A User's View of CARE III

Salvatore J. Bavuso; NASA Langley Research Center; Hampton

Key Words: Fault-tolerant, CARE (Computer Aided Reliability Estimation), Ultrareliable, Estimation, Modeling, Reliability model, Coverage, CARE III

Abstract

A novel, powerful, computerized reliability predictor for highly reliable digital fault-tolerant systems has recently been developed and will soon be released. CARE III (Computer-Aided Reliability Estimation) was designed to model very large systems on the order of 10^6 Markovian equivalent states. Through the use of advanced stochastic modeling techniques, CARE III implements a mixed Markov model that enables it to drastically reduce the state size of hitherto computationally unreachable models that are of practical interest.

This paper introduces the basic concepts of CARE III from a user's point of view. After describing the major attributes of the reliability evaluator, a discussion of the applicable class of fault-tolerant, digital-based computer architectures is presented. Following this discussion, the notions of failure, fault, and error are presented in the context of CARE III's fault/error-handling models. The paper concludes with a description of CARE III's user-friendly interface and an example dialog portraying the assessment of a highly reliable fault-tolerant system. Also, some mention is made of the extensive testing and verification of the CARE III stochastic model and computer program.

Introduction

Digital fault-tolerant computer-based systems are on the verge of becoming commonplace in military and commercial avionics. These systems hold the promise of increased availability, reliability, and maintainability over conventional analog-based systems through the application of replicated digital computers arranged in fault-tolerant configurations. Three tightly coupled factors of paramount importance which ultimately determine the viability of these systems are reliability, safety, and profitability. Reliability, the major driver, involves virtually every aspect of design, packaging, and field operations with regard to safety and maintainability, and translates invariably to profit for commercial applications or national security for military uses.

The antithesis of promise for the digital computer is, however, the Achilles' heel of the reliability engineer. The utilization of digital computer systems makes the task of producing a credible reliability assessment a formidable one. The root of the problem is embodied in the very essence that makes the digital computer such an outstanding device for a host of applications; namely, its adaptability to changing requirements, its computational power, and its ability to test itself efficiently.

The development of a novel methodology for the reliability assessment of fault-tolerant digital computer-based systems, Computer-Aided Reliability Estimation III (CARE III), is reported in this paper. The assessment technology was developed to mitigate a serious weakness in the design and evaluation process of ultrareliable digital systems. The weak link is the lack of a sufficiently powerful modeling technique for comparing the stochastic attributes of one system against others.

A long-term goal of the NASA Langley Research Center is the development of this methodology. The technology development process is shown in figure 1. The quest for achieving our goal began about 1973. State-of-the-art reliability evaluators were typical of CARE, a Jet Propulsion Laboratory computer program, and TASRA (Tabular System Reliability Analysis), developed by Battelle Memorial Laboratories, Columbus, Ohio. The Raytheon Company at Sudbury, Massachusetts, and Langley developed the CARE II, which provided a superset CARE model with an extensive fault-handling model. Langley was also involved in the development of CAST (Combined Analytic Simulative Technique), which provides the current Langley modeling concept. CAST was developed by Ultra Systems, Inc., of Newport Beach, California, and CARsRA (Computer-Aided Redundant System Reliability Analysis) is a spin-off from the Boeing ARCS (Advanced Reconfigurable Computer System) study. (See ref. 1.) Langley has also been involved in numerous technology development studies, some of which are depicted in figure 1. This long-term involvement has culminated in the development of the CARE III (ref. 1). CARE III was codeveloped by NASA Langley and the Raytheon Company, and the work was cosponsored by NASA and the U.S. Air Force Avionics Laboratory of AFWAL (refs. 2 to 5).

Figure 1.- Technological development leading to CARE III. (From ref. 1.)

CARE III Overview

CARE III is a novel system reliability predictor applicable to very large, highly reliable fault-tolerant systems that utilize digital computers and replicated hardware to achieve fault tolerance and increased system reliability. Figure 2 depicts two applicable systems (refs. 6 and 7). Its main application was directed toward flight-critical aircraft systems; however, the CARE III stochastic model is robust enough to be applicable to a wide range of highly reliable systems.

To date, CARE III has received considerable scrutiny at the detailed mathematical modeling level as well as at the application level. Extensive testing has been done to bring CARE III to an operational level for the CDC CYBER 170 series computers and the

U.S. Government work not protected by U.S. copyright.

Figure 2.- Highly reliable fault-tolerant systems to which CARE III is applicable. (From ref. 1.)

VAX 11/700 series systems. The CDC code is written entirely in Fortran IV and has plotting capability for DISSPLA and Plot 10 supported systems. The VAX version is written in VAX-11 Fortran under VMS and has a user-friendly prompting front end. The most common CARE III output is in tabular form and delineates the probabilities of user-defined system configurations achieving system/function success or failure at user-specified mission times.

The outstanding attributes of CARE III are as follows.

- CARE III has the ability to model very large, highly reliable systems on the order of millions of states in the Markov model sense. This capability was achieved by computationally exploiting the wide separation of time constants in the fault/error-handling model (on the order of 10^{-6} hour) and the failure occurrence model (on the order of 10^4 hours).

- CARE III has a powerful and flexible fault/error-handling model that accounts for fault and error latency, error propagation, and fault and error detection. The fault/error-handling model spans the spectrum from single-point failures to double critically coupled failures.

- CARE III has failure occurrence models that can be exponentially or Weibull distributed which cover permanent, transient, intermittent, and software failures. The Weibull option permits the proper treatment of wearout failures common to mechanical, hydraulic, and some electronic devices. The exponential model is applicable to a wide range of electronic device failures.

- CARE III uses a flexible, user-defined, system success/failure configuration language based on the fault tree notation. The user-defined tree logic enables the description of large, complex, system failure configurations to define mission failure/success, system or function abort, and functional readiness

probabilities. The configuration can be the common series-parallel arrangement or the nonseries-parallel arrangement and may include functional as well as module dependencies.

- CARE III incorporates computational decoupling of the fault/error-handling and failure occurrence models. This capability is a direct result of the first listed attribute. It allows the user to exercise the fault/error-handling model irrespective of the failure occurrence model and can be very useful for system fault-handling analyses.

The clever mathematical techniques that provide CARE III with these outstanding attributes are well documented and will not be addressed here. A considerable amount of independent effort has been expended to verify CARE III (ref. 8).

Failure, Fault, Error View

CARE III treats failures, faults, and errors in a precise way. The word failure is used to mean loss of function; thus systems fail, piece-parts and software fail. Errors are undesirable deviations of processed information and are manifestations of software failures. Software failure has a different interpretation than random hardware failure. Software, therefore, does not have failure mechanisms like hardware, but software can cease to function properly and hence can fail. The cause of the software failure is, of course, a design defect; however, its manifestation, an error, appears to occur stochastically much like a random hardware failure. There is another important difference between hardware failures and software errors: random hardware failures are typically assumed to be independent of the software being executed, while software error manifestation appears to be dependent on the code that is being executed. A fault is the manifestation of a hardware failure at the piece-parts level, and may take the form of a stuck-at (e.g., stuck at logic 1 or logic 0). A fault may or may not produce an error (an undesirable deviation in information). Like a software error, a fault producing an error is dependent on the code that is being executed.

CARE III captures the notions of failure, fault, and error in two types of models. Failures are treated by the failure occurrence model (FOM), which is similar

Figure 3.- Delineation of hardware and software failure and error models. (From ref. 1.)

to the traditional piece-parts reliability model. The most general FOM in CARE III uses the Weibull failure distribution, and an important subset of the Weibull is the common exponential failure distribution. For hardware failures, the FOM is treated in the conventional sense as random time to failure. For software, the FOM must be viewed as time-to-failure manifestation rather than the time from the creation of the design defect. In conjunction with the FOM, CARE III offers an extensive fault/error-handling model (FEHM). The FEHM accounts for the system's reaction, if any, to the occurrence of failures and errors. The types of models of interest are presented in figure 3. Each tree branch represents a possible failure and/or error model. The common piece-parts analysis is represented by the permanent random hardware failure branch. The CARE III FEHM was designed with an eye toward accommodating these types of models.

Architectural View

CARE III is composed of two basic models: a failure occurrence model and a fault/error-handling model. Because of the mathematical structure of CARE III, these two models are treated independently; the failure occurrence model can be processed without initial regard to system fault/error handling, and fault/error handling can be processed without regard to failure occurrences. The final results of the reliability assessment integrate the results from the two models to form composite solutions, just as if failure occurrence and fault/error handling had been treated together all along. The details supporting this decomposition-aggregation technique are well documented (ref. 9) and are beyond the scope and purpose of this paper. Suffice it to say that this mathematical technique makes it possible to assess very large systems with hitherto unmanageable computations. Figure 4 depicts the relationship between the FOM, the FEHM, and the CARE III computer program structure. The FOM is described by the fault tree notation, and the FEHM takes the appearance of a directed graph resembling a graph of a Markov model.

Figure 4.- CARE III user process. (From ref. 1.)

Failure Occurrence Model

CARE III views a system as a collection of subsystems called stages. A stage may comprise one or more modules, each of which has an identical Weibull time-to-failure distribution. A stage may be composed of hardware/software modules or may represent a function, such as an aircraft elevator math model, that is implemented by a computer or a replicated group of computers for fault tolerance. The use of dissimilar

redundancy as is described below is in widespread practice in the commercial aviation industry. If a stage has replicated modules configured for fault tolerance, CARE III treats that stage as an M out of N subsystem, where N is the maximum number of modules needed and M is the minimum number for stage survival. N and M are user definable for each system stage. Stages may also be redundant to achieve additional fault tolerance. The relationship between stages is defined by the system failure configuration tree (fig. 5). The user describes this relationship by using the CARE III fault tree language.

Figure 5.- System failure configuration tree.

System failure configurations are described by a system tree. The simplest system configuration is the series stage configuration. In this configuration, if any stage in the system fails, then the entire system fails. The fault tree for this system configuration is an OR gate where the inputs are probabilities of stage failure (fig. 5). Each stage may be fault tolerant. For example, at stage 1, $N = 3, M = 2$; at stage 2, $N = 4, M = 2$; and at stage 3, $N = 2, M = 1$. To get system failure, at least one stage must have more than $N - M$ failures. When $M = N = 1$, stages are treated as modules, and module dependencies can be defined. A simple example of redundancy across stages (modules for $M = N = 1$) is to substitute an AND gate for the OR gate in figure 5. Thus all stages must fail before the system fails; this is parallel redundancy. CARE III currently supports 70 stages with a maximum value of 70 for N and $M > 0$. To describe system trees, CARE III supports AND, OR, M out of N, and input gates, as well as up to 2000 total events and 70 data input events. Input events are the lowest level gate inputs; the other events are higher level gate inputs and outputs.

Another example of the flexibility of the fault tree language is depicted in figures 6 and 7. Figure 6 shows a block diagram of a proposed fault-tolerant flight control system. Of particular interest is the pitch augmentation stability (PAS) short-cycle function. The system fault tree for this function is presented in figure 7. It is composed of eleven stages, each of which comprises only one module; thus $N = M = 1$ for all stages. The stages are C-A, C-B, C-C, C-D (computers A, B, C, D); SIR-A, SIR-B, SIR-C (inertial reference sensors A, B, C); SPRD (pitch rate detector sensor); and AE-A, AE-B, AE-C (actuator electronics A, B, C). A more complex version of this tree would allow N to be greater than M (for any given stage), thereby including redundancy not only across stages, as is the case above ($N = M = 1$), but also within stages. This flexibility gives CARE III a very powerful capability to define system failure configurations. The tree in figure 7 illustrates that not only hardware redundancy but also functional redundancy can be represented. The

Figure 6.- Fault-tolerant flight control system for pitch augmentation stability function (PAS). (From ref. 1.)

Figure 7.- CARE III fault tree input for PAS function. (From ref. 1.)

elevator math model is functionally redundant to the secondary actuators. The melding of hardware and functional redundancy is a common practice in aircraft design. The proper entry of this fault tree into CARE III with the necessary failure rate and fault-handling data would yield a prediction of the probability of loss of PAS function as a function of mission time. Figure 7 is read as follows: An output from logic OR gate 212 constitutes loss of PAS function, which can occur if an output from OR gate 211 occurs, or if an output from gate 210 occurs, or both. Gate 210 yields an output if at least 3 out of 4 secondary actuators, including actuator function (elevator math model), fail. Secondary actuator A will fail if computer A fails, or actuator A fails, or both. A similar description can be used to delineate failures due to loss of computation or loss of sensors.

Fault/Error-Handling Model

Thus far no mention has been made of the effects of system fault/error handling on system failure. As mentioned earlier, a stage failure is defined by the M out of N model. A stage may also fail due to system fault/error-handling problems; however, the system may or may not fail as a result, depending on the redundancy described by the system tree. There may also be cases in which the system may fail even though there is sufficient hardware redundancy remaining. This situa-

tion arises when the system is susceptible to the occurrence of certain critically coupled failures. Such known critically coupled failures may be permitted to remain in a fault-tolerant system provided that their joint occurrence is less probable than the desired mission failure probability. The paths to system failure are threefold: the one shown in figure 5 (exhaustion of hardware), the single-point fault-handling paths, and the critical-pair failure path yet to be described. The latter two failure paths are also user definable. They may not exist in certain system models.

Single-fault model.—The FEHM is comprised of a single-fault model and a double-fault model. The parameters for the single-fault model are user specified, whereas the parameters of the double-fault model are determined from the single-fault model and therefore require no user specification. The user is optionally required, however, to identify system-critical pairs of failures. (This will be discussed later.) The complete single-fault model is depicted in the dashed box in figure 8. Three additional system

Figure 8.- Overall reliability model of two-unit system. (From ref. 1.)

states (0, 1, and F) have been added for illustrative purposes, so that the state diagram is a mixed Markov model of a two-unit system. Initially, the system is in state 0 and has experienced no failures. When a failure occurs, the system enters state A, the latent state, given by the Weibull arrival density $\lambda(t)$. Depending upon the nature of the failure (i.e., permanent, transient, intermittent, etc.), the FEHM will be defined differently. For example, if the failure is intermittent, $\lambda(t)$ would be the probability density function (pdf) for the arrival of an intermittent, and states A and B would define the intermittent model where α and β are constant transition rates into and out of state B. When the system is in state B, the benign state, the failed unit appears to have healed itself (i.e., the manifestation of the failure, a fault, vanishes). However, when the failed manifestation is once again resumed (i.e., the fault reappears), the system enters state A, where the failure looks like a permanent failure. It could be detected by a self-test program with pdf (exponential or uniform) $\delta(t')$, and the system would enter state A_D, the active detected state. Given that a spare exists, the system will purge the faulty unit and switch in the spare (dashed arc to state 1). Or, while in the active state, the fault could generate errors with pdf (exponential or uniform) $\rho(t')$. The system will then enter state A_E, active error state. The intermittent failure could manifest its intermittent state again, in which case the system would enter state B_E, the benign error state. Although the failure is benign, the error may

not be benign and may cause system failure, which is denoted by the B_E to F transition $(1-c)\epsilon(\tau)$. The error detection pdf (exponential or uniform) is $\epsilon(\tau)$, and $1-c$ is the proportion of errors from which the system is unable to recover. While the system is in state B_E , the error can be detected and corrected. In this event, the system enters state B_D (benign detected) by the transition $c\epsilon(\tau)$. At this point, the system may choose to do nothing further with the detected and corrected error and so move to the benign state, or the system may choose to reconfigure out the module containing the error and therefore move to state 1. The dashed arcs are instantaneous transitions. The other transition out of state A_E is to state F, the single-point failure transition $(1-c)\epsilon(\tau)$. This transition is similar to the B_E to F transition. In a well-designed fault-tolerant system, $(1-c)\epsilon(\tau)$ should be near zero. If $\lambda(t)$ is the pdf for the arrival of a transient, α would be set to a value greater than zero and β would be equal to zero. The pdf $\lambda(t)$ for the arrival of a permanent failure would necessitate that $\alpha = \beta = 0$. The dashed arc going from state A_D to state A enables the analyst to include the effects of the system decision that the detected fault that took the system from state A to A_D was, in fact, a transient. In this regard, the system would not reconfigure out a nonfailed module. A judicious choice of values for the single-fault model affords the analyst a wide range of models. A different fault model may be assigned to each stage, or several models may be assigned to a given stage to cover the effects of different failure mechanisms such as transients, intermittents, and permanent failures.

The reader will note that the reliability model in figure 8 has three measures of time associated with it, which necessarily makes the model a mixed Markov process. The added complexity is required because the behavior of the system is dependent on the time of entry into state A or A_E and not the total elapsed time to each respective state. This treatment of fault behavior greatly increases the ability of the FEHM model to capture the realism of competing fault/error-handling events. The sophistication of the advanced modeling capability separates the CARE III fault/error-handling treatment from all other existing reliability assessment capabilities. CARE III can process otherwise intractable models solely because of its ability to process very large state space models. Prior to CARE III, the outcome of system fault handling was modeled by constant parameters called coverage. Many popular models (CARSRA, ARIES, TASRA, GRAMPS) assume that transition times to recovery states (e.g., to state 1, fig. 8) are exponentially distributed. CARE III does not assume this and therefore provides a realistic treatment of competing fault-handling events (refs. 10 to 13). Other models that approximate non-exponential distributions by using Erlangian distributions (method of stages) are plagued with an explosion in the size of the state space.

Double-fault model.—With the inclusion of a double-fault model, CARE III captures the notion of critically coupled failures. Highly reliable fault-tolerant systems are commonly designed without single-point failure mechanisms, or, perhaps more realistically, with an extremely low probability of single-point failure occurrence. Often, the dominant system failure cause is a critically coupled double failure. In these systems, most double faults are tolerated by the system; however, certain critical groupings of double faults are often not tolerated by the system and will cause system failure. It is the aim of these system designs to make critically coupled failures less probable than the mission desired unreliability. An illustrative example of a critically coupled failure is the occurrence of two failures, one each in two of three different voting or comparison-monitoring computers performing the same flight-critical control

computations. A double failure in the same computer of the voting triad, however, would not be critically coupled.

The user specification of the critically coupled failures is accomplished by the user defining a critical-pair tree. The critical-pair tree specification utilizes a subset of the fault tree language used to define the system failure configuration tree. The critical-pair tree for a voting triad is simply a 2 out of 3 logic gate. In more practical systems the critical-pair tree can become quite complex, as critical pair failures may occur not only within a stage (such as the voting triad) but also across stages. An example of the latter application would be a critically coupled failure between a CPU (central processing unit) stage and the CPU bus stage. Such a double failure may cause the faulty CPU to capture the CPU bus (bus babbling) when a CPU bus failure also occurs.

The precise nature of the critically coupled failure is defined by the double intermittent fault model (fig. 9) and the single-fault model (fig. 8). In order

Figure 9.— CARE III double intermittent critically coupled fault model. (From ref. 2.)

to keep the model within computational feasibility, CARE III makes the implicit conservative assumption that all critically coupled (user-specified, if any) latent faults will cause system failure. Thus any paired combination of critically coupled failures occupying states A, A_E , or B_E in the single-fault model (see fig. 8) will constitute system failure irrespective of whether or not the system still has sufficient operational units left to meet the M out of N stage requirements. In addition, CARE III supports a double intermittent critically coupled fault model, which is depicted in figure 9. In the event the user defines the single-fault model to represent the system behavior resulting from the occurrence of one or more intermittent failures, the double intermittent fault model will be invoked. On the occurrence of the second failure, the first having been an intermittent failure, state A_2B_1 will be entered. If A_2 also becomes intermittent, then state B_1B_2 may be entered. Alternately, if while in states A_1B_2 or A_2B_1 the active fault is detected with pdf $\delta_1(t)$ or $\delta_2(t)$, then the fault is determined to be permanent and the system enters state D, the detected state. At this point, CARE III will reconfigure out the faulty detected module if a spare is available. The last possibility is a transition to state F, the system failure state. That transition can occur if the benign intermittent (B_1 or B_2) becomes active (giving two coexisting critically coupled latent faults), if the active fault begins to propagate errors ($\rho_1(t)$ or $\rho_2(t)$), or if a fault is detected as nonper-

manent $((1-P_{A1})\delta_1(t) \text{ or } (1-P_{A2})\delta_2(t))$. P_{A1} is the probability that the i th fault, $i = 1, 2$, is detected as permanent. More detailed models could have been implemented (they were in fact seriously considered), but only at the expense of creating greater computational cost with little justification to back up the need. The present FEHM's represent a compromise between a conservative model with reasonable detail and computational cost.

The authors of CARE III feel that the structure of the FEHM is sufficiently robust to enable the modeling of a wide variety of failure/error models (fig. 3). It is recognized that data for many models are nonexistent; however, in many of these cases, the models may still be of value by providing worst case or best engineering results. Methodologies for estimating $\delta(t')$ and $\epsilon(t)$ shown in figure 8 have been studied and documented (refs. 14 and 15). Some work was initiated to predict $\lambda(t)$, α , and β for intermittent failures, but much more work is required. The reader is further alerted to the need to apply ingenuity in the interpretation or reinterpretation of the meaning of the transition parameters associated with the FEHM; after all, these are but stochastic models. The interpretation of the transition parameters, in many cases, is no easy task and will require expertise in at least computer science and computer architectural design. In the event the FEHM proves to be deficient, CARE III has the robustness to accommodate changes.

CARE III Applications - Example

An example of the CARE III assessment process as depicted in figure 4 is given by figure 10. Figure 10 also presents the input data for the CARE III CDC version. The VAX user-friendly input for the same example will be discussed subsequently. Figure 10 depicts a sketch of an ultrareliable fault-tolerant multiprocessor composed of 10 memory-processor pairs which communicate with each other over five individual bus lines (shown in the chart as a solid bus line). This system survives if at least two computers and two buses are operational. The analyst wants to compute the probability of system survival at 10 hours of mission time for this multiprocessor system and the probability of system failure at 10 hours due to spare hardware depletion and improper system-fault handling. For this simple illustrative example, the analyst creates two fault trees and a state diagram for system fault handling.

The SYSTEM FAULT TREE in figure 10 describes the system stage configurations that cause system failure. The computer stage is comprised of 10 computers, each having an identical failure rate. The bus stage is composed of five buses, each having an identical failure rate that is different from that of a computer. The OR gate in the SYSTEM FAULT TREE means that the system fails if a computer stage fails or a bus stage fails. A computer stage fails if less than two computers are operational, and a bus stage fails if less than two buses are operational. These conditions are described in the line beginning with "\$STAGES." This statement is a Fortran NAMELIST statement. It says there are two stages (NSTGES=2) (i.e., 10 computers and 5 buses (N=10,5)), and the minimum for stage survival is two for each stage (M=2,2). The remainder of the line describes the form of output data requested. The fault tree description for CARE III input is shown under the heading SYSTEM FAULT TREE. It describes the gate interconnections and the types of gates. Another cause of system failure which is implicit in the system fault tree is system failure due to single-point failures in either stage. For this example, the default value for C is set to unity (by omission of an assignment), thereby precluding a single-point failure (see fig. 8).

(a) Ultrareliable fault-tolerant multiprocessor.

(b) CARE III input for ultrareliable fault-tolerant multiprocessor.

Figure 10.- Example of CARE III assessment process. (From ref. 1.)

A unique modeling capability of CARE III is the incorporation of the effects of synergistic pairs of failures. In fault-tolerant systems, the system could contain many undetected (latent) failures which individually would not cause system failure; however, certain groupings of failures which coexist may bring the system down. The CRITICAL-PAIR TREE (fig. 10) enables the analyst to specify the conditions under which synergistic paired failures cause system failure. For this case, any two latent computer failures out of 10 computers or any two latent bus failures out of five buses cause system failure. This failure assignment is very conservative and was chosen to keep the example simple and illustrative. The CRITICAL-PAIR TREE is described by the data listed under the heading CRITICAL-FAULT PAIRS (fig. 10).

The next step in the CARE III input process is the description of the FAULT-HANDLING MODEL (fig. 10). This simple state model is composed of two system states, active (A) and active detected (AD). The active system state is entered when a failure occurs. It is an undetected or latent state. If a fault detector is employed, δ is the rate at which failures are purged from the system. CARE III assumes that if the system enters the active detected state and it has spare hardware, it will reconfigure out the faulty module and the system will recover. Note that as

δ increases, the probability of synergistic failures occurring diminishes since there will be fewer latent failures present. In figure 10, line 1 (\$FLTTP) shows that there is one fault model (NFTYPS=1) and defines the value of δ as 3.6×10^2 detections per hour. Line 5 (\$RNTIME) specifies flight time of 10 hours.

CARE III input data for this example system begins with the statement \$FLTTP and includes all the statements that follow in figure 10. The CARE III output includes the total system probability of failure, the system probability of failure due to improper fault handling, and the probability of system failure due to spare exhaustion. For example, figure 11 depicts the plotted data for the total system failure probability. Upon request, CARE III will also plot fault-handling information such as the probability that a fault was diagnosed as permanent by time t' or the probability of a latent fault or undetected error at time t' , where t' is a measure of the time since the entry into state A (fig. 8).

Figure 11.- CARE III output data for example system.

User-Friendly Interface

The VAX user-friendly menu-driven interface (UFI) was designed to provide the user with a communication mechanism that follows and prompts the user thought processes for performing a reliability assessment. The UFI also does some processing of input data to relieve the user burden of linking failure occurrence and fault/error-handling data for system stages. The UFI offers a brief tutorial on CARE III, provides user assistance through the availability of help files associated with the various menus, checks for out-of-range input values, and provides a flexible editor for making changes or corrections. The conclusion of a session with the UFI results in the creation of an input file for CARE III which is equivalent to the CDC input format. The file may be operated on by either the VAX CARE III or the CDC CARE III code.

Because of the breadth of the CARE III capability, it is not recommended that a novice user attempt to use the UFI for serious assessments. A thorough understanding of the CARE III capability should first be obtained from the user's manual and other CARE III documents. The UFI was not designed to be a learning tool; rather it is an aid to enhance the production of reliable data entry. Most users would probably formulate some sort of sketch of the system's primary elements, similar to the fault tree diagrams and FEHM in figure 10, before addressing the UFI. The UFI conversation is delineated into the following elements:

- Stage Descriptions
- Fault-Handling Models
- Fault Occurrence Models
- Fault Tree Descriptions
- Output Control/Format Selections

Within each element, the UFI will prompt the user for the required data. For most of the entries, preassigned default values exist, so that in many cases the user can simply choose default values by depressing the return key. After supplying the UFI with a name for the system, the following screen menu is displayed:

Stage Description Input

Stage name:
Number of beginning modules in stage:
Minimum number of modules for stage operation:
Set(s) of modules subject to critical pair failures:
Critical fault threshold:

If at any time the user requires assistance, a question mark is entered in lieu of a data item. The UFI will then provide a short tutorial on the subject. The user can then return to the screen menu to continue. When the data are entered for this stage, the UFI repeats the screen menu to enable multiple stage entries. To end stage input, the user keys in "end" in response to a stage name request. The UFI will then display the Fault-Handling Models screen menu. The purpose of this menu is to provide data for the model enclosed in dashes in figure 8. The screen menu appears thus:

Fault-Handling Models

Fault type:
Alpha = (Exponential FOM)
Beta = (Exponential FOM)
Delta = (Uniform/Exponential)
Rho = (Uniform/Exponential)
Epsilon = (Uniform/Exponential)
Pa =
Pb =
C =

The user responds by typing "permanent" for the fault type. Since the fault is permanent, the user would set alpha equal to zero. Exponential FOM is shown to remind the user of the failure occurrence model (FOM) associated with alpha. The default value is zero, so the user would press the return key. When alpha = 0, the value for beta is a "don't care," so the default value is chosen. Delta is assigned the value of $3.6E+2$ detections per hour, to be consistent with the example in figure 10. The user has the choice of assigning a uniform or an exponential distribution for delta. The default, uniform distribution, is selected. The default values for all the remaining parameters are selected by pressing the return key seven more times. If a mistake is made while a value is being entered, the user merely presses the delete key as required and reenters the data. On the other hand, if after completing a screen menu the user desires to alter an entry, he requests "no verify" when the UFI prompts a "verify yes/no" at the end of a completed screen menu. A "no verify" will return the user to the top of the current screen. All previous values are remembered, so the user presses the return key until he arrives at the value to be altered. After making the necessary changes, further return key presses return the menu to the "verify yes/no" request. A "yes verify" advances the screen to the next screen menu, in this case to the Fault Occurrence Model screen menu. The dialog is similar to previously described screen menus. In this screen, the user specifies failure distributions and parameters for the previously defined stages. Weibull

and exponential distributions are the available choices.

The Fault Tree Description is implemented with scrolling screens in lieu of screen menus to allow the user to see the previously typed data for some time. The UFI prompts the user by requesting the type of data required, such as "Enter System Fault Tree label" and "Enter Input Event ID Range, Output Gate ID Range." This fault tree format is similar to the CDC version; however, the UFI offers some help in organizing and prompting the data.

The Output Control/Format selection queries the user to select output information such as coverage moment function data, reliability print or plot data, coverage function plots, and plot axis selections. Mission time for the assessment is selected here, and the numerical integration routines are given such information as truncation value and number of integration steps. At this point in the dialog, the user has completed the data entry task and may list the CARE III input (which is equivalent to the input data for the example of fig. 10), or CARE III may be instructed to operate on this input file. The UFI file data lists all the parameter values, including the selected defaults. This file may be further manipulated prior to execution in either a VAX or CDC computer. Because of the potentially large amount of data required for all but the most trivial systems, the UFI has proven to be an indispensable user tool.

Concluding Remarks

Ultrareliable fault-tolerant systems are rapidly gaining in sophistication and complexity and have already surpassed the point at which it would be practical to life test these systems to estimate their reliability. As many of these systems are designed for high-risk applications, including potential loss of life, credibility of design is of paramount importance. Credible reliability assessment techniques must be realized before ultrareliable systems can attain their full application potential. The CARE III capability, which was developed from a long legacy of reliability technology, makes a major contribution toward fulfilling the need for a powerful, credible, but practical reliable assessment methodology.

In the short time that CARE III has been analyzed and exercised by the developers, numerous enhancements have been identified, including a redesign of the FEHM and of some numerical integration schemes. The advanced version of CARE III which is under development is called HARP (Hybrid Automated Reliability Predictor) (ref. 16). An outstanding modeling contribution made by HARP is the incorporation of a Petri-net-based FEHM that is solved by simulation in lieu of the analytic CARE III FEHM. The development of HARP and other reliability assessment technologies is a continuing process at the Langley Research Center, as it should be to meet the assessment demands of new fault-tolerant architectures and parts technologies.

References

1. Bavuso, S. J.: Advanced Reliability Modeling of Fault-Tolerant Computer-Based Systems. Electronic Systems Effectiveness and Life Cycle Costing, J. K. Skwirzynski, ed., Springer-Verlag, 1983, pp. 279-302.
2. Stiffler, J. J., et al.: CARE III Final Report Phase I, Vol. 1. NASA CR-159122, 1979.
3. Stiffler, J. J., et al.: CARE III Final Report Phase I, Vol. 2. NASA CR-159123, 1979.
4. Bavuso, S. J.: Trends in Reliability Modeling Technology for Fault Tolerant Systems. Avionics Reliability, Its Techniques and Related Disciplines, AGARD CP 261, April 1979, pp. 17-1 - 17-12.

5. Stiffler, J. J.; and Bryant, L. A.: CARE III Phase II Report, Mathematical Description. NASA CR-3566, 1982.

6. Hopkins, A. L., et al.: FTMP - A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft. Proceedings of the IEEE, vol. 66, no. 10, Oct. 1978, pp. 1221-1239.

7. O'Hern, E. A.: Space Shuttle Avionics Redundancy Management. AIAA Paper No. 75-571, 1975.

8. Rose, D. M., et al.: Review and Verification of CARE III Mathematical Model and Code: Interim Report. NASA CR-166096, 1983.

9. Trivedi, K. S.; and Geist, R. M.: A Tutorial on the CARE III Approach to Reliability Modeling. NASA CR-3488, 1981.

10. Bjurman, B. E., et al.: Airborne Advanced Reconfigurable Computer System (ARCS). NASA CR-145024, 1976.

11. Ng, Y.: Reliability Modeling and Analysis for Fault-Tolerant Computers. NSF-MCS-7203633 (UCLA-EWG-7698), September 1976.

12. Blazek, R. H., et al.: Demonstration of Combined Reliability Prediction and Verification Techniques to a Typical Flight Control System, Vol. I, Development and Application of Tabular System Reliability Analysis to the F-111 Pitch Flight Control System. AFFDL-TR-71-128, Vol. I., Oct. 1971. (Available from AF Flight Dynamics Lab., Wright-Patterson AFB.)

13. Dolny, L. J., et al.: Fault-Tolerant Computer System Design Using GRAMP. 1983 Proceedings Annual Reliability and Maintainability Symposium, Jan. 1983, pp. 417-422.

14. Bavuso, S. J., et al.: Latent Fault Modeling and Measurement Methodology for Application to Digital Flight Controls. Advanced Flight Control Symposium, USAF Academy, Colorado Springs, CO, Aug. 1981.

15. McGough, J. G., et al.: Measurement of Fault Latency in a Digital Avionic Miniprocessor Part II. NASA CR-3651, 1983.

16. Geist, R., et al.: Design of the Hybrid Automated Reliability Prediction. 5th Digital Avionics Systems Conference, Seattle, WA, Nov. 1983.

Biography

Salvatore J. Bavuso
NASA Langley Research Center
Hampton, Virginia 23665 USA

Mr. Bavuso is a senior researcher at NASA Langley Research Center in Hampton, Virginia. He received the B.S. degree in mathematics from the Florida State University in 1964 and the M.S. degree in applied mathematics from the North Carolina State University at Raleigh in 1971. He has been instrumental in the development of advanced reliability modeling technology for over a decade.