NASA Contractor Report 181769

RELIABILITY AND PERFORMANCE EVALUATION
OF SYSTEMS CONTAINING EMBEDDED
RULE-BASED EXPERT SYSTEMS

Robert M. Beaton
Milton B. Adams
James V. A. Harrison

THE CHARLES STARK DRAPER LABORATORY, INC.
Cambridge, Massachusetts

NASA

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

TABLE OF CONTENTS

ABSTRACT

A method for evaluating the reliability of real-time systems containing embedded rule-based expert systems is proposed and investigated. It is a three stage technique that addresses the impact of knowledge base uncertainties on the performance of expert systems. In the first stage, a Markov reliability model of the system is developed which identifies the key performance parameters of the expert system. In the second stage, the evaluation method is used to determine the values of the expert system's key performance parameters. The performance parameters can be evaluated directly by using a probabilistic model of uncertainties in the knowledge base or by using sensitivity analyses. In the third and final stage, the performance parameters of the expert system are combined with performance parameters for other system components and subsystems to evaluate the reliability and performance of the complete system. The evaluation method is demonstrated in the context of a simple expert system used to supervise the performance of an FDI algorithm associated with an aircraft longitudinal flight-control system.

KEYWORDS: Expert Systems; System Evaluation; Markov Models;
            Uncertain Knowledge Base Systems; Sensitivity Analyses

1 INTRODUCTION

1.1 Background

Recent applications of expert systems technology to a large class of engineering problems ([1] - [9]) suggest that expert systems may be useful for solving complex aerospace planning and control problems. Indeed, major efforts are underway to develop expert systems for a wide variety of aerospace applications including DARPA's Pilot's Associate program [10], NASA's space station energy management system [11], and NASA's Systems Autonomy Technology program (SADP) [12].

Some of the aerospace applications for which expert systems are currently being considered are closely associated with real-time life-critical operations. Thus, it is reasonable to expect that eventually such expert systems will play real-time, life-critical roles in these applications. As with other systems that perform life critical tasks, these expert systems will have to be thoroughly evaluated before they can be flight tested. This evaluation process is necessary to insure that reliability and performance requirements are satisfied. However, a satisfactory and generally accepted methodology for evaluating the reliability and performance of expert systems does not currently exist.

Historically, because expert systems have not addressed life critical and/or real-time problems, there has not been a compelling need to address the evaluation problem. To date, most investigations of the evaluation problem have been either incomplete [13], strictly qualitative [14], [15], or problem specific [16], [17], [18] and [19]. Recently, NASA convened a workshop to address issues associated with verification and validation of knowledge based systems [20]. The lack of a general, complete and quantitative evaluation methodology is a major impediment to exploiting the potential of expert systems in aircraft and spacecraft systems. This report describes a method for performing sensitivity analyses on the input/output performance of an expert system in the presence of inexact input values and imperfect rules. This method represents a first step towards the development of a more general evaluation methodology.

1.2 Problem Definition.

Two types of evaluation capabilities are required to support the development of real-time, rule-based expert systems. First, an ability to evaluate the performance of the expert system itself, operating in a stand alone mode is required. The methodology to perform this evaluation must identify the key performance parameters of the expert system, as well as specify how these parameters are computed. Second, because, in general, real-time expert systems will be imbedded within larger systems, an ability to evaluate the effect of the expert system's performance on the overall system is required. The methodology that is developed for this purpose must integrate the expert system's performance parameters into existing methods for evaluating system performance and reliability.

1.3 Scope.

The term expert system is often applied loosely to a large class of problem solving software tools. An attempt to develop a new and general evaluation methodology for such a large class of systems is an ambitious goal. In order to reduce the scope of the problem,

we have restricted the evaluation methodology to **forward-chaining rule-based expert systems**. Rule-based expert systems are an important class of knowledge based systems that are used to address real problems. Therefore, an evaluation methodology that specifically focuses on such systems is not of limited interest.

The development of an expert system is a three stage process, any one of which can have an effect on the performance and reliability of the larger system into which the expert system is integrated. These stages are system development, software implementation and hardware integration. *System development* refers to the development of the knowledge representation for the assertions and rules used in the expert system and the design of the inference engine which operates on those rules and assertions. *Software implementation* is the process of coding the system defined in Stage 1. *Hardware integration* is the process of porting the coded system onto a specific machine.

This study addresses the evaluation of the product of the system development stage in terms of its reliability and performance. In particular, we focus on *how errors in the assertions and the rules affect the performance and reliability of the expert system*. In order to make the results of the evaluation useful for comparing alternative expert system designs and for evaluating the impact of the performance of the expert system on the overall integrated system, the effect of assertion and rule errors must be described by quantitative means.

This study does not address the software implementation or hardware integration stages. In this sense, our investigation is analogous to the evaluation of the performance of control or Fault Detection and Isolation (FDI) algorithms, where the objective is to investigate stability or decision performance before considering specific problems relating to software and hardware implementation.

1.4 Approach.

The expert system performance discussed in this paper is the ability of the system to perform its primary function. The words performance and reliability will be used interchangeably to refer to this ability. The approach for evaluating expert system performance consists of two stages. In the first stage, an overall system reliability model is developed that incorporates the expert system as a subsystem. In the approach described in this study, we use Markov models to characterize the reliability of the complete system. It will be assumed that the ability of the system to perform its primary function (what we shall refer to as performance), can be characterized in terms of the reliability information that is provided by the Markov model. We believe that Markov models represent the easiest and most powerful method for modeling the performance and reliability of complicated systems. However, the techniques described in this report could support other types of system modeling.

Markov models are used to model the effects of the expert system, as well as other subsystems, on the performance of the complete system. This allows the key performance parameters of all of the subsystems (including the expert system) to be identified. These performance parameters must be supplied for the Markov model evaluator to evaluate system performance. Methods for computing these performance parameters exist for many types of subsystems (e.g., control systems, navigation systems, power systems, etc.). They do not currently exist for expert systems.

In the second stage of the evaluation, the parameters describing the performance of the expert system are computed using the evaluation methodology proposed in this report. The methodology provides a means of evaluating the propagation of the uncertainty in the expert system's knowledge base to the degree of uncertainty of the outputs produced by the system. *The ability to compute the uncertainty in the knowledge base makes it possible to quantify the accuracy of the decisions being made by the expert system.* For realistic problems where the knowledge base is large and where knowledge of the a priori input assertion errors and rule errors is limited, sensitivity analyses can provide greater insight into the impact of error propagation through the expert system's knowledge base. In this way the system designer will know which rules and input assertions are most critical to the decision-making performance of the expert system and thus shows the designer where effort should be made in reducing potential errors.

1.5 Overview.

The remainder of this report is divided into 4 sections. In Section 2, we describe the design of the rudimentary expert system that was used in this study. In Section 3, a Markov model of the aircraft redundancy management system is developed that incorporates the expert system. This model is then used to demonstrate the effect of the expert system on the reliability of the system and to identify the important performance parameters of the expert system. In Section 4, we present the evaluation methodology and discuss the use of the methodology to perform sensitivity analyses. Section 5, contains an example of the evaluation methodology being applied to the aircraft redundancy management system described in Section 2. Section 6 contains our conclusions and recommendations for further research.

2 THE EXPERT SYSTEM

2.1 Motivation

In order to exercise and demonstrate the evaluation methodology it was necessary to develop a test case expert system. The test case system had to satisfy several important criteria to be useful for this effort. First, the expert system had to be simple. It was important to keep the scope of the evaluation methodology to a modest effort because no previous work had been done in this area. Concentrating on simple problems was felt to be more productive than becoming too ambitious. Second, the expert system had to be oriented toward real-time applications because it is in this situation where rigorous evaluation is critical. For these two reasons it was decided to develop our own expert system to support the development of the evaluation methodology. Developing our own expert system had the advantage of allowing us to limit the scope of the problem to keep it simple.

2.2 The Application

The application selected for the demonstration expert system is the task of monitoring the decision making process of an FDI algorithm onboard an aircraft with redundant control-surfaces. Such an aircraft can tolerate certain combinations of control surface failures and remain flying. The objective of this expert system is to reduce the false alarm rate of the FDI algorithm. False alarms in FDI algorithms can cause unfailed resources to be taken off-line, which may degrade the performance of the system and ultimately reduces system reliability. Therefore, reducing the adverse effects of false alarms can therefore increase the reliability of the aircraft.

FDI algorithms produce false alarms for three major reasons. First, it is necessary for FDI algorithms to be conservative because undetected failures (often referred to as uncovered failures) can often lead to system loss. Second, it is necessary for FDI algorithms to make fast decisions using minimum amounts of data because of the extremely short time constraints of most aircraft control systems. This increases the likelihood that noisy sensor measurements will cause false alarms. Third, it is often difficult for FDI algorithms to isolate failures to the faulty component once they are detected. For example, it is difficult to distinguish between flap and aileron failures.

The expert system used in this study is intended to operate in parallel with an FDI algorithm. The expert system will identify false alarms and will allow unfailed resources to be returned to active use. This expert system will be referred to as the FDI Monitor expert system. The purpose of this expert system is to monitor the performance of the traditional FDI system in an attempt to discover false alarms. In effect, the FDI Monitor expert system's job is to scrutinize the decisions of the FDI system and make sure that the FDI does not remove unfailed resources from the configuration. This application was selected because it is a real-time application for an expert system and because the rules could be developed in the short time frame of the study. The expert system can do a better job of assessing the failure status of the resources than the FDI because of two advantages. First, the expert system will have seconds instead of milliseconds (as does the FDI) to make its decisions. This will allow the expert system to base its decisions on many measurements, reducing the effects of measurement errors on its decisions. Second, the expert system will

be given the authority to command small maneuvers to help correctly isolate failures. It is assumed that the expert system will not command maneuvers that can crash the aircraft.

The complete redundancy management system for the aircraft flight control system consists of the FDI algorithm, the Reconfiguration algorithm and the Expert System as shown in Figure 1. Both the FDI system and the Expert System make decisions about the status (component failures) of the aircraft. The FDI system makes its decisions at a high frequency using limited data, while the Expert System makes its decisions at a low frequency using large amounts of data. The Reconfiguration algorithm takes the current indication of the system's status (which can be updated by either the FDI system or the Expert System) and reconfigures the system to (1) remove newly declared *failed* components from active use and to (2) restore newly declared *functional* components to active use. The FDI system is responsible for declaring components to be failed, while the Expert System is responsible for declaring components to be functional. In this implementation, the Expert System is only concerned with examining the failure status of components that have previously been identified as failed by the FDI. Therefore, the Expert System will only be concerned with a small number of the total components in the system and in the case where no failures have been declared by the FDI, the Expert System will be idle.
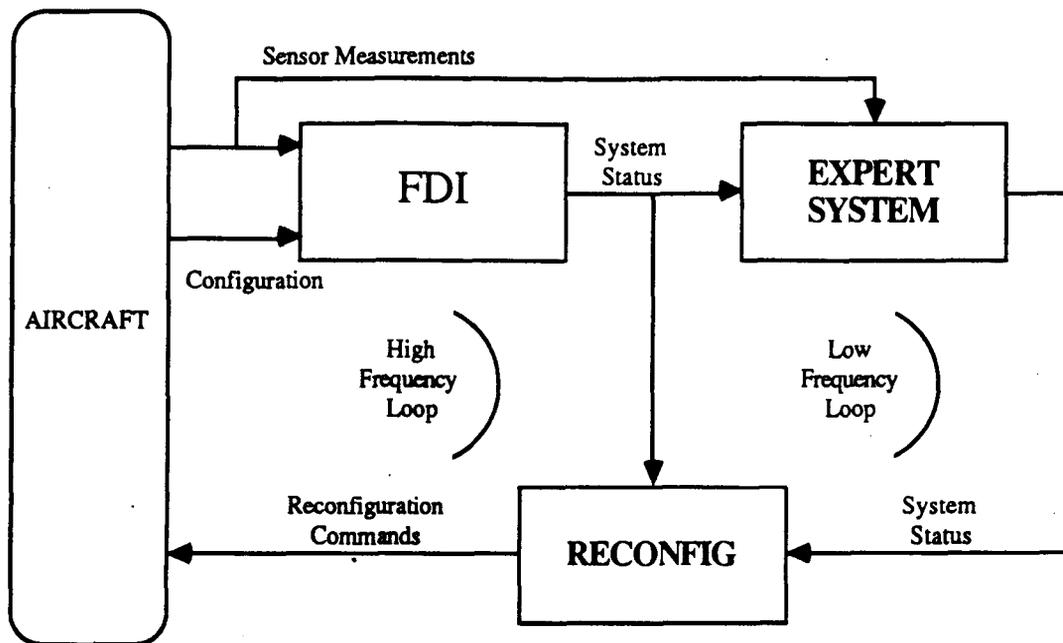

FIGURE 1. ARCHITECTURE OF THE EXPERT SYSTEM REDUNDANCY MANAGEMENT

The scope of the redundancy management problem has been limited to the longitudinal flight control surfaces of a fighter aircraft. We assume the aircraft to have 2 elevators and 2 canards and that the aircraft is controllable using any three of the four surfaces. We also assume that an undetected failure of any surface leads to the loss of the aircraft. These assumptions were made to limit the complexity of the problem and do not affect any of the results that are presented in this report. The primary motivation for the simplifications were to keep the scope of the problem within the resources and intent of the study.

2.3 The FDI Monitor Expert System.

The FDI Monitor Expert System operates whenever the FDI algorithm indicates that a failure of a control surface has occurred. This expert system was built to support the development of the evaluation methodology and is not meant to represent an actual design for a real aircraft. The Expert System operates in two stages. First, filtered measurements and predictions about aircraft pitch rate and control surface deflections are generated. Then the Expert System uses this information in combination with its knowledge base to make decisions about the status of the control surface in question. For the remainder of this discussion the index (i) will refer to a control surface where $i = 1$ => port canard, $i = 2$ => starboard canard, $i = 3$ => starboard elevator and $i = 4$ => port elevator.

To clarify what is occurring internal to the expert system, we define four attributes associated with each control surface (i) ,A_{ij} ;$j = 1,2,3,4$:

A_{i1} = The angular deflection of control surface (i) is large.

A_{i2} = The pitch rate response to a dither on control surface (i) is too small.

A_{i3} = The difference between the measured and predicted angular deflection of control surface (i) is small.

A_{i4} = The difference between the measured and predicted pitch rate response to a dither on control surface (i) is large.

The filtered measurements and predictions that are used by the expert system in making decisions are referred to here as *evidence*. For each control surface i there are four pieces of evidence that are used by the Expert System, $E_{i,k}$, $k = 1,2,3,4$:

$E_{i,1}$ = abs[delta(i)$_{measured}$]
    = The absolute value of the measured angular deflection of control surface (i)

$E_{i,2}$ = abs[theta-dot $_{measured}$]
    = The absolute value of the measured pitch rate of the aircraft resulting from a dither command applied to control surface (i)

$E_{i,3}$ = abs[delta(i)$_{measured}$ - delta(i)$_{predicted}$]
    = The absolute value of the difference between the measured angular deflection of control surface (i) and the predicted angular deflection of control surface (i)

$E_{i,4}$ = abs[theta-dot $_{measured}$ - theta-dot $_{predicted}$]
    = The absolute value of the difference between the measured and predicted pitch rate of the aircraft resulting from a dither command applied to control surface (i)

For our purpose, it is not important how the evidence is generated. We assume that the evidence can be produced and that, in general, there will be errors in the evidence as the result of measurement and modelling errors.

After the evidence is generated, the expert system uses the evidence in reasoning about the state of the aircraft. In particular, the inferences drawn by the expert system depend on whether or not each of the four pieces of evidence falls within specified

intervals. Before presenting the rules, threshold parameters that delimit those intervals for making these decisions are defined. These parameters represent some of the "expert" knowledge that is contained in the expert system's knowledge base. We emphasize that there is nothing sacred about the values assigned to these parameters. They are merely meant to be a representative set of values that might be used for this application. In the following we use $T_{k,low}$ and $T_{k,high}$ to represent the thresholds for the kth evidence type.

$$T_{1,low} = 4.5 \text{ deg} \qquad\qquad T_{1,high} = 5.5 \text{ deg}$$

$$T_{2,low} = 0.8 \text{ deg/sec} \qquad\qquad T_{2,high} = 1.2 \text{ deg/sec}$$

$$T_{3,low} = 0.4 \text{ deg} \qquad\qquad T_{3,high} = 0.6 \text{ deg}$$

$$T_{4,low} = 0.7 \text{ deg/sec} \qquad\qquad T_{4,high} = 1.3 \text{ deg/sec}$$

These threshold values imply three evidence intervals: evidence that falls below the lower threshold (low interval), evidence that falls between the two thresholds (mid interval) and evidence that falls above the higher thresholds (high interval).

The expert system assigns probabilities to each of the possible states of the surface that has been declared to be failed by the FDI system (below that surface is represented by the index i). These assignments are based on which intervals the various pieces of evidence have fallen into. Here too, these probabilities are assumed to be representative of "expert" knowledge and should be approximately the correct values.

The following 16 rules make up the FDI monitor expert system:

Rules for assigning a probability to internal surface state 1:

(1) **IF** $E_1 < T_{1,low}$ **THEN**   Pr(State of surface i = S_{i1}) = 0.9

(2) **IF** $T_{1,low} < E_1 < T_{1,high}$ **THEN**   Pr(State of surface i = S_{i1}) = 0.2

(3) **IF** $E_1 > T_{1,high}$ **THEN**   Pr(State of surface i = S_{i1}) = 0.014

Rules for assigning a probability to internal surface state 2:

(4) **IF** $E_2 < T_{2,low}$ **THEN**   Pr(State of surface i = S_{i2}) = 0.9

(5) **IF** $T_{2,low} < E_2 < T_{2,high}$ **THEN**   Pr(State of surface i = S_{i2}) = 0.3

(6) **IF** $E_2 > T_{2,high}$ **THEN**   Pr(State of surface i = S_{i2}) = 0.012

Rules for assigning a probability to internal surface state 3:

(7) **IF** $E_3 < T_{3,low}$ **THEN**   Pr(State of surface i = S_{i3}) = 0.8

(8) **IF** $T_{3,low} < E_3 < T_{3,high}$ **THEN**   Pr(State of surface i = S_{i3}) = 0.3

(9) **IF** $E_3 > T_{3,high}$ **THEN**   Pr(State of surface i = S_{i3}) = 0.1

Rules for assigning a probability to internal surface state 4:

(10) **IF** $E_4 < T_{4,low}$ **THEN**   Pr(State of surface i = S_{i4}) = 0.02

(11) **IF** $T_{4,low} < E_4 < T_{4,high}$ **THEN**   Pr(State of surface i = S_{i4}) = 0.2

(12) **IF** $E_4 > T_{4,high}$ **THEN**   Pr(State of surface i = S_{i4}) = 0.85

(13) **IF** True   **THEN** Pr(Surface i is stuck) = $Pr(S_1) \times Pr(S_2)$

(14) **IF** True   **THEN** Pr(Surface i is partially lost) = $[1 - Pr(S_3)] \times Pr(S_4)$

Decision Rules:

(15) **IF** Pr(Stuck) + Pr (Lost) < Thr   **THEN**   DECISION = Veto FDI

(16) **IF** Pr(Stuck) + Pr (Lost) > Thr   **THEN**   DECISION = Support FDI

Pr(Surface i is stuck) and Pr(Surface i is partially lost) are internal knowledge states which indicate the probability of the control surface being stuck or having lost some of its nominal effectiveness. These probabilities are compared to a threshold (Thr) in making the decision to override the FDI decision to declare the surface failed. The output of the expert system is the value of DECISION, which either vetos or supports the failure indicated by the FDI algorithm. It is the quality of this decision which will ultimately characterize the performance of this expert system.

3 MARKOV MODELS OF SYSTEMS WITH EXPERT SYSTEMS

3.1 Background.

It is imperative that complex systems performing life-critical missions be desensitized to imperfections and/or failures in their components and subsystems. This requirement has led to the development of fault tolerant systems that are designed to tolerate failures and remain operating safely. The design of a fault tolerant system requires the use of analytical techniques to verify the reliability of the system design. These techniques enable a designer to compute the reliability and expected performance of a system using reliability information about the system's components and subsystems. One of the most powerful techniques available for performing these types of analyses is the *Markov model* [21].

Markov reliability models are developed by representing the system's status by a finite number of states, each of which represents an operational mode of the system. That is, each operational state corresponds to a combination of failed and functioning components of the system. At any instance during its operation, the system is operating in one of these states. Initially, a system will begin operation in a state corresponding to no failed components or subsystems. As the system continues to operate, it may experience failures and thus, transition to other states representing operational modes with failed components and/or subsystems. The rate at which these transitions occur is a function of the reliability and performance of the components and subsystems making up the complete system. A Markov model defines the operational states of a system and the transition rates between these states, and thus enables a designer to evaluate the reliability and performance of the overall system.

3.2 Markov Model of the Aircraft Longitudinal Flight Control RMS.

Markov models have been used to evaluate the reliability of a large variety of life-critical and fault tolerant systems consisting of many types of components and subsystems. In this section we demonstrate how a Markov model is used to model the aircraft longitudinal flight-control redundancy management system (RMS) described in Section 2.2. Note that the simplicity of the Markov model here reflects the limited scope of the problem that was required to demonstrate the evaluation methodology developed here. In general a Markov model would consist of many more states than the 3 and 5 state models shown here. The Markov model of the RMS without the expert system is shown in Figure 2. In that figure XF refers to failure level X (i.e., X is the number of failed components), FA denotes false alarm, DET is shorthand for detected and identified failure and SL represents a system loss state.

To keep this example simple, we have assumed that the aircraft has 4 control surfaces of which at least three must be functional for the system to be functional. State 1 of the Markov model corresponds to the condition that all surfaces are functioning properly. The system can transition out of state 1 when (1) the FDI detects a failure or the FDI gives a false alarm or (2) when a failure goes undetected by the FDI. In the first situation, the RMS removes the appropriate surface from use and the aircraft continues to fly, (although at a reduced level of performance). In the second situation (an undetected failure), we conservatively assume a system loss.

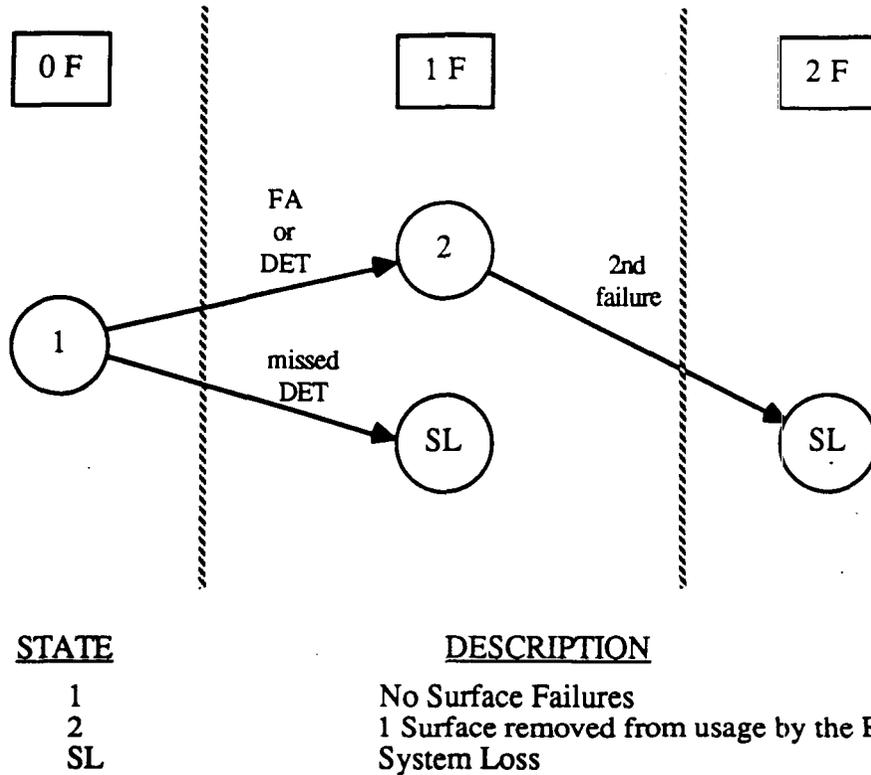| STATE	DESCRIPTION
1	No Surface Failures
2	1 Surface removed from usage by the RMS
SL	System Loss

FIGURE 2. MARKOV MODEL OF THE RMS WITHOUT THE EXPERT SYSTEM

From this model it can be seen that the probability of a missed detection contributes directly to the unreliability of the system. To reduce the effects of this failure mode, an FDI algorithm will typically have lower detection thresholds such that the probability of missed detection is very small. Reducing the missed detection rate will, unfortunately, increase the false alarm rate. This means that part of the time, the aircraft will be operating in state 2 (one surface not being used) even though the surface has not failed. As a result, the aircraft is (unnecessarily) sacrificing performance for the sake of reliability. The purpose of the expert system is to reduce the false alarm rate of the RMS.

The state transition matrix for this model can be specified in terms of the performance parameters of the aircraft flight control surfaces and the FDI algorithm. Let λ be the failure rate of the flight control surfaces, let μ be the false alarm rate of the FDI and let β be the conditional probability that the FDI detects a failure given that a failure has occurred. Using these definitions, the state transition matrix for the Markov model described above is given by

$$\Phi(\Delta t) = \begin{bmatrix} 1 - \mu\,\Delta t - 4\,\lambda\,\Delta t & 0 & 0 \\ \mu\,\Delta t + 4\,\lambda\,\Delta t\,\beta & 1 - 3\,\lambda\,\Delta t & 0 \\ 4\,\lambda\,\Delta t\,(1 - \beta) & 3\,\lambda\,\Delta t & 1 \end{bmatrix} \qquad (3.2\text{-}1)$$

where $\Phi_{ij}(\Delta t)$ is the probability of transitioning from state j to state i over the time interval of length Δt. The factor of four accounts for the four control surfaces. Here we have assumed that all surfaces have the same failure rate λ and we have aggregated the two

system loss states into a single state (the third element of the state transition matrix.) Note that the expression in (3.2-1) is an approximation for the state transition probability matrix that holds for small Δt, (e.g. $\Delta t < 0.02$ hours; during which the probability of two simultaneous failures is small enough to be neglected).

The state transition probability matrix is used to evaluate the probability that the system is in a particular operational state as a function of time as follows. For example, assume $\lambda = 1.0e\text{-}5$ failures/hour, $\mu = 1.7248e\text{-}4$ false alarms/hour, $\beta = 0.98$ and use $\Delta t = 0.0167$ hours (1 minute), then the state transition matrix for the system is given by

$$\Phi(0.167 \text{ hour}) = \begin{bmatrix} 0.999996453333 & 0 & 0 \\ 3.533333e\text{-}6 & 0.9999995 & 0 \\ 1.3334e\text{-}8 & 5.0e\text{-}7 & 1 \end{bmatrix} \quad (3.2\text{-}2)$$

The reliability of the system is defined as the probability that the system is in a non-system-loss state. This reliability can be computed for the time interval of interest as follows. Define the state probability vector $\pi(t \mid \pi_0)$ as the vector whose i^{th} element is the probability of the system being in state i at time t given an initial state probability distribution at time 0 of π_0 (the explicit conditioning on π_0 will be eliminated from here on). The value of $\pi(t)$ is computed by raising the state transition matrix to the appropriate power (i.e., $t/\Delta t$) and multiplying by the initial distribution π_0:

$$\pi(1 \text{ hour}) = \Phi(0.167 \text{ hour})^{60} \pi_0 \; ; \quad \pi_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad (3.2\text{-}3)$$

where here we have assumed that the aircraft begins operating in the no failure state at t_0. Performing this calculation, one can show that the state probability vector for the system after 1 hour of operation is given by

$$\pi(1 \text{ hour}, \beta = 0.98) = \begin{bmatrix} 0.9998677 \\ 2.1166325e\text{-}4 \\ 8.030689e\text{-}7 \end{bmatrix} \quad (3.2\text{-}4)$$

Similarly, one can show that the state probability vector for the system after 1 hour of operation for the case of perfect FDI coverage ($\beta = 1$) is given by

$$\pi(1 \text{ hour}, \beta = 1) = \begin{bmatrix} 0.999789 \\ 2.12375e\text{-}4 \\ 3.1326592e\text{-}9 \end{bmatrix} \quad (3.2\text{-}5)$$

The reliability at one hour is the sum of the probabilities of being in states 1 and 2 (the non-system-loss states). This reliability measure will be used later as a means of comparing the flight-control system with and without the expert system.

3.3 Incorporating the Expert System into the Markov Model.

After a Markov model has been developed for the overall system, the next step in the evaluation process is to integrate a quantitative characterization of the performance of the expert system into that model. The resulting Markov model for the Redundancy Management System (RMS) with the embedded expert system is shown in Figure 3. Integrating the expert system into the original Markov model requires the introduction of two new states (A and B). State A corresponds to the situation where the FDI has generated a false alarm. State B corresponds to the situation where the FDI correctly detects a failure of one
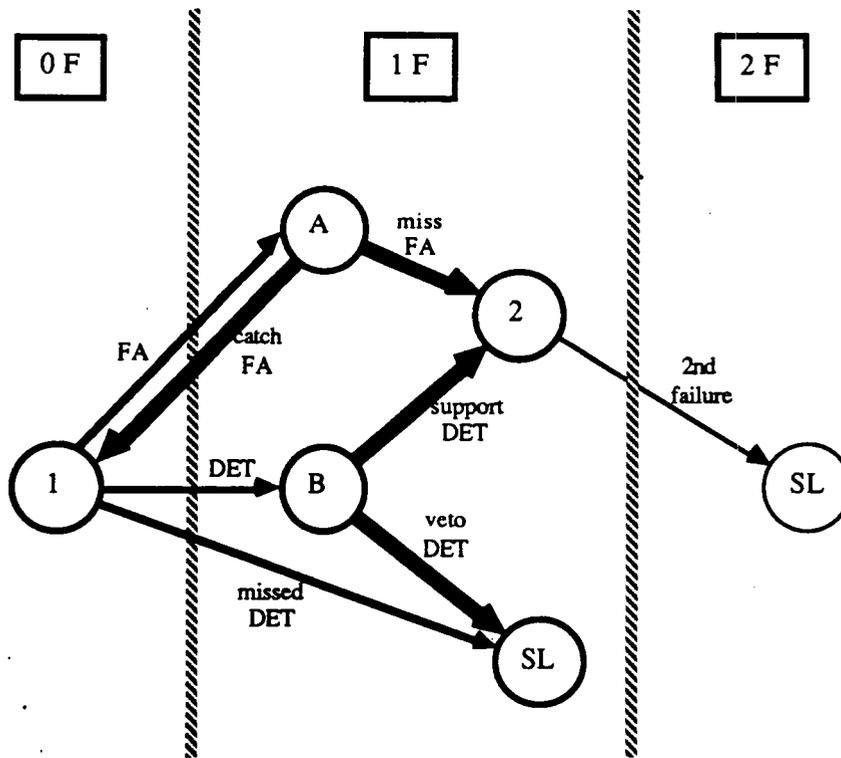
STATE	STM	DESCRIPTION
1	1	No Surface Failures
A	2	No Surface Failures, FDI indicates failure (FA)
B	3	1 Surface failure, FDI detects failure (DET)
2	4	1 Surface removed from usage by the RMS
SL	5	System Loss

Transitions due to the FDI and Failures

Transitions due to the Expert System

FIGURE 3. MARKOV MODEL OF THE RMS WITH EXPERT SYSTEM

of the surfaces. These two additional states are needed to capture the impact of the expert system on the reliability of the RMS. In the original RMS, there was no need to differentiate, in terms of system performance, between a correctly detected failure and a false alarm. For both cases, the RMS system removes a surface from active use. Of course in the case of a false alarm there is no need to do so. For the case of the RMS with the embedded expert system, this no longer holds true. A False alarm by the FDI does not imply that the system automatically transitions to state 2. There is now the possibility that the expert system will catch the false alarm and return the surface to active use. This transition (from state A back to state 1) can be thought of as a repair mode for the RMS that is the result of adding the expert system to the RMS.

Of course there is a caveat. The expert system can improve system performance and reliability by correcting false alarms, but it can also decrease performance and reliability by incorrectly declaring an actual failure to be a false alarm. This failure mode, due to erroneous decisions made by the expert system, is indicated by the transition from state B to system loss. (Here we assume that when the expert system incorrectly declares a false alarm the effect on the system is the same as an undetected failure, which we assumed to be a system loss.) It should be noted that there are no transitions from either state A or B to a system loss state at the second failure level. This occurs because we have assumed that the time span between entering states A or B and exiting these states is sufficiently small to neglect the possibility of a failure during this period. This assumption was made only for simplicity. If desired, such transistions could be accounted for within the Markov model.

It can be seen that the impact of the expert system on the overall performance of the system depends on the two performance parameters **P(correcting a false alarm)** and **P(vetoing a correctly identified failure)**. Ideally one would like the expert system to be designed so that **P(correcting a false alarm) = 1** and **P(vetoing a correctly identified failure) = 0**. These two parameters characterize the performance of the expert system discussed in this example. In general, the performance parameters for any given expert system will differ from the performance parameters of our example system; however, they can be obtained by constructing a Markov model in a similar manner to that outlined above. In Section 4, we discuss our approach to computing these expert system performance parameters.

In addition to the performance parameters of the aircraft flight control surfaces and the FDI algorithm (λ, μ and β), the state transition matrix for the Markov reliability model that includes the expert system is also specified in terms of the expert system performance parameters: P(correcting a false alarm) and P(vetoing a correctly identified failure). Let α = P(correcting a false alarm) and let ζ = P(vetoing a correctly identified failure). Using these definitions, the state transition matrix for this Markov model is given by

$$
\Phi(\Delta t) = \begin{bmatrix} 1 - \mu\,\Delta t - 4\lambda\,\Delta t & \alpha & 0 & 0 & 0 \\ \mu\,\Delta t & 0 & 0 & 0 & 0 \\ 4\lambda\,\Delta t\,\beta & 0 & 0 & 0 & 0 \\ 0 & (1-\alpha) & (1-\zeta) & 1 - 3\lambda\,\Delta t & 0 \\ 4\lambda\,\Delta t\,(1-\beta) & 0 & \zeta & 3\lambda\Delta t & 1 \end{bmatrix} \quad (3.3\text{-}1)
$$

where the states indicated in Figure 3 have been ordered 1, A, B, 2, SL. The state transition matrix for this system differs from typical state transition matrices in that it contains a nonzero element in the upper right diagonal $[\Phi_{12}(\Delta t) = \alpha]$. Transition probabilities in the lower left diagonal represent transitions to states of *increased* failures, while transition probabilities in the upper right diagonal represent transitions to states of *decreased* failures. For many systems it is not possible to repair a failure during operation and therefore the upper right diagonal typically consists of zeros.

The reliability and performance of the new system that includes the expert system can be evaluated in the same manner as shown in Section 3.2. However, before this evaluation can be performed, it is necessary to assign values to the key expert system performance parameters in a systematic fashion. The determination of these expert system performance parameters is discussed in the next section.

4 EVALUATION OF EXPERT SYSTEMS

4.1 Rule-Based Expert Systems.

A rule-based expert system is a reasoning program consisting of a knowledge base and an inference engine. The knowledge base consists of an assertion base - a collection of assertions (facts) about the current state of the world, and a rule base - a set of IF-THEN rules that operate on the assertions contained in the assertion base. The inference engine is a control mechanism which selects and applies rules to the assertion base in order to generate new assertions and/or decisions (actions). Viewed as a black box, a rule-based expert system generates assertions and decisions from incoming real time assertions. The performance of rule-based expert systems is characterized by the accuracy of its assertions as shown in Figure 4 and by the quality of its decisions. The objective in developing an evaluation methodology is to quantify what is meant by assertion accuracy and decision quality.
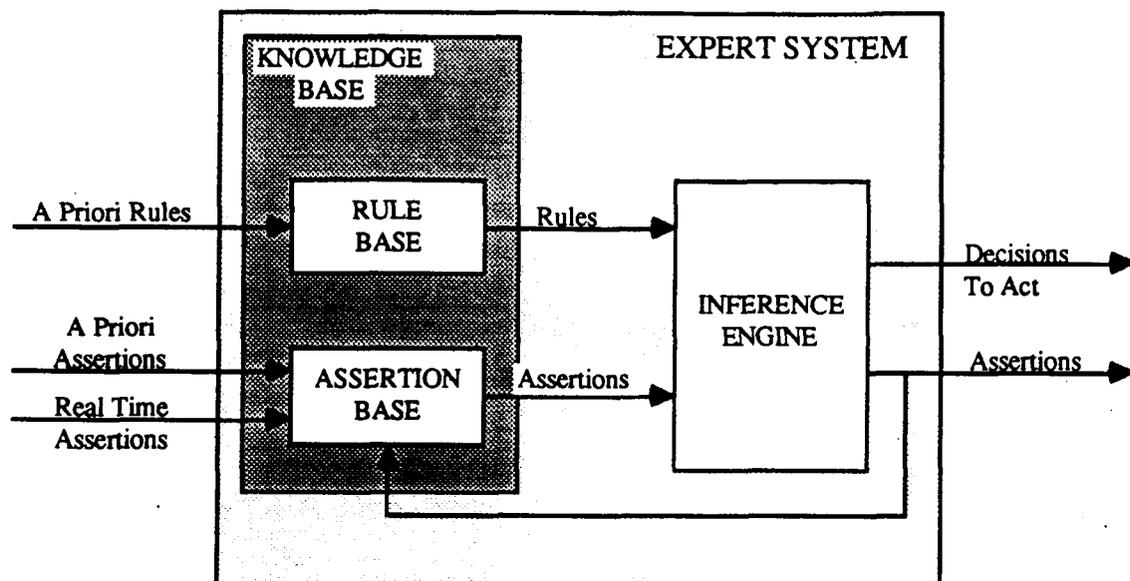

FIGURE 4. BLOCK DIAGRAM OF A GENERIC RULE-BASED EXPERT SYSTEM

Any errors in the output of the expert system (assuming that there are no errors in the inference engine or logical errors in the software) are due to errors in the rule base (the *a priori* rules) and errors in the assertion base (the *a priori* and real-time assertions). Therefore, the analysis presented here focussed on how to quantify output errors as a function of these two classes of input errors.

4.2 Knowledge Base Errors.

The performance of an expert system is a function of the uncertainty in the expert system's knowledge base. It is necessary to model this uncertainty as it evolves in time if we are to successfully evaluate the performance of expert systems in real-time environments. As mentioned in the introduction, uncertainties in the knowledge base occur as the result of assertion errors and rule errors.

The effect of an *Assertion Error* occurs when an inference is made using faulty data. For example suppose we have a rule of the form: **IF A_1 and A_2 THEN A_3.** When the knowledge base of the expert system holds A_1 and A_2 to be true and, in reality, A_1 is false, then A_3 will be held true even though in reality it may not be true. The error in A_3 is the result of an assertion error in A_1. Thus, the effect of such an assertion error can become magnified as it is propagated through the system.

The effect of a Rule *Error* occurs when an inference is made using a faulty rule. For example suppose we again consider the rule **IF A_1 and A_2 THEN A_3.** In reality, this inference may not always be true. There may be some situations where A_1 and A_2 do not imply A_3. In this case, errors have been introduced into the knowledge base as a result of the faulty rule and not because of faulty data. Of course assertion errors and rule errors may also serendipitously interact with each other to cancel each others erroneous effects and thus lead to a correct inference in spite of the knowledge base errors. The purpose of developing the evaluation methodology is to quantify the effects of the propagation of errors through the knowledge base due to both sources. We do this using a probabilistic model of uncertainty.

4.3 Error Model.

We now present an error model for rules and assertions in an expert system of the form: **If A_{in} THEN A_{out}** where A_{in} and A_{out} are the input (antecedent) and output (consequent) assertions of the rule, respectively. Before we present the error mode we first define our notation. We assume that the expert system holds assertions A_{in} and A_{out} to be true with probabilities $P_{xs}(A_{in})$ and $P_{xs}(A_{out})$, respectively, while in reality A_{in} and A_{out} will be true with probabilities $P(A_{in})$ and $P(A_{out})$. Assertion errors are represented as the difference between the true probability and the expert system's probability and are defined using the Δ notation as follows:

$$\Delta P(A_{in}) = P(A_{in}) - P_{xs}(A_{in}) \tag{4.3-1}$$

$$\Delta P(A_{out}) = P(A_{out}) - P_{xs}(A_{out}) \tag{4.3-2}$$

Similar conventions could be adopted for systems employing certainty factors or belief functions to represent uncertainty. Indeed, the expert system will rarely hold the correct (truth) values for the certainty factors for the former or support and plausibilities for the latter.

The accuracy of a rule can be captured by the conditional probabilities $P(A_{out} \mid A_{in})$, $P(A_{out} \mid \sim A_{in})$, and $P_{xs}(A_{out} \mid A_{in})$, $P_{xs}(A_{out} \mid \sim A_{in})$ where again no subscript represents the correct or truth values and the subscript XS corresponds to the values assumed by the expert system. Rule errors are defined as the difference between the true conditional probabilities and the conditional probabilities assumed by the expert system. The notation $\sim A_{in}$ denotes the event **not(A_{in})**. Following the convention in (4.3-1) and (4.3-2), rule

errors are expressed as:

$$\Delta P(A_{out} \mid A_{in}) = P(A_{out} \mid A_{in}) - P_{xs}(A_{out} \mid A_{in}) \qquad (4.3-3)$$

$$\Delta P(A_{out} \mid \sim A_{in}) = P(A_{out} \mid \sim A_{in}) - P_{xs}(A_{out} \mid \sim A_{in}) \qquad (4.3-4)$$

For the rule **If A_{in} THEN A_{out}** it follows that $P_{xs}(A_{out} \mid A_{in}) = 1$. If the rule were modified to read **If A_{in} THEN A_{out} with $P(A_{out}) = 0.7$**, then $P_{xs}(A_{out} \mid A_{in}) = 0.7$. Note that our notation allows us to work with expert systems that reason using uncertainty.

The preceding definitions establish the notation for representing errors in both assertions and rules. Again using the generic rule: **If A_{in} THEN A_{out}**, we now present the method for computing the error in the output of the rule [$\Delta P(A_{out})$], in terms of the error notation described above. $\Delta P(A_{out})$ is computed by first recalling its definition

$$\Delta P(A_{out}) = P(A_{out}) - P_{xs}(A_{out}) \qquad (4.3-5)$$

Using the law of total probability, the actual and expert system probabilities for the assertion A_{out} can be expressed as

$$P(A_{out}) = P(A_{out} \mid A_{in}) P(A_{in}) + P(A_{out} \mid \sim A_{in}) P(\sim A_{in}) \qquad (4.3-6)$$

$$P_{xs}(A_{out}) = P_{xs}(A_{out} \mid A_{in}) P_{xs}(A_{in}) + P_{xs}(A_{out} \mid \sim A_{in}) P_{xs}(\sim A_{in}) \qquad (4.3-7)$$

Substituting the appropriate error definitions into these two equations and manipulating leads to the following result

$$
\begin{aligned}
\Delta P(A_{out}) = {} & [\, P_{xs}(A_{out} \mid \sim A_{in}) - P_{xs}(A_{out} \mid A_{in}) \,] \, \Delta P(A_{in}) && ;\text{assertion errors} \\
& + P_{xs}(A_{in}) \, \Delta P(A_{out} \mid A_{in}) + P_{xs}(\sim A_{in}) \, \Delta P(A_{out} \mid \sim A_{in}) && ;\text{rule errors} \\
& + \Delta P(A_{in}) \, \Delta P(A_{out} \mid A_{in}) + \Delta P(\sim A_{in}) \, \Delta P(A_{out} \mid \sim A_{in}) && ;\text{higher order terms} \\
&&& (4.3.8)
\end{aligned}
$$

The error in the output assertion A_{out} in (4.3-8) is written as (1) a first order term (linear) in input assertion errors [as shown on the first line], (2) a first order term (linear) in rule errors [as shown on the second line], and (3) a higher order term of products of the assertion errors and rule errors [as shown on the third line].

The result that we have just developed holds for the case where A_{in} is a compound event of the form $A_{in} = F(A_1, A_2, \ldots A_n)$ where $F(\ )$ refers to any function involving the Boolean operations **AND, OR,** and **NOT**. In this situation, however, it is necessary to compute the error in A_{in} in terms of the errors in $A_1, A_2, \ldots A_n$. In order to illustrate this process, expressions for the input assertion errors are developed for the Boolean operations listed above.

First, recall the following definitions:

$$\Delta P(A_1) = P(A_1) - P_{xs}(A_1) \qquad (4.3-9)$$

$$\Delta P(A_2) = P(A_2) - P_{xs}(A_2) \qquad (4.3-10)$$

$$\Delta P(A_1 A_2) = P(A_1 A_2) - P_{xs}(A_1 A_2) \qquad (4.3-11)$$

N O T : The assertion error for A_{in} when $A_{in} = \sim A_1$ is given by

$$\Delta P(A_{in}) = -\Delta P(A_1) \tag{4.3-12}$$

A N D :   The assertion error for A_{in} when $A_{in} = A_1 \text{ AND } A_2$ is given by

$$\Delta P(A_{in}) = P(A_1 A_2) - P_{xs}(A_1 A_2) = \Delta P(A_1 A_2) \tag{4.3-13}$$

This characterization of $\Delta P(A_{in})$ contains error probabilities of joint events. To avoid requiring the expert system to explicitly carry knowledge of these joint events, it will be useful to use a linear approximation to this error term as follows. Employing conditional probabilities we can write

$$P(A_{in}) = P(A_1 A_2) = P(A_2 \mid A_1) P(A_1) \tag{4.3-14}$$

$$P_{xs}(A_{in}) = P_{xs}(A_1 A_2) = P_{xs}(A_2 \mid A_1) P_{xs}(A_1) \tag{4.3-15}$$

subtracting (4.3-15) from (4.3-14) and using the definitions (4.3-1) and (4.3-2) gives

$$\Delta P(A_{in}) = \Delta P(A_1 A_2) = P(A_2 \mid A_1) P(A_1) - P_{xs}(A_2 \mid A_1) P_{xs}(A_1) \tag{4.3-16}$$

Recall that by definition $P(A_1) = P_{xs}(A_1) + \Delta P(A_1)$, therefore:

$$\Delta P(A_{in}) = P(A_2 \mid A_1) [P_{xs}(A_1) + \Delta P(A_1)] - P_{xs}(A_2 \mid A_1) P_{xs}(A_1) \tag{4.3-17}$$

$$\Delta P(A_{in}) = P(A_2 \mid A_1) \Delta P(A_1) + [P(A_2 \mid A_1) - P_{xs}(A_2 \mid A_1)] P_{xs}(A_1) \tag{4.3-18}$$

Also recall that by definition $\Delta P(A_2 \mid A_1) = [P(A_2 \mid A_1) - P_{xs}(A_2 \mid A_1)]$[1]. Using this definition, (4.3-18) can be rewritten as

$$
\begin{aligned}
\Delta P(A_{in}) &= P(A_2 \mid A_1) \Delta P(A_1) + P_{xs}(A_1) \Delta P(A_2 \mid A_1) \\
&= P_{xs}(A_2 \mid A_1) \Delta P(A_1) + P_{xs}(A_1) \Delta P(A_2 \mid A_1) \quad \text{;linear in error terms} \\
&\quad - \Delta P(A_2 \mid A_1) \Delta P(A_1) \quad \text{;nonlinear in error terms}
\end{aligned}
\tag{4.3-19}
$$

Neglecting the nonlinear terms gives an expression for $\Delta P(A_{in})$ that is first order in error terms.

O R :   The assertion error for A_{in} when $A_{in} = A_1 \text{ OR } A_2$ is given by

$$
\begin{aligned}
\Delta P(A_{in}) &= \Delta P(A_1) + \Delta P(A_2) - \Delta P(A_1 A_2) \\
&= \Delta P(A_2) + (1 - P_{xs}(A_2 \mid A_1)) \Delta P(A_1) - P_{xs}(A_1) \Delta P(A_2 \mid A_1) \quad \text{;linear} \\
&\quad + \Delta P(A_2 \mid A_1) \Delta P(A_1) \quad \text{;nonlinear}
\end{aligned}
\tag{4.3-20}
$$

Where (4.3-20) has been obtained by substituting for $\Delta P(A_1 A_2)$ from (4.3-19).

---

[1] Here we assume that the expert system's rule base is sufficiently rich to either explicitly or implicitly contain this conditional probability.

In the more general case, where F( ) is a more complicated Boolean function, the error in A_{in} can be computed by repeated applications of the preceding three results.

The preceding evaluation method can be used to compute the errors introduced into the knowledge base of the expert system by assertion and rule errors for the firing of a single rule. To compute the errors resulting from many inferencing cycles, it is necessary to apply the evaluation method to each rule as it is fired and keep track of the resulting *propagated* errors. A preliminary approach to the problem of error propagation follows.

4.4 Error Propagation

The error model defined in 4.3 demonstrates how assertion and rule errors can introduce additional errors into the knowledge base through the application of a single rule. In order to utilize this information to evaluate the performance of a rule based system, it is necessary to propagate the effects of uncertainty through an inferencing cycle (the application of many rules). There are two approaches that can be used to investigate the propagation of uncertainty through an inferencing cycle. The first approach is to compute the values of the output assertion errors $\Delta P(A_{out})$ as a function of the input assertion errors and the rule errors as described in Section 4.3. A difficulty encountered with this approach is that one must know with high accuracy the values of the input assertion errors and the rule errors. In other words, this method requires one to have a good understanding of the error sources for the rule based system of interest. In practice these error sources are not well understood and, therefore, will not be known to sufficient levels of accuracy to leave one confident in the computed value of the output assertion errors. However, in situations where the input assertion errors $\Delta P(A_{in})$ and the rule errors $\Delta P(A_{out} \mid A_{in})$ and $\Delta P(A_{out} \mid \sim A_{in})$ are well defined, then uncertainty can be propagated through the knowledge base by repeatedly applying the error propagation equations defined in 4.3 for each rule that fires.

An alternative approach for investigating the propagation of uncertainty through the knowledge base is to perform sensitivity analyses. In this case, the sensitivities of the values of $\Delta P(A_{out})$ to errors in both rules and a priori and real-time input assertions are computed rather than the actual values of $\Delta P(A_{out})$. That is, in situations where the input assertion errors and rule errors are not accurately known, sensitivity analyses can be used to determine those errors to which the quality of the information contained in the knowledge base is the most sensitive. In the remainder of this section we present a method by which a sensitivity analysis of a rule based system can be performed using the error models developed in 4.3.

Our discussion of the development of the sensitivity analysis begins by recalling Equation (4.3-8), the relation between output assertion errors and input assertion and rule errors for the firing of a given rule:

$$\Delta P(A_{out}) = [\ P_{xs}(A_{out} \mid \sim A_{in} - P_{xs}(A_{out} \mid A_{in})\ )\ ]\ \Delta P(A_{in}) \qquad \text{;assertion errors}$$
$$+\ P_{xs}(A_{in})\ \Delta P(A_{out} \mid A_{in}) + P_{xs}(\sim A_{in})\ \Delta P(A_{out} \mid \sim A_{in}) \qquad \text{;rule errors}$$
$$+\ \Delta P(A_{in})\ \Delta P(A_{out} \mid A_{in}) + \Delta P(\sim A_{in})\ \Delta P(A_{out} \mid \sim A_{in}) \qquad \text{;higher order terms}$$
$$(4.3.8a)$$

This relation can be expressed in shorthand notation as:

$$\Delta P(A_{out}) = [C_A]_{xs} \Delta P(A_{in}) + [C_{in}]_{xs} \Delta P(A_{out} \mid A_{in}) + [C_{\sim in}]_{xs} \Delta P(A_{out} \mid \sim A_{in})$$
$$+ \text{ higher order error terms} \qquad (4.3\text{-}8b)$$

where $[*]_{xs}$ represents a quantity that is explicitly known within the expert system. Neglecting the higher order error terms (i.e., assuming that the values of second order error terms can be neglected when compared to first order values), (4.3-8b) can be re-expressed as

$$\Delta P(A_{out}) \cong \left[[C_A]_{xs} : [C_{in}]_{xs} : [C_{\sim in}]_{xs} \right] \begin{bmatrix} \Delta P(A_{in}) \\ \cdots \\ \Delta P(A_{out} \mid A_{in}) \\ \cdots \\ \Delta P(A_{out} \mid \sim A_{in}) \end{bmatrix}$$

$$= [C]_{xs} \begin{bmatrix} \Delta P(A_{in}) \\ \cdots \\ \Delta P(A_{out} \mid A_{in}) \\ \cdots \\ \Delta P(A_{out} \mid \sim A_{in}) \end{bmatrix} \qquad (4.4\text{-}1a)$$

where $[C]_{xs} = \left[[C_A]_{xs} : [C_{in}]_{xs} : [C_{\sim in}]_{xs} \right]$

Equations (4.3-12), (4.3-19) and (4.3-20) illustrate that the term $\Delta P(A_{in})$ in (4.3-8b) can be replaced by the general form (again, this is an approximation that is linear in error terms):

$$\Delta P(A_{in}) = \left[[A]_{xs} : [A_{i|j}]_{xs} \right] \begin{bmatrix} \Delta P(A) \\ \cdots \\ \Delta P(A_i \mid A_j) \end{bmatrix} \qquad (4.4\text{-}1b)$$

where the nature of the terms $[A]_{xs}$ and $[A_{i|j}]_{xs}$ depends an the Boolean nature of A_{in} and where A is the (nx1) vector of all input and output assertions contained in the knowledge base,.and $\Delta P(A)$ is the (nx1) vector of the errors in those assertions. Similarly, $\Delta P(A_i \mid A_j)$ is the n^2x1 vector of errors in the conditional assertions for all combinations of the n input and output assertions.

Let N be the total number of rules and let $\Delta P(A_{out} \mid A_{in})$ and $\Delta P(A_{out} \mid \sim A_{in})$ be (Nx1) vectors of the rule errors. We define the uncertainty vector R to be the following composite vector representing all of the assertion and rule errors.

$$R = \begin{bmatrix} \Delta P(A) \\ \hline \Delta P(A_i \mid A_j) \\ \hline \Delta P(A_{out} \mid A_{in}) \\ \hline \Delta P(A_{out} \mid \sim A_{in}) \end{bmatrix} \tag{4.4-2}$$

The vector R represents the uncertainty in the knowledge base for both assertions and rules. Ultimately, the purpose of the sensitivity analyses is to determine the sensitivity of errors in A and any attendant decisions that are made as side effects to the errors R.

Let R_i be the uncertainty vector just before the ith rule firing of the inferencing process, and let R_{i+1} be the uncertainty vector after the ith rule firing. To perform a sensitivity analysis we will first relate the uncertainty vectors R_i, and R_{i+1} using linear transformations. In the following it is assumed that the kth rule is fired at the ith firing. Thus, we seek a relationship between R_i, and R_{i+1} of the following form:

$$R_{i+1} = S_i^k R_i \tag{4.4-3}$$

Here S_i^k is the sensitivity matrix corresponding to the ith firing of the kth rule. An expression for S_i^k can be obtained by substituting the expression for $\Delta P(A_{in})$ in (4.4-1b) into (4.4-1a) as follows.

$$\Delta P(A_{out})^k = \left[[\mathbb{C}_A]_{xs}^k \left[[A]_{xs}^k [A_{i|j}]_{xs}^k \right] : [\mathbb{C}_{in}]_{xs}^k : [\mathbb{C}_{\sim in}]_{xs}^k \right] \begin{bmatrix} \Delta P(A) \\ \hline \Delta P(A_i \mid A_j) \\ \hline \Delta P(A_{out} \mid A_{in}) \\ \hline \Delta P(A_{out} \mid \sim A_{in}) \end{bmatrix} \tag{4.4-4}$$

Since $\Delta P(A_{out})$ is an element of $\Delta P(A)$, the sensitivity matrix S_i^k can be constructed from an identity matrix with the row corresponding to A_{out} for the kth rule replaced by the row matrix on the right hand side of (4.4-4), namely,:

$$\mathbb{C} = \left[[\mathbb{C}_A]_{xs}^k \left[[A]_{xs}^k [A_{i|j}]_{xs}^k \right] : [\mathbb{C}_{in}]_{xs}^k : [\mathbb{C}_{\sim in}]_{xs}^k \right] \tag{4.4-5}$$

For example if A_{out} for the kth rule (the ith rule fired) were the first element of A, then S_i^k would be given by:

$$S_i^k = \begin{bmatrix} & & \mathbb{C} & & & \\ 0 & I & : & 0 & 0 & 0 \\ 0 & & : & I & 0 & 0 \\ 0 & & : & 0 & I & 0 \\ 0 & & : & 0 & 0 & I \end{bmatrix}$$ (4.4-6)

Thus, over several rule firings, we would have, for instance (omitting the rule indexing superscript):

$$R_{i+3} = S_{i+3} \, S_{i+2} \, S_{i+1} S_i \, R_i$$ (4.4-7a)

or in more compact notation:

$$R_{i+3} = S_{i+3|i} \, R_i$$ (4.4-7b)

where the *multi-step sensitivity matrix* $S_{i+3|i}$ is given by

$$S_{i+3|i} = S_{i+3} \, S_{i+2} \, S_{i+1} S_i$$ (4.4-7c)

Therefore, by storing the individual sensitivity matrices, one can compute the sensitivity or assertion errors to the errors existing at any point in the inferencing cycle. We now consider two cases: sensitivity to real-time input and a priori assertion errors and sensitivity to rule errors.

A Priori Assertion Errors

The sensitivities of assertion errors after the ith rule firing to a priori assertions (those held before inferencing has been initiated) are contained in the Multi-step sensitivity matrix $S_{i|0}$:

$$S_{i|0} = \prod_{j=1}^{i} S_j$$ (4.4-8)

That is, the uncertainty vector R_0 contains the uncertainties in the a priori assertions.

Real-Time Assertion Errors

The sensitivities to real-time assertions (those that may be acquired by, for instance, sensors during the middle of the inferencing process) are contained in the multi-step sensitivity matrix:

$$S_{ilm} = \prod_{j=m}^{i} S_j \qquad\qquad (4.4\text{-}9)$$

where m is the rule firing number corresponding to the time that the real-time assertions were acquired.

It should be noted that the sensitivities contained in the sensitivity matrix apply only to the specific sequence of rule firings associated with a specific set of a priori and real-time assertions. Thus, operation of the expert system under a variety of scenarios will result in unique sensitivity matrices for each scenario.

5 RESULTS

5.1 Problem Description.

Numerical results are presented for an evaluation of the aircraft longitudinal flight-control RMS described in Section 3. The example addresses both the evaluation of the stand-alone performance of the expert system as well as the performance of the larger redundancy management system in which it is embedded.

Several simplifying assumptions have been made to narrow the scope of this research effort. (For all matrix definitions, the subscript i refers to the column and j refers to the row.) The first assumption is that the aircraft control surfaces will fail in one of three ways. A control surface can (1) become stuck (2) float (i.e. flap in the breeze) and (3) be damaged (e.g. a piece of the control surface is torn off in battle). When the expert system is first invoked, we assume that either the FDI has correctly detected that a control surface is in in one of the preceding states, or that the FDI has produced a false alarm. We define S to be this control surface state vector and assume the following a priori probability distribution on these states when the expert system is invoked.

$$\pi_0 = \{ P(S_i) \} = \begin{bmatrix} \text{stuck} \\ \text{float} \\ \text{damage} \\ \text{false alarm} \end{bmatrix} = \begin{bmatrix} 0.01 \\ 0.09 \\ 0.02 \\ 0.88 \end{bmatrix} \quad (5.1\text{-}1)$$

The second assumption that we make concerns the conditional probability that an attribute is true given the state of the control surfaces as described above. As defined in Section 2.2 let $A(k)$ be the attribute vector for control surface k. We define the conditional probabilities that the jth attribute $A_j(k)$ is true given that a control surface is in state (i) to be

$$Pr[A_j(k) \mid S_i] = \begin{bmatrix} 0.80 & 0.90 & 0.50 & 0.50 \\ 0.10 & 0.90 & 0.50 & 0.50 \\ 0.50 & 0.50 & 0.80 & 0.80 \\ 0.10 & 0.10 & 0.50 & 0.10 \end{bmatrix} \text{ for all k} \quad (5.1\text{-}2)$$

It follows from these assumptions that the a priori distribution on the attribute vector A is given by

$$\{ P(A_k) \} = \begin{bmatrix} 0.11 \\ 0.18 \\ 0.51 \\ 0.15 \end{bmatrix} \quad (5.1\text{-}3)$$

The final assumptions that we make are to define the a priori conditional probabilities that the measured values of the evidence lie within the intervals defined in Section 2.3. Recall that in section 2.3 we defined intervals (here represented by I) for the evidence in

terms of thresholds. These probabilities indicate the probability that a measurement will fall within a given interval given that one of the attributes is true or is not true. These matrices are defined below

$$Pr(I_j \mid A_i) = \begin{bmatrix} 0.10 & 0.20 & 0.70 \\ 0.80 & 0.15 & 0.05 \\ 0.80 & 0.15 & 0.05 \\ 0.10 & 0.25 & 0.65 \end{bmatrix}$$

(5.1-4)

$$Pr(I_j \mid \sim A_i) = \begin{bmatrix} 0.89 & 0.10 & 0.01 \\ 0.02 & 0.08 & 0.90 \\ 0.20 & 0.35 & 0.45 \\ 0.80 & 0.18 & 0.02 \end{bmatrix}$$

(5.1-5)

respectively. Given these assumptions, we can then proceed to use the evaluation methodology to generate reliability and performance results for both the expert system and the complete flight control RM system.

5.2 Expert System Evaluation Results.

The direct evaluation methodology defined in Section 4 (not the sensitivity approach) was used to compute exact values for the performance parameters of the expert system, $\alpha = Pr$ (correcting a false alarm) and $\zeta = Pr$(vetoing a correctly identified failure). These results were generated for a range of values of **Thr**, the threshold parameter defined in Section 2.3 for rules 15 and 16 of the expert system. The values of these performance parameters are presented in Figure 5.
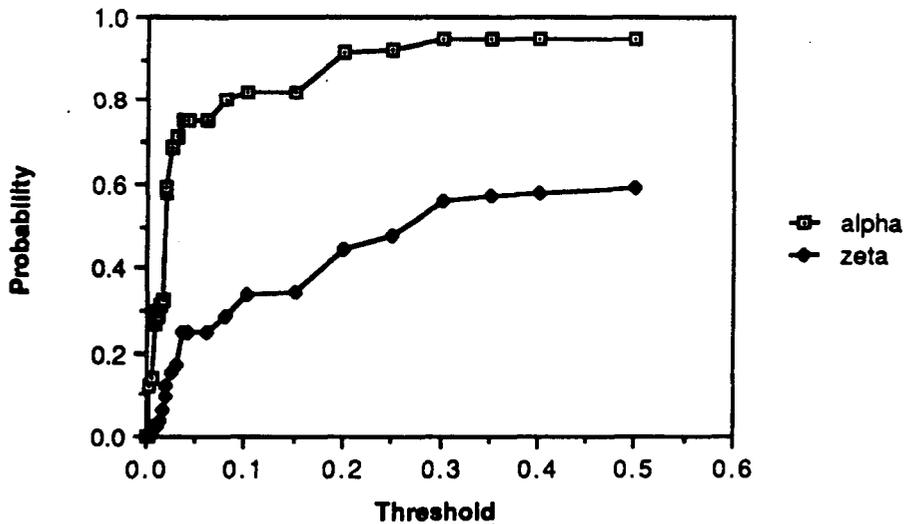

FIGURE 5. EXPERT SYSTEM PERFORMANCE PARAMETERS A AND Z.

As one would expect, the results show that the probability of correcting false alarms increases with the size of the threshold **Thr**. However, as indicated in Figure 5, correcting more false alarms comes at the expense of incorrectly overruling the FDI algorithm when failures have occurred. Nevertheless over the complete range of operation, the probability of correcting the false alarms is greater than the probability of incorrectly overruling the FDI algorithm.
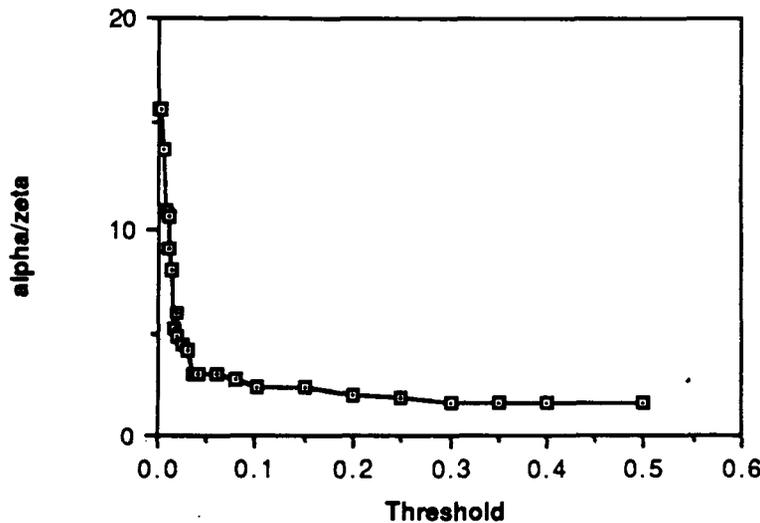
FIGURE 6. THE RATIO OF A TO Z VERSUS THRESHOLD.

This result is encouraging in that it suggests that the expert system is helping things more than hurting them. If we look at the ratio of α to ζ as shown in Figure 6, it can be seen that the expert system provides the largest values of this ratio at the lowest values of the threshold. However, true insight into the effectiveness of the expert system can only be ascertained in the context of the overall system reliability evaluation.

The evaluation methodology was used to predict the values of α and ζ over the range of thresholds as shown in Figure 6. To evaluate the accuracy of these results, we performed a Monte Carlo simulation to generate values for α and ζ. The differences between the predicted and simulated values are shown in Figure 7, indicating that our analytical predictions are correct.
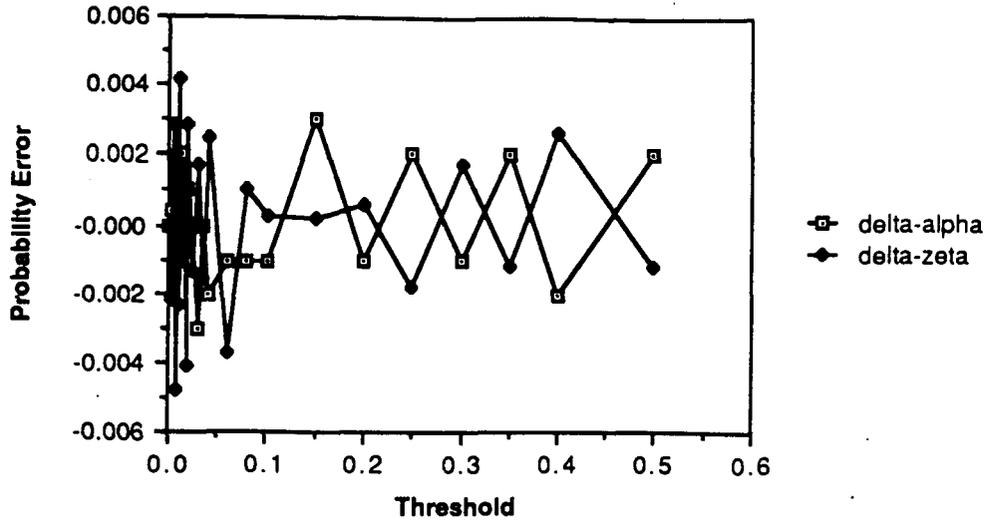
FIGURE 7. A COMPARISON OF THE EXPERT SYSTEM EVALUATION WITH SIMULATION.

5.3 Flight Control System Reliability Results.

The system reliability evaluation is performed using the performance parameters of the expert system (α and ζ) along with the performance parameters for the flight control surfaces (λ) and the FDI algorithm (μ and β). The reliability of the flight control system is evaluated for two levels of FDI coverage ($\beta = 1$ and $\beta = 0.98$) both with and without the expert system. The values for the other system parameters are the same as the example presented in Section 3, $\lambda = 1.0e\text{-}5$ failures/hour and $\mu = 1.7248e\text{-}4$ false alarms/hour. The time step used in solving the Markov model is $\Delta t = 0.0167$ hours. The reliability is presented for an operational period of 1 hour. These results are presented over the range of thresholds that were used by the expert system as shown in Figure 8.

There are two major conclusions that can be drawn from these results. The first and perhaps most obvious is that the simple expert system developed for this example has a significant *negative* impact on the reliability of the flight control system. For small thresholds the impact of the expert system on system performance is less adverse than for large thresholds, but regardless of the the choice of the threshold, the expert system does not provide any measure of increased reliability for the aircraft longitudinal flight-control system. This observation holds true for both modes of operation for the FDI algorithm ($\beta = 1$ and $\beta = 0.98$).
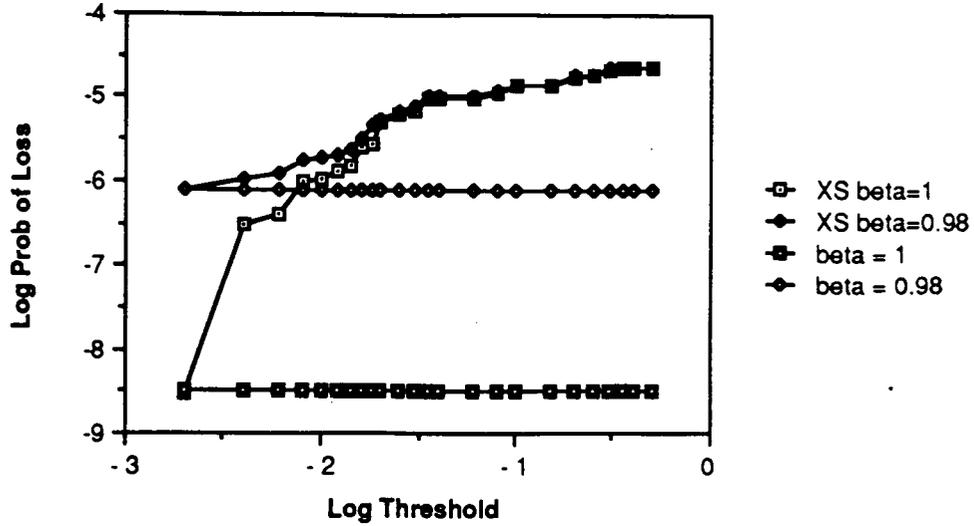
FIGURE 8. SYSTEM RELIABILITY WITH AND WITHOUT THE EXPERT SYSTEM

The second conclusion that must be emphasized is that the objective of this effort was to *evaluate* the performance of the expert system. The fact that the expert system that was developed to be evaluated turned out to be a poor design is irrelevant. Indeed we would expect that an expert system that monitors the performance of a complicated flight-control redundancy management system using only 16 rules, 1 man month of design effort and questionable expertise is bound to perform poorly. The important observation to be made here is that poor performance can be identified early in the design process using such analytical evaluation techniques.

6 SUMMARY

6.1 Conclusions.

A method for evaluating the reliability and performance of systems containing embedded rule-based expert systems has been developed. It is a three stage technique. In the first stage, a Markov reliability model of the system is developed which identifies the key performance parameters of the expert system. In the second stage, the expert system evaluation method is used to assign values to the performance parameters of the expert system. The performance parameters can be evaluated directly using a probabilistic model of the uncertainty knowledge base or by performing sensitivity analyses. In the third and final stage, the performance parameters of the expert system are combined with performance parameters for other system components and subsystems to evaluate the reliability and performance of the complete system. The quantitative results that are produced by this evaluation process can be used in the design process of both the expert system and the system of which the expert system is embedded. In most situations it will be beneficial to use an iterative design process by which the expert system is modified in response to its computed impact on the complete system's performance. An iterative design process will ultimately lead to both greater effectiveness of and greater confidence in real-time expert systems.

An application of the evaluation method has been presented for the case of a simple expert system used to supervise the performance of an FDI algorithm associated with an aircraft longitudinal flight-control system. Using the evaluation method it has been shown that the proposed expert system has a negative impact on the overall system reliability even though it was observed that the expert system did a good job of identifying false alarms. This result underlines the importance of examining the effectiveness of expert systems not just in terms of their individual performance, but in terms of their impact on system performance as well.

6.2 Recommendations for Further Research.

Of the three stages of the evaluation methodology (system modeling, expert System evaluation, and system evaluation) the second stage - expert system evaluation - is the area where there is the greatest lack of experience and where there is the greatest need for future work. The evaluation method that has been proposed assumes that a reasonably good model of the expert system's operating environment is available. In many situations this may be a good assumption. However, further research into the impact of poor environment models on the accuracy of the evaluation method is needed. In the evaluation method, errors of compound assertions were approximated as linear in Δ error terms. However, the errors resulting from these approximations should be small. Again, however,the effect of this approximation on the evaluation method needs to be investigated in more detail. A final area of investigation that should be addressed is the computational problems that may be encountered when the evaluation technique is applied to expert systems containing large knowledge bases. Some of the computations described in this report may become difficult when the number of rules and assertions that are manipulated by the expert system become large. It is important to verify that the evaluation method will scale up to address realistically sized expert systems.

REFERENCES

[1] Buchanan, B.G., and T.M. Mitchell. 1978. *DENDRAL and Meta-DENDRAL: Their applications and dimensions*. Artificial Intelligence 11:5-24.

[2] Martin, W.A., and R.J. Fateman. 1971. *The MACSYMA system*. In Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation, Los Angeles, pp. 59-75.

[3] McDermott, J. 1980. *R1: An expert in the computer systems domain.*, AAAI vol 1, pp. 269-271.

[4] Miller, R.A., H.E. Pople, and J.D. Myers. 1982. *INTERNIST-1, an experimental computer-based diagnostic consultant for general internal medicine.* New England Journal of Medicine, vol. 19., pp. 468-476.

[5] Weiss, S.M., C.A. Kulikowski, and A. Safir. 1977. *A model-based consultation system for the long- term management of glaucoma*, IJCAI vol. 5, pp. 826-832.

[6] Moses, J. 1967. Symbolic integration. Ph.D. Dissertation Rept. MAC-TR-47, Department of Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

[7] Shortliffe, E.H. 1976. *Computer-based medical consultation: MYCIN*, New York: American Elsevier.

[8] Brown, J.S., R.R. Burton, and A.G. Bell. 1974. *SOPHIE: A sophisticated instructional environment for teaching electronic troubleshooting (an example of AI in CAI)*, BBN Rept. 2790. Cambridge, MA.

[9] Duda, R.O., P.E. Hart, P. Barrett, J.G. Gaschnig, K. Konolige, R. Reboh, and J. Slocum. 1979. *Development of the PROSPECTOR consultation system for mineral exploration*, Final Rept. SRI, Projects 5821 and 6415, Artificial Intelligence Center, SRI International, Menlo Park, Calif.

[10] Air Force Systems and Command. *The Pilot's Associate Program Executive Summary*. Dept. of the Air Force, Air Force Systems and Command. Wright-Patterson Air Force Base, Ohio, Jan 31 1985.

[11] Cohen,A., *NASA's Requirements and Expectations of Automation and Robotics for Space Station*, AIAA/NASA Symposium on Automation, Robotics and Advanced Computing for the National Space Program, September 1985, Washington, D.C.

[12] SADP Session, 3rd Annual Conference on AI for Space Applications, Huntsville, Alabama, Nov. 2-3, 1987.

[13] Obermeier, K.K., *Expert Systems - Enhancement of Productivity?*, Productivity in the Information Age: Proceedings of the 46th ASIS Annual Meeting, Knowledge Industry Publications, White Plains, New York, 1983.

[14] *Evaluation of real-time expert systems: An application to the Pilot's Associate*, Charles Stark Draper Laboratory Rept. CSDL-C-5783, 1985.

[15] Hayes-Roth, F., et al., (editors), *Building Expert Systems*, Addison-Wesley Publishing Company Inc. Reading, MA, 1984.

[16] Buchanan, B.G. and E.H. Shortliffe. *Rule Based Expert Systems*, Addison-Wesley Publishing Company Inc. Reading MA, 1984.

[17] Gaschnig J., *Preliminary Performance Analysis of the Prospector Consultant System for Mineral Exploration*, IJCAI, vol. 6, 1979.

[18] McDermott J., *R1: The Formative Years*, The AI Magazine, vol. 2, No. 2, pp. 21-29.

[19] Miller, F.D., et al., *ACE (Automated Cable Expert) Experiment: Initial Evaluation of an Expert System for Preventative Maintenance*, Artificial Intelligence in Maintenance: Proceedings of the Joint Services Workshop, Boulder, Colorado, 1983, pp. 421-428.

[20] Richardson, Kieth, *Overview and Meeting Summary, Workshop on Verification and Validation of Knowledge Based Systems*, AOG/AAIC 1987 Joint Conference on Readiness 2000: Expert System Impact, Oct. 5-9, 1987, Dayton Ohio.

[21] Pages, Alain and Michel Gondran, *System Reliability*, North Oxford· Academic Publishers, London, 1986.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA CR-181769	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Reliability and Performance Evaluation of Systems Containing Embedded Rule-Based Expert Systems		5. Report Date February 1989
		6. Performing Organization Code
7. Author(s) Robert M. Beaton, Milton B. Adams, and James V. A. Harrison		8. Performing Organization Report No.
		10. Work Unit No. 505-66-21-02
9. Performing Organization Name and Address The Charles Stark Draper Laboratory, Inc. 555 Technology Square Cambridge, MA 02139		11. Contract or Grant No. NAS9-17560
		13. Type of Report and Period Covered Contractor Report
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225		14. Sponsoring Agency Code

15. Supplementary Notes
Langley Technical Monitor: Sally C. Johnson Final Report for Task 87-51 prepared for Langley Research Center under Lyndon B. Johnson Space Center contract NAS9-17560.

16. Abstract

A method for evaluating the reliability of real-time systems containing embedded rule-based expert systems is proposed and investigated. It is a three stage technique that addresses the impact of knowledge-base uncertainties on the performance of expert systems. In the first stage, a Markov reliability model of the system is developed which identifies the key performance parameters of the expert system. In the second stage, the evaluation method is used to determine the values of the expert system's key performance parameters. The performance parameters can be evaluated directly by using a probabilistic model of uncertainties in the knowledge-base or by using sensitivity analyses. In the third and final stage, the performance parameters of the expert system are combined with performance parameters for other system components and subsystems to evaluate the reliability and performance of the complete system. The evaluation method is demonstrated in the context of a simple expert system used to supervise the performances of an FDI algorithm associated with an aircraft longitudinal flight-control system.

17. Key Words (Suggested by Author(s)) Expert Systems; System Evaluation; Uncertain Knowledge-Base Systems; Markov Models; Sensitivity Analyses	18. Distribution Statement Unclassified - Unlimited Subject Category 62		
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 34	22. Price A03

NASA FORM 1626 OCT 86