completing the effort in less time. The simulated knowledge-checking tool, based on supplemental information, was the most effective of the tools.

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