NASA Contractor Report 178380

ON THE NEXT GENERATION OF RELIABILITY ANALYSIS TOOLS

PHILIP S. BABCOCK IV,
FRANK LEONG,
and ELI GAI

THE CHARLES STARK DRAPER LABORATORY, INC.
CAMBRIDGE, MA

Contract NAS1-18061
OCTOBER 1987

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

1. INTRODUCTION

The need for highly-reliable systems is increasing. Areas of application cover civilian aircraft, military systems, nuclear power plants, and spacecraft, to name a few. The very nature of these systems, their high reliability, makes them challenging to design and analyze. In particular, how does one go about demonstrating that a system that is designed to fail once in ten years does, in fact, meet this design goal? Highly-reliable systems are most often constructed from a collection of moderately reliable components that are used in a way that promotes fault tolerance and high reliability. This approach is not only economical, it provides for a graceful degradation into backup modes that is often necessary for safe operation of the system.

Life testing to establish the reliability of a highly-reliable system is not practical. The time to failure is too long and the system costs are too high to permit the accumulation of a statistically significant sample. Instead, system reliability is predicted using a mixture of component testing and system modeling. The results of the component testing, failure rates and coverage, are used as inputs to the system model. This model predicts the system reliability by reflecting the interactions between the components.

There are three candidate techniques for analytically predicting a fault-tolerant system's reliability. First, a simulation technique such as a Monte Carlo approach could be used [1, 2]. The high reliability of the system necessitates a prohibitively large number of simulation runs to obtain a statistically significant solution. Second, a combinatorial technique such as a fault tree approach could be used [3, 4]. These techniques are highly efficient computationally. However, they present three problems: there is no simple way to deal with the sequence dependencies inherent in fault-tolerant systems, combinations of events for all time must be depicted, and the fault tree is designed to predict only the probability of a single event. Additionally, the user must keep track of the exclusivity and independence of the events so that the proper equations for the "AND" and "OR" functions can be used. The third technique, Markov and semi-Markov models, predicts the system reliability by evolving the state probability as a differential equation [3, 5, 6, 7]. The full state vector is available so all possible system probabilities are found, permitting the prediction of various operating mode reliabilities. The differential nature of the model means that only events that occur over the time period dt need to be modeled; sequences of events are captured naturally. A key disadvantage of Markov techniques is that the state space can be prohibitively large for real-world systems.

The benefits of Markov and semi-Markov models (hereafter referred to as Markov models) indicate that they are the preferred technique for modeling fault-tolerant system reliability. The problem of large state spaces is dealt with using a variety of techniques. For example, a common method of state space reduction is behavioral decomposition [8].

In general, there are two types of events occurring in a fault-tolerant system: the occurrence of faults and the handling of the fault. Fault occurrence happens on a slow time scale. Fault handling, the detection of the fault and subsequent system reconfiguration, occurs on a fast time scale. If these two time scales are sufficiently separated, it is possible to model the fault-handling behavior independently. The effectiveness, or coverage, of this fault-handling process is then used as a parameter in the fault-occurrence model. In this way, each potential component failure does not require a group of states to describe the intricacies of that fault-handling process. Instead, each component failure generates two states: component failed and the system successfully reconfigured, and component failed and the system did not successfully reconfigure.

The analytical power and flexibility of Markov techniques has provided a basis for the creation of many computer based tools for the analysis of fault-tolerant systems. In this report we investigate some of the more commonly used tools: CARE III, HARP, SURE, and MARK 1. This study is not meant to be exhaustive, only representative of the current generation. We have not considered all available tools; new ones are appearing daily. Further, we have not considered all potential applications of these tools. By using each tool to examine an identical real-world system that is under design, we hope to show that there are common areas that have not yet been addressed by these tools. Since the example system design is preliminary, coverage issues have not yet been addressed. The system is an integrated control system so it is composed of sensors, actuators, processors, and interfacing equipment. As such, the detailed behavior of these components is not reflected in this analysis.

In general, these tools are still under development. Hence, new versions appear regularly. In this environment it is difficult to do justice to the latest and most sophisticated versions of each tool. Readers should note that this is only a snapshot of an evolving field. To be fair to each tool, we restricted our use of the tools to the features that are described in their respective user's guides. Although we could see potential ways of sidestepping some of the limitations indicated either explicitly or implicitly in the guides, it was decided that these techniques would not be available to the general user and so were not appropriate in this study. Situations where there are possible circumventions are noted.

The goal of this paper is to identify the strengths and weaknesses of the current generation of reliability tools and to make recommendations for the next generation of reliability tools. By applying the tools to an example problem, an assessment of not only the tools, but also their utility in assisting in the entire process of obtaining a reliability prediction, will be explored. Section 2 discusses the selected tools from the current generation. The integrated control system used for the comparison study is described in Section 3. Results of using these tools on the example system are presented in Section 4. Section 5 proposes a potential direction for the next generation of reliability tools and discusses some related research done at CSDL.

Throughout this paper the term FMEA (failure modes and effects analysis) is used. In practice, there are many types of FMEA's. The most common type of FMEA is the single fault FMEA performed on each piece of the system. Often these pieces are down at the level of chips, switches, and even individual wires. While these FMEA's provide part of the input to the process of constructing the Markov model, other information is needed. For example, the consequences of reconfiguration and multiple faults must be considered in the Markov model of the system. Thus, the Markov model represents a **system-level** FMEA where each state is a specific system configuration with an associated impact on the system performance or operating mode. Note that the Markov model has as its inputs the **component-level** FMEA's which represent the impact of component-part faults on the performance of the component. Throughout this paper **system-level** FMEA's will be distinguished from **component-level**, single fault FMEA's.

2. THE CURRENT GENERATION OF TOOLS

In this section we examine some of the current generation of reliability analysis tools. The focus here is on the strengths and limitations inherent in the tools. Once again, we note the snapshot nature of the following descriptions.

2.1 CARE III

CARE III (Computer Aided Reliability Evaluator) [9, 10] is a Markov-based reliability modeling package that exploits behavioral decomposition to reduce the state space. The fault-handling processes are modeled dynamically as a semi-Markov process. The output of the fault-handling models is merged into the fault-occurrence model through the introduction of an approximation. This approximation introduces a negligible error if the time constants of the fault-handling processes are much faster than those of the fault-occurrence processes (four orders of magnitude are recommended in reference [9]). A measure of this error is not given in the output.

The CARE III program calculates the probability of an event at the top of a combinatorial tree. The user defines the tree to give the probability of a certain operating mode or system failure. Components can be grouped into "stages" which permit the efficient description of n-of-m failure modes and these stages can be used as primitive events in the tree. If a system has a variety of operating modes, a tree must be input for each. This can be a quite subtle task when the modes are subsets of one another, such as when a system gracefully degrades through a series of modes. The inclusion of repairs in the model is not discussed in the user's manual. It is not clear if it is possible to model some repair scenarios through clever application of stage definitions.

Variable failure rates are permitted; input forms exist for exponential and Weibull distributions for fault-occurrence and exponential and constant distributions for fault-handling. In the general case, the program solves the system model with a constant-step integration algorithm. However, if there are no semi-Markov fault-handling models, a matrix doubling algorithm is used to integrate the model equations. There are limitations on the sizes of the time steps used for integration due to the assumption that the fault-handling and fault-occurrence time constants fall into a certain class. Integration errors are not provided in the output.

2.2 HARP

HARP (Hybrid Automated Reliability Predictor) [11, 12], is similar to CARE III in that it uses behavioral decomposition to separate the modeling of fault-occurrence and fault-handling events. The unique feature of HARP is that it provides a wide range of frameworks to describe and solve the fault-handling models. These range from constant coverage models, through Markov models solved by integration, to stochastic-extended Petri nets which are solved by simulation. The output of the fault-handling models is merged into the fault-occurrence model through the use of a coverage parameter that is the output of the fault-handling model. This process introduces a negligible approximation if the time constants of the fault-handling processes are much faster than those of the fault-occurrence processes [13, 14]. A measure of this error is not given in the output.

The HARP program calculates the probability of the entire state vector as it evolves in time. Hence, the determination of various operating mode probabilities is possible by grouping states in the output vector. Repairs are easily included since the fault-occurrence model is in a Markov form. An option exists for inputting a fault tree description of the system which is then internally translated into a Markov model.

In addition to accepting failure rates for exponential and Weibull distributions, HARP allows for the input of a range associated with each rate. The corresponding state probability bounds are given in the output. A variety of distributions are available for use in the fault-handling model. The system equations are solved with a variable-step Runge-Kutta algorithm which provides integration error estimates.

2.3 SURE

SURE (Semi-Markov Unreliability Range Evaluator) [15] solves semi-Markov models algebraically. Although the fault-occurrence and fault-handling processes are modeled together, the different time scales between the two permit the program to use a two-step process in solving the model. First, the Markov part (fault occurrence) is solved with a traditional technique such as numerical integration. Then, the semi-Markov part (fault handling) is included through an algebraic technique. SURE solves system models which have constant failure rates and the fault-handling processes are semi-Markov with a mean and standard deviation.

The algebraic solution technique [16] provides a unique approach to predicting the unreliability of semi-Markov systems. However, it can only give the probability of trapping states (hence the term 'unreliability'). If a system has a variety of operating modes, such as when a system gracefully degrades through a series of modes, it is not clear if a series of SURE models could predict the mode probabilities. Repairs are included in the model by "unfolding" the cyclical model n times so that it appears as a purely degrading system with trapping states. While this is not an efficient way to solve standard Markov models, the algebraic solution with unfolding is a very efficient technique for semi-Markov systems with repair. The error introduced by truncating the unfolding is not provided in the output.

There are two techniques available for solving the Markov part of the system. The first uses the first non-zero term of the Taylor series expansion of the state probability. While this is an efficient way to approximate a state probability without solving for the entire state vector, it is only valid for times that are short compared with the time constants along the path to the state. The magnitude of this approximation is included in the solution bounds SURE provides. The alternative solution technique is a Pade expansion with rescaling [17]. Measures of the accuracy of this method are not provided in the output. Both of these techniques require constant failure rates.

The algebraic part of the solution which incorporates the semi-Markov behavior appears as a modification of the pure-Markov part. It reflects the performance of the fault-handling processes. The approximation introduced in this part of the solution is included in the SURE output bounds. These bounds are closer together when there is a larger separation between the fault-handling and fault-occurrence time constants.

2.4 MARK 1

MARK 1 (Markov Modeling Package) [18, 19] is a general program for solving systems described by Markov models. MARK 1 provides a means of solving many Markov models simultaneously and combinatorially merging the results to obtain system reliability. In this way, the state space size is greatly reduced for systems that can be broken into pieces whose interactions can be represented in a combinatorial expression. Modeling of the fault-handling process is not performed in a separate model, as is done in CARE III and HARP, but is modeled together with the fault-occurrence processes.

MARK 1 calculates the probability of the entire state vector as it evolves in time. Hence, the determination of various operating mode probabilities is possible by grouping states in the output vector. Repairs are easily included since the fault-occurrence model is in a Markov form.

The system equations are solved with an algorithm that rescales the matrix at successive powers of 10, providing output in a logarithmic form. This rescaling prohibits the use of time-varying failure rates. Measures of the numerical errors introduced in the integration are not provided in the output.

2.5 Discussion

While these tools have a variety of origins and original intents, there are some general statements that can be made concerning all of them. These tools provide the user with a means of numerically solving the differential equations that are represented by the Markov model. Further, they all provide for increased efficiency by incorporating some type of decomposition to reduce the state space. CARE III and HARP give the user assistance in formulating and inputting the fault-handling models, SURE provides a solution technique for a class of semi-Markov models, and MARK 1 permits the simultaneous solution of a set of Markov models that are merged combinatorially.

Besides these common features, the tools have common deficiencies in two areas. First, none of the tools provides **complete** error analysis. To show the undesirable nature of incomplete error analysis consider the situation where a user has obtained a solution for a particular problem. The tool does not inform the user of the accuracy of the solution and the user cannot draw conclusions without knowledge of the solution's integrity. If, for example, a comparison of the tool's results and the design goal show a difference of 10%, the user needs to know if the output has one digit of accuracy in solving this particular problem or if it has four digits of accuracy. The solution with one digit of accuracy is not sufficient to discern whether this design meets its goal yet the four digit output provides the resolution needed to make this decision. Hence, the user does not know how to interpret tool results or even whether the tool has provided a valid result for the user's problem unless an accounting of all of the solution errors are available.

Secondly, these tools do not give any assistance with the problem of generating the fault-occurrence model. Before the tool can be used a pre-analysis is required to obtain the fault-occurrence model from the description of the system. It will be shown in the following sections that this process of manual model generation is an analysis bottleneck with many opportunities for error. Further, the model generation process is that of constructing a **system-level** failure modes and effects analysis (FMEA) from the **component-level**, single fault FMEA's. This means that the model is nontrivially derived from the system description including its architecture and operating rules. Thus, any changes in system description require that the model (the system-level FMEA) be completely regenerated. In summary, all of these tools require the user to have a knowledge of Markov models to be able to generate the fault-occurrence model and numerical analysis to have an understanding of **all** of the sources of error. It is these deficiencies we hope to focus on in the next generation of tools.

3. EXAMPLE PROBLEM

To provide an environment to use these reliability analysis tools, a hydraulic actuator control system was selected as an example. As shown in Figure 1, the system is an integrated control system containing processors, actuators, and sensors. Actuation is accomplished by hydraulically moving the ram at the left-hand side of the figure. There are two electro-hydraulic channels and an emergency mechanical-hydraulic channel. Each electro-hydraulic channel can control the actuator in either an automatic mode using the channel's processor or in a manual mode using the manned control station. In both cases, electrical commands are passed to the channel's servo valve amplifier which generates the hydraulic control signal. The emergency mechanical-hydraulic channel is implemented through mechanical linkages to pilot valves which port hydraulic fluid to the ram. A series of transfer valves permits switching between the two electro-hydraulic channels and the mechanical-hydraulic channel. Appropriate feedback and indicator signals are available for each mode of control. External system data is available through two signal data converters which are cross-strapped to the two control processors.

Hence, we are interested in modeling the probability of four system modes: automatic, manual, emergency, and null. Operating rules require that use of the automatic mode is permitted only if the manual mode on that channel is operational. In each case, the mode indicated is the highest mode obtainable in that state. For example, if one channel does not have its automatic or manual available yet the other channel has its automatic and manual available, then the system is in the automatic mode. Note that in this particular example the status of the emergency system is not considered since it is assumed that the system will be operating in the more desirable automatic mode. The null mode is the condition where the system cannot be operated.

Using the architecture of Figure 1 with the above mode descriptions, the Markov model of Figure 2 was generated. The model is truncated at the third-failure level and the states at the second-failure level have been aggregated into four states associated with the four operating modes. Repair transitions are indicated; the transition rates from the second-failure level to the first contain an approximation that permits aggregation of states at the second-failure level. The model has 33 states. Constant failure and repair rates are assumed. It required four hours to build this model, i.e., determine the transition rates indicated in Figure 2. An overview of the model construction process and a detailed discussion of the approximations used are contained in reference [22].

A few notes need to be made about the model of Figure 2. First, this example represents a real system in a stage of design where a feasibility analysis is being performed. It was of interest to the designers at this preliminary stage to predict the mode probabilities assuming that all component failures could be perfectly covered. Therefore, the Markov model does not have any transitions representing uncovered failures. Hence, this is a fault-occurrence model. Second, the operating procedures used for this system do not involve specific reconfiguration rules. It was of interest to the designers to know what operating modes are available to choose from at a given moment in time. The above description of selecting the highest operating mode is a subset of interest.

The model in Figure 2 has been truncated at the third-failure level. This truncation is an approximation used to avoid the state space explosion problem associated with the higher failure levels. Instead of explicitly modeling all 29 failure levels, we take advantage of the fact that the states at most of these higher failure levels have vanishingly small probabilities. State 33 is used to bound the error introduced by this model truncation. For example, the probability of the null mode can be approximated by the null mode states at the first- and second-failure levels (states 28 and 32). Clearly, the sum of the probabilities of these two states provides a lower bound on the null mode probability since contributions from states at the third- and greater-failure levels have not been included. If we treat the state at the third-failure level (state 33) as being composed entirely of null mode contributions, it provides, when summed with states 28 and 32, an upper bound on the null mode probability. This is an upper bound because all null mode contributions at the first- and second-failure levels have been accounted for exactly, and all configurations at the third- and greater-failure levels are counted as being the null mode. Similar calculations can be done for the other operating modes where state 33 provides a measure of the truncation approximation bound by accounting for all possible appearances of the given mode at the higher failure levels. Hence, state 33 is referred to as the truncation bound state.

To permit a comparison of the reliability tools, the following are assumed. The initial condition of the system is that there are no failures. Hence, at time = 0 the probability of state 1 in Figure 2 is 1. Since the CARE III user's guide does not indicate how to incorporate repairs in the stage descriptions, we will use the Markov model in Figure 2 without repairs for these evaluations. The model without repairs is used, in fact, for predicting the performance of the system during limited time periods during which no repairs are possible. It is noted that the inclusion of repairs would not drastically alter the conclusions of this study. The simulation period is 100 hours. We wish to obtain the probability of operation in each of the four modes.

4. APPLICATION OF THE TOOLS

In this section we apply each of the tools described in Section 2 to the example problem of Section 3. We constrain the inputs to the tools to be the Markov model and scenario description (mission time, operating modes, etc.) from Section 3. The ease of use of each tool and the accuracy of the results are of interest. The input time referred to in the descriptions is the time required to describe the transition matrix that corresponds to the Markov model in Figure 2 to the specific analysis tool. Due to the evolving nature of these programs, we attempt to focus on their current state with the goal of uncovering the common areas that will need to be addressed in the next generation of tools. A table summarizing the model comparisons is provided in this section.

4.1 CARE III

CARE III was not totally evaluated since it required information additional to the Markov model of Section 3 to satisfy the stage descriptions. More than 5 hours were spent on this task, considerably more than the other tools, yet the input description was not complete. The sticking point is the use of stages to describe the operating modes. This is a combinatorial description which is not trivially obtained from the Markov model. In fairness to the CARE III program, it is noted that deriving a combinatorial description from a Markov model is not the standard approach to such problems. For systems which have many replicated elements operating in an n-of-m fashion, the stage description is straightforward and efficient. However, for integrated control systems with multiple operating modes, such as the integrated control system of Section 3, the stage descriptions do not permit efficient system descriptions.

4.2 HARP

HARP is the largest of the programs evaluated, due primarily to the great variety of fault-handling model descriptions and the associated solution algorithms that are available. There are some restrictions on the input forms used to describe the transition matrix which made the input process somewhat difficult. It took 3 hours to input the Markov model. HARP provides a full state vector output. The accuracy of the results appears to be very good with 15 digits of accuracy reported for the numerical integration portion of the analysis (Table 1). It should be noted that obtaining 15 digits of precision for this calculation from a machine with approximately 16 significant digits is not a trivial task. An inspection of the integration code [20] shows that errors introduced by approximating the continuous derivative are carefully accounted for but errors due to machine roundoff seem to be dealt with in a cursory manner. We suspect that HARP may not be, in fact, obtaining 15 digits in its result. However, for this case the solution accuracy is clearly better than 13 digits which is sufficient for virtually any real system. The numerical integration portion of the program showed itself to be very robust to a variety of tests we performed.

4.3 SURE

SURE provides straightforward formats for the specification of the Markov model. The time required for model input was 1 hour. Results were obtained for only the trapping states, hence the probabilities of the automatic, manual and emergency modes are not available (Table 1). The program generated the solution extremely quickly. The reason for this speed is two-fold. First, the full state vector is not calculated when the first-term solution is used, just the probability of the trapping states. Second, SURE has the ability to model semi-Markov systems but this part of the tool was not exercised. Therefore, SURE only needed to calculate the probability of the paths for the Markov part of the model. Since the simulation period is short with respect to the system time constants (inverse of the failure rates), the solution using the first, non-zero term of the Taylor series expansion is acceptable as is shown by the bounds in Table 1, SURE first-term solution. The ability to use this approximation without having to use the full Pade expansion solution provides a substantial time savings. The run time using the Pade expansion is an order of magnitude greater than the first-term solution (see Table 1, Pade solution). Since there are no fast transitions, no approximations, excepting integration and roundoff, are introduced in the solution. Therefore, bounds appear only for the solution where the first, non-zero term of the series is used. We note that a front-end program for SURE called ASSIST [21] exists, but was not included in this evaluation.

4.4 MARK 1

MARK 1 provides the user with a very efficient and straightforward scheme for inputting the description of the Markov model. In fact, MARK 1 required the least time for input: less than 1 hour. Additionally, this efficient input scheme extends to the user specifications for graphical and tabular output. Results were obtained for the entire state vector (Table 1). MARK 1 uses a default time step of 10^{-5} hours. The time constants of this example system only require a time step of 10^{-1} hours. Using this shorter time step gives a significantly shorter run time. While numerical error bounds are not provided, tests show that the results have an acceptable level of accuracy. It is noted that the algorithm used in MARK 1 lends itself to some potential error prediction algorithms; these are discussed in Section 5.3.

4.5 Discussion

Table 1 shows a comparison of the solutions for each mode and for state 33, which is used to bound the model truncation error. There are no discrepancies among the results. The number of significant digits is derived from the program output. HARP provides a measure of integration error showing that 15 digits of accuracy were retained. SURE, using the first, non-zero term of the Taylor series, shows bounds that indicate that the solution is known to 2 digits. SURE, when using the Pade expansion, and MARK 1 give no indication of error. It should be noted that, in general, none of the programs give a full accounting of the errors introduced in the analysis.

Unfortunately, this example does not present a challenging numerical problem; note that even the first-term analysis yields 2 significant digits. Even so, serious questions remain. How does one interpret the results from the two programs that give no measure of error? Even though much computation time is spent reducing the errors, state 33, which provides a bound on the error introduced by model truncation, shows that the mode probabilities are modeled with 2 to 5 significant digits. Given this modeling accuracy, is there a need for 10 or more significant digits in the numerical portion of the solution? For example, are the 15 digits HARP provides needed or do the 2 that SURE generates suffice? More specifically, can HARP's run time be improved if only 2 significant digits are required? Finally, how does the user know when model assumptions are being violated, such as the time constant separation required for behavioral decomposition, if no measure of the induced approximation is provided?

The input times for the tools were less than 1 hour for MARK 1, 1 hour for SURE, and 3 hours for HARP. Often, input times are dependent on the user's familiarity with the tool. We do not feel this was the primary cause of the input time differences. The programmer for these experiments had a similar level of familiarity with these three tools. Rather, the input times seem to be related to the number of input lines required to describe the Markov model. It should be noted that the input times have a close correlation to the tool's generality. HARP has the most powerful fault-handling models, SURE is somewhat constrained due to its lack of behavioral decomposition, and MARK 1 does not permit semi-Markov or decomposed fault-handling model descriptions. Thus, this problem, which does not include explicit fault-handling descriptions, is most easily described with the tool which has the fewest special features for fault-handling.

It may be viewed that this example is an anomaly since it does not exercise the powerful fault-handling model aspects of some of the tools. However, this example is typical of the type of modeling task required during the design of integrated control systems and highlights the common missing element in these tools. The manual construction of this fault-occurrence model (Section 3) required **4 hours**. Input times ranged from less than **1 hour** to **3 hours** and solution times were in the range of **seconds**. Hence, the major investment of time and money is currently in the generation of the fault-occurrence model from a system description, followed by the process of inputting that model into the tool.

This is not to imply that these tools do not provide a real benefit in the analysis of fault-tolerant systems—their ability to aid the user in behavioral decomposition makes many previously intractable problems solvable. Rather, their lack of assistance in generating the fault-occurrence model, the task which currently comprises the major time investment in obtaining system reliability, makes this task the design and analysis bottleneck for the current generation of tools. Some of these tools have "user friendly" interfaces available, such as SURE's front end program ASSIST [21]. These programs may have the ability to reduce the model input times but they do not address the 4 hours of pre-analysis required to **construct** the model. In summary, significant gains in productivity can not be made from improving the program run times; instead we must address the time-consuming tasks of model construction and program input.

5. THE NEXT GENERATION OF TOOLS

The process of performing a reliability evaluation may be divided into steps, as shown in Figure 3. Given a system description which includes the architecture, reconfiguration rules, and the operating mode requirements the fault-handling and fault-occurrence models can be constructed. The outputs of the fault-handling models are incorporated as parameters in the fault-occurrence model. The input of system parameters and the modeling goal permits problem specific approximations to be introduced. Finally, an analysis program operates on the resulting system of equations to produce the reliability prediction.

The current generation of reliability analysis tools has concentrated on means of improving the efficiency of the description and solution of the fault-handling processes and providing a set of solution algorithms for the full, system model. As was shown in the preceding sections, the current generation of tools has succeeded in improving user efficiency in these areas to the point that the problem of describing the fault-occurrence model is now the major analysis bottleneck.

For the next generation of reliability tools, we propose that techniques be developed to improve the efficiency of generating and inputting the fault-occurrence model. Further, the goal is to create an environment that permits a user to provide a top-down design description of the system from which a Markov reliability model is automatically constructed. In this way the user is relieved of the tedious and error-prone process of model construction and an independent validation of the system's operation is obtained. Additionally, the process of exploring variations of the system design is facilitated since the user need only change the system description; the tool automatically generates the new Markov model.

An additional benefit of automating the tedious model construction process is the opportunity for reducing the specialized knowledge required of a user. While it will always be desirable for the user to have a knowledge of the fundamentals of reliability analysis, the tool will have the ability to automatically handle "advanced" concepts such as those involved in controlling the size of the state space while maintaining sufficient model integrity. Hence, the user need only be an expert in the system he is analyzing; the expertise in reliability analysis techniques is supplied by the analysis tool. In this vein, the next generation of tools should have more robust analysis abilities when compared to the current generation. Further, in keeping with the concept of reduced user expertise, the tool should have the ability to deliver a user-requested accuracy by selecting among its solution techniques and the free parameters of the analysis.

These two areas of enhancement for the next generation reliability tool, improved analysis/error reporting techniques and automated fault-occurrence model construction, are discussed in the following sections. Section 5.3 discusses some relevant research at CSDL.

5.1 Accuracy vs. Efficiency

In general, a solution of any accuracy can be obtained given enough time (money). However, the user has in mind certain acceptable ranges for these commodities. Usually, a certain minimum accuracy is required and, given that requirement, the cost of obtaining a solution can be determined. This determination is by no means trivial. However, it may be possible since we are dealing with a restricted class of problems and a limited set of algorithms.

Solution errors arise from three sources: modeling approximations such as behavioral decomposition, algorithm approximations such as discretizing a continuous derivative, and roundoff errors due to finite machine precision. The primary goal of the improved evaluation ability of the next generation of reliability analysis tools is to relieve the user of all decisions in selecting model approximations, algorithms, and parameters. Instead, the user is queried for an acceptable error bound for the **entire** analysis and the tool automatically makes selections to obtain this accuracy with the minimum cost. On output, the user is informed of the accuracy of the solution so that proper interpretation is possible.

For the next generation of reliability analysis tools, we propose to use a variety of algorithms to solve the Markov model. The tool should have the ability to automatically select the set of appropriate algorithms for the specific system being analyzed. Additionally, the tool should select from this set of algorithms the one that gives the appropriate level of accuracy using the minimum of resources (memory, cpu time, etc.). Any parameters associated with the algorithm, such as the time step or the number of terms in a truncated series required to give the user-selected accuracy, are determined automatically by the tool. Hence, the user is relieved of all numerical analysis expertise; the tool provides the appropriate knowledge.

The point may be raised that tracking all errors as they propagate through the analysis is very resource consuming. While this is true to an extent, we believe that this extra burden is partially compensated by the increased efficiency of the algorithm and parameter selection. Further, an answer with **total** error bounds is substantially more valuable than an answer with no error bounds that was obtained in half the time.

5.2 Fault-Occurrence Model

Although the procedure for constructing the Markov reliability model is conceptually straightforward, the implementation is usually a formidable task due to the effort required to perform the **system-level** failure modes and effects analysis (FMEA) from the **component-level**, single fault FMEA and to cope with the state proliferation problem associated with these models. As discussed above, the current generation of tools has dealt with this state space problem through behavioral decomposition. Although aid is provided in specifying the fault-handling processes, the user is currently responsible for providing the fault-occurrence model.

Experience indicates that the formulation of a fault-occurrence model for a complex system requires considerable effort. Therefore, this step in reliability prediction presents a design/analysis bottleneck which severely limits the number of candidate designs that can be examined in practical situations. It is also possible that significant failure modes may be omitted from the model formulation. This situation may arise from errors introduced by the analyst as a result of the iterative, and hence, tedious nature of the model formulation, or through the omission of a failure mode because it was not identified during the system-level FMEA.

In light of the above, we conclude that a 'missing-link' in reliability evaluation techniques and associated research efforts is a computer-aided engineering (CAE) tool which automatically constructs a Markov fault-occurrence model from a top-level system description. This CAE tool would reduce modeling errors and would serve both as a design and validation aid for a broad class of fault-tolerant systems. In order to extend its range of applicability, the CAE tool should exploit state-reduction techniques [22] which mitigate the state proliferation problem. Further, this tool must be integrated with the current tools in order to obtain a tool which generates a Markov reliability model capturing both the fault-occurrence and fault-handling processes.

Hence, the next generation tool should be able to construct a Markov model for a system by automatically performing the underlying **system-level** FMEA when given **component-level**, single fault FMEA's and a system description. In this manner, every phase of the analysis/evaluation process will be computer aided. Consequently the level of effort associated with constructing these models for large, complex systems is mitigated. Furthermore, pre-analysis efforts are minimized or eliminated because the task of the designer is now one of describing the system and its operation, not analyzing it. Modeling errors will be reduced and the tool will serve both as a design and validation aid. These features will be enhanced by providing the CAE tool with the ability to explain the automated process associated with the system-level FMEA and model formulation.

While the process of decomposing fault-handling and fault-occurrence behaviors reduces the state space, the fault-occurrence model's state space may still be too large for analysis. Additional techniques are needed to reduce the state space. For example, the integrated control system of Section 3 has 29 components. The total number of states in the Markov model is more than 10^8, assuming that the order of component failure does not impact system performance. The time required for manual construction of such a model is prohibitive. However, this approach to model construction is so straightforward that a simple algorithm may be written. In fact, with the addition of aggregating system loss states, this is the model that HARP would construct if given a fault tree description of the system. The difficulty with this approach is that the cost of solving the system equations for a system with n state variables increases as n^2 or n^3, depending on the algorithm used. Further, models with this many states do not lend themselves to intuitive insights.

Therefore, it is not enough for the fault-occurrence model construction to proceed blindly in its task. Means must be employed to reduce the state space size of the fault-occurrence model, itself. Clearly all 10^8 states are not equally likely to occur. For example, the state with 29 failures is extremely unlikely in this reliable system. This concept of focusing the model construction efforts on the states that are most likely, and therefore contribute the most significance to the solution, is at the heart of state space reduction techniques such as model truncation [22]. The process of model truncation, or using most other state-reduction techniques, introduces an approximation; a measure of that approximation must be provided in the output.

The goal of automating the construction of the fault-occurrence model is to relieve the user of the need to possess all of the expertise required for this task. In this way the tool provides all of the relevant expertise to perform the reliability analysis, including techniques to mitigate the state proliferation problem. In addition to providing a reliability analysis tool to a larger audience, this automated approach relieves the user of the tedious and error prone process of model construction, permitting an efficient exploration of the design space. Hence, an independent validation of the design is generated: the user indicates how the system operates, the automated tool shows how well the system meets its operating requirements.

5.3 Relevant Research

During the past two years CSDL has been performing an Internal Research and Development project focused on demonstrating the feasibility of automated fault-occurrence model construction from a user's system description with appropriate state reduction [23, 24, 25]. Additionally, issues of algorithm selection and error prediction have been addressed under this project. This section reports on some of the progress made in these areas.

5.3.1 Error Prediction

The ability to predict the errors induced in the solution of a Markov model as a function of the algorithm parameters serves as a basis for the automatic selection of the solution algorithm and associated parameters. Each solution algorithm generates errors in a unique way. Hence, algorithm selection is not simply a function of the algorithm that is fastest; selection must be made based on the algorithm that is the fastest and provides the accuracy requested by the user.

Solution errors, other than modeling approximations which are discussed in Section 5.3.2, result from two causes: roundoff errors due to finite machine precision and truncation errors due to approximations in the algorithm. An example of a truncation error is the integration error introduced by approximating a continuous differential equation with a discrete-time difference equation. Every solution algorithm presents different levels of difficulty in predicting its errors.

For example, research at CSDL has shown that the solution algorithm used in MARK 1 lends itself to very simple error predictions [26]. MARK 1 solves the Markov model by taking powers of the transition matrix. This is a very fast and stable technique for solving time-invariant systems [17].

The truncation error introduced through approximating the continuous differential depends directly on the fastest time constant and the discretization interval (time step) chosen. The fastest time constant is the inverse of the largest eigenvalue. For systems that only degrade, the eigenvalues can be found by inspection (they lie on the diagonal). For systems with repair, the largest eigenvalue can be approximated by a decomposition that decouples the (fast) repair processes from the (slow) fault-occurrence processes. Using a time step of Δt and a largest eigenvalue of λ, $(\lambda\Delta t)^2$ provides a measure of the local integration error. Means exist for extending this to a global error [26].

The roundoff error introduced through finite machine precision can be propagated as follows. At time = 0, the relative error in the matrix elements are characterized by the machine precision. Squaring the matrix results in the elements characterized by relative errors of 2 times the machine precision. Obtaining the nth power of the matrix results in the elements having relative errors equal to n times the machine precision. An unusual property of this algorithm is that for large n the matrix errors become "well mixed", all approaching a common value.

As an example of how the truncation and roundoff errors interact, we return to the integrated control system example. The IBM machine precision for double, floating point numbers is 10^{-16} (sixteen significant digits). The simulation time is 100 hours. MARK 1 was first run with a default time step of 10^{-5} hours. Therefore 10^7 time steps, or powers of the matrix, were needed. The roundoff error is (10^7) $(10^{-16}) = 10^{-9}$, or nine significant digits. The largest eigenvalue (no repairs) is on the order of 10^{-4} hours^{-1}. The global integration error is less than 10^{-16}, or 16 significant digits. Hence, the total relative error is 10^{-9}. To improve the run time a time step of 10^{-1} hours was used. This results in a roundoff error of 10^{-13} and an integration error of 10^{-8}, giving 8 significant digits.

As shown in this example, there is a tradeoff between the two errors as the time step is varied. Larger time steps require fewer calculations to reach a given simulation time so the roundoff error is reduced. However, these larger steps are poorer approximations of the continuous differential so the integration error is increased. The relationship between run time, error, and time step can be written as a function and minima of interest found as a function of the time step. It is often more useful to choose a time step that delivers the user's requested accuracy. For example, if the user desired 3 significant digits in the solution of the integrated control system, a time step of 30 hours could be used. This would provide a substantial reduction in the run time when compared to the default time step.

Not all algorithms may have tractable error predictions. However the ability to predict the error as a function of the analysis parameters permits both the automatic selection of those parameters and the assurance that the result will be useful. This ability to predict the usefulness of the result ahead of time may be of sufficient importance so as to require that all algorithms used in the tool have this property. In any case, all algorithms should provide measures of **all** of the induced errors so the user will know to what extent the assumptions of the analysis have been invalidated and what level of meaning to associate with the solution.

5.3.2 Automated Fault-Occurrence Model Construction

The key innovation suggested for the next generation of reliability analysis tools is the automatic generation of the fault-occurrence model from a user's system description. Research at CSDL in this area has resulted in the Computer Aided Markov Evaluation (CAME) program [24, 25]. This program is written in LISP and resides on a Symbolics computer. The goals of this research are to demonstrate the feasibility and benefits of this mode of model construction.

A block diagram of the CAME program is shown in Figure 4. The user interface provides a graphical means for the user to input the system description and application information. This input is stored in the system data base. Automatic Markov model construction is performed by the model builder using the description in the system data base and the rules for model building in the procedural knowledge base. The resulting model is stored in the model data base. The model builder also uses the information in the model data base in performing certain model reductions such as state aggregation. The explanation module permits the user to examine and challenge both the reasoning process underlying the system-level failure modes and effects analysis and the use of model-building procedures.

The rules that are at the heart of the CAME system are formalisms of the techniques and expertise used at CSDL in the construction of Markov models. For example, some of the rules relate directly to the process of state generation: given that the system is in a certain state, the following procedures are applied to create subsequent states. Another example of a rule is deciding when model truncation should occur and what types of state aggregation are appropriate for the given failure level.

To demonstrate some of the power inherent in automatic generation of reliability models, the integrated control system example was evaluated using the CAME program. Each input window has mouse-driven menus that permit the creation and labeling of objects, connections between objects, and associated parameters.

Figure 5 shows a screen image of the first user input window. The System Architecture Window contains not only information about the connections between the components, but also information about the components themselves. Note that this architecture diagram has a one-to-one correspondence to the system architecture in Figure 1. Each circle represents a component and has a failure rate, coverage value, and repair rate associated with it. For this example the coverage values are equal to one so perfect coverage is modeled and the repair rates are set to zero so no repair transitions are created. These parameters, and others that permit grouping of components, are accessible from menus associated with each object. Notice that the arrow heads on the component connectors contain information to be used in the modeling process; information flows bi-directionally between processors (proc-a, proc-b) and buses (ba1, ba2, bb1, bb2) and flows uni-directionally between manual control synchros (s1, s2, p1, p2) and their associated servo valve amplifiers (sva-a, sva-b).

The next user-input window (Figure 6) sets the performance levels, or operating modes, for the system. Starting from the left-hand side of the figure, performance level 1 for the system (control-system p-level: 1) requires that channel a meet the requirements for operating in both automatic and manual. Performance level 2 is similarly described for channel b. Performance levels 3 and 4 require that manual operation be achieved in channel a or channel b, respectively. The ability to operate in the emergency mode is required for meeting performance level 5. Note that the restriction that automatic operation is only permitted if the manual is operational on that channel is enforced by the definitions of performance levels 1 and 2.

The blocks representing automatic or manual mode requirements for channel a, etc. are user-defined blocks. These are defined in the Further Specifications Window (Figure 7). On the left-hand side of the figure, "emergency" is defined to require that 4 elements of the emergency class are unfailed. The class that an object belongs to is specified in the menu associated with the component in the System Architecture Window (Figure 5). Requirements for channel a to operate in the automatic mode (cha-automatic) are defined in the top half of the figure. Cha-automatic requires that eleven of the components in the class "claauto" must be operating, one of the two buses (ba1 or ba2) be unfailed, one of the two signal data converters (sdc-a or sdc-b) be unfailed, and one of the two manual synchros (p1 or p2) be unfailed. Requirements for channel a operating in the manual mode are defined in the lower half of the figure. Similar definitions are included for automatic and manual operation of channel b.

There are some points to notice in this description. The CAME program presumes the system will be operating in the highest performance level possible in each state (i.e.: closest to level 1). The achievement of none of the specified performance levels is a system failure or system loss (called the null mode in this example). The introduction of classes to group components permits the natural description of an n-of-m criteria in the performance specification. The performance requirements reflect some design decisions. For example, the automatic mode cannot be used unless the manual mode is available on that channel.

Finally, notice that the performance level diagrams are **not** success trees. Success trees (the inverse of fault trees) must be exhaustive. If the CAME program required the input of a success tree it would not serve any function since the success tree contains the **system-level** FMEA. Further, a success tree would limit the model formulation to static relationships among the components; this is undesirable since a fault-tolerant system's reconfiguration introduces dynamic dependencies between the components. Therefore, the CAME program requests a description of what capabilities are needed for system operation in the various modes. These capabilities are often expressed, particularly with integrated control systems, as which inputs and outputs are needed. Notice that the performance level diagram does not contain all of the information needed to construct the Markov model; i.e., to construct the **system-level** FMEA. Rather, it contains the **minimum** specification for each operating mode and it must be used in conjunction with the architecture and reconfiguration diagrams. As an example, note that in the performance diagram of Figure 6 some information needed to construct the Markov model, such as the need for a functional path between sensors and processors, is obtained from the architecture diagram. The construction of the Markov model requires information form all three windows: architecture, performance, and further specifications.

Other descriptions of the system are possible. If the user does not want to distinguish which channel the modes occur in, automatic could be defined as requiring either automatic and manual on the a channel **or** the b channel. The manual mode could be similarly defined. Also, abstract objects, representing the performance of channels, could be used to describe the system in a more concise form.

These three windows (architecture, performance, and further specifications) constitute a complete system description. The user may now move to the Markov Modeler Window. Here menus exist for setting the truncation and aggregation rules to be applied. For this system we have selected to truncate the model at the second-failure level **after** the first system-loss state. The aggregation rules have been selected so all system-loss states at a common failure level will be aggregated and states with common performance at the next-to-last failure level will be aggregated.

Selecting the 'construct model' command from the menu causes a **system-level** failure modes and effects analysis to be performed on the user-described system (Figures 5-7) and generates the Markov reliability model in this window (Figure 8). The states are organized into columns at each failure level. At the left is state '0,1': state 1 at the zero-failure level. This is the initial, no-failure state. Various transitions lead to states designated '1,x' at the first-failure level. All system-loss states at the first-failure level have been aggregated into state 'lo-sl-1'. Various failure events cause transitions from operational states at the first-failure level to states at the second-failure level. Following the user-set rules for aggregation, the states at the second-failure level have been aggregated into states at common performance levels, a state for performance levels 1 through 5 (p-1, p-2, ... p-5), and a system-loss state (lo-sl-2). As requested, the model has been truncated at the third-failure level, which is the second failure level **after** the first system loss. All states at the third-failure level have been aggregated into the state 'up-sl'.

The truncation of the model at the third-failure level introduces an approximation. However, because of the way the CAME program performs this truncation, the model provides bounds on this approximation. By considering **all** states at the third-failure level to be system losses, we clearly have an upper limit on system-loss contributions from the third-, and subsequent-, failure levels. The sum of system-loss probabilities from the first- and second-failure levels (lo-sl-1 and lo-sl-2) provide a lower bound on the system-loss probability since they do not contain all possible system-loss configurations. The sum of these two system-loss states and the third-failure level state (up-sl) provide an upper bound on system-loss probability since all losses at the first two failure levels are captured exactly and the 'up-sl' state is an upper limit on all remaining system-loss probabilities. The difference between these two bounds for the system-loss probability, namely the probability of state 'up-sl', is a direct measure of the approximation introduced by the model truncation.

Moving the mouse onto states and transitions in the Markov Model Window permits the inspection of the properties of that state or transition. This is equivalent to inspecting the system-level FMEA. It is through an examination of the FMEA that the designer confirms the system operation and the Markov model is checked. For example, inspecting state '1,6' shows that this state is the result of a failure of the bus ba2 (see Figure 9). The system is operating at performance level 1. Lists of objects in use and the originating state and destination states are shown along with the associated failure events.

The CAME program also has an input window that permits the user to describe the reconfiguration process involved in the redundancy management strategy. As with the other input windows, this one is mouse- and menu-driven. Although no specific reconfigurations were given in the example problem, Figure 10 shows some possibilities. The top row indicates that the processor in use (proc-in-use) is initially processor a (proc-a) and if the condition that the system cannot achieve performance level 1 is met, the system reconfigures to processor b. Similar reconfigurations are set for the bus pairs. The CAME program uses this information, which may be referred to in the performance requirements as "current proc-in-use", etc., in its generation and evaluation of the model states. An important point to notice is that this description is not exhaustive; information from the other input windows is used in conjunction with this input in the model construction.

The manual construction of the fault-occurrence model for the integrated control system (Section 3) required 4 hours. Inputting this description into the various current generation tools required from 1 to 3 hours. Taking the best case, a total of 5 hours was required to analyze this system using the current generation tools. Inputting the graphical description to the CAME program took 2 hours, including the time required for the user to learn how to use the CAME program. The CAME program constructed the model in 3 minutes. Hence, the next generation tool provided a time reduction of greater than a factor of 2. This measure of improvement in productivity does not reflect the reduction of required specialized knowledge to use the next generation tool.

This example does not tell the whole story. First, the integrated control system example represents systems of moderate complexity. In our IR&D research, we performed a similar productivity comparison for an electronic, two-channel engine controller that had more than 40 components, imperfect coverage, and extensive reconfiguration rules. The process of manually constructing the model and inputting that description into a current generation tool required 80 hours. Graphically describing the system to the CAME program took 5 hours and the automatic model construction was performed in 40 minutes. Further, the CAME program found 15 errors in the manually constructed model.

Second, once a system is described to the CAME program, design modifications are simple. These may range from changes in operating rules to changes in the system components or the system topology. Thus, the next generation tool provides an environment for exploring the entire design space. For example, implementing the specific reconfigurations of Figure 10 required 10 minutes with another 3 minutes for automatic model construction. Manually, the addition of specific reconfigurations, which introduce sequence dependencies requiring a complete reevaluation of the system-level FMEA and a reconstruction of the model, required another 5 hours. Further, repairs are included by simply indicating the repair rates for each component in the System Architecture Window. This addition was done in 10 minutes with 3 more needed for constructing the model. Comparing this result to that manually generated in Section 3 showed 2 errors in the manually constructed model.

Comparing the CAME program's abilities to the goals stated for the next-generation tool, shows that it provides an independent means of validating both the system operation and the reliability model since it automatically generates the system-level FMEA for the system. Inspecting states allows the user to confirm the system operation and the validity of the model. The automation of the fault-occurrence model increases productivity and the ability to alter the system description permits exploration of the design space. The goal of having the tool contain all the Markov modeling expertise needed for reliability analysis has not yet been achieved; the program asks Markov-type questions such as specifying truncation level, etc.

Currently, the CAME program can generate Markov reliability models for systems with constant, non-state-dependent coverage parameters and manual repairs. Several model truncation rules are available which produce upper- and lower-bounds on the system reliability measures. Exact and/or approximate state aggregation to states at a common user-specified performance level is another model-constructing option.

Hence, the CAME program can automatically perform the **system-level** FMEA based on the **component-level** FMEA's and a top-level system description. This capability has been tested on the following systems: an abstraction of an Advanced Information Processing System (AIPS) architecture, a hydraulic actuation system, an electronic jet engine controller, and a submarine control system.

6. CONCLUSIONS

The process of performing a reliability evaluation may be divided into steps. The fault-handling and fault-occurrence models can be constructed, given a system description which includes the architecture, reconfiguration rules, and operating mode requirements. The outputs of the fault-handling models are incorporated as parameters in the fault-occurrence model. The input of system parameters and the modeling goal permits problem specific approximations to be introduced. Finally, an analysis program operates on the resulting system of equations to produce the reliability prediction.

The current generation of reliability analysis tools has concentrated on means of improving the efficiency of the description and solution of the fault-handling processes and providing a solution algorithm for the full system model. The tools have succeeded in improving user efficiency in these areas to the point that the problem of describing the fault-occurrence model is now the major analysis bottleneck.

For the next generation of reliability tools, we have proposed that techniques be developed to improve the efficiency of the fault-occurrence model generation and input. Further, the goal is to provide an environment that permits a user to provide **component-level** FMEA's and a top-down design description of the system from which a Markov reliability model is **automatically** constructed. In this way the user is relieved of the tedious and error-prone process of model construction. Since the model is constructed automatically from the system description, the design space can be efficiently explored by simply altering the system description; each new model is constructed automatically. Finally, an independent validation of the system's operation is obtained by comparing the automatically constructed **system-level** FMEA with the designed system operation.

An additional benefit of automating the model construction process is the opportunity for reducing the specialized knowledge required of a user. While it will always be desirable for the user to have a knowledge of the fundamentals of reliability analysis, the tool will have the ability to automatically handle "advanced" concepts such as those involved in controlling the size of the state space while maintaining sufficient model integrity. Hence, the user need only be an expert in the system he is analyzing; the expertise in reliability analysis techniques is supplied by the analysis tool. In this vein, the next generation of tools should have more robust analysis abilities when compared to the current generation. Further, in keeping with the concept of reduced user expertise, the tool should have the ability to deliver a user-requested accuracy by selecting among its solution techniques and the free parameters of the analysis.

IR&D research at CSDL has shown the feasibility of achieving these goals. Much work is left to do in further automating the fault-occurrence model generating ability and integrating it with the best features of the current generation tools.

7. REFERENCES

1 "TIGER Computer Program," Naval Sea Systems Command, Washington, D.C., 1985.

2 Rubinstein, R.Y., Simulation and the Monte Carlo Method, J. Wiley, Inc., New York, 1981.

3 Shooman, M., Probabilistic Reliability: An Engineering Approach, McGraw-Hill, New York, 1968.

4 Vesely, W.E., et al., Fault Tree Handbook, NUREG-0492, Office of Nuclear Regulatory Research, US Nuclear Regulatory Commission, Washington, D.C., January, 1981.

5 Howard, R.A., Dynamic Probabilistic Systems, J. Wiley, Inc., New York, 1971.

6 Kemeny, J.G. and J.L. Snell, Finite Markov Chains, Springer-Verlag, New York, 1976.

7 Geist, R.M. and K.S. Trivedi, "Ultra-high Reliability Prediction for Fault-Tolerant Computer Systems," IEEE Transactions on Computers, Vol. C-32, December 1983, pp. 1118-1127.

8 Trivedi, K.S. and R.M. Geist, "Decomposition in Reliability Analysis of Fault-Tolerant Systems," IEEE Transactions on Reliability, Vol. R-32, December 1983, pp. 463-468.

9 Bavuso, S.J., P.L. Peterson, D.M. Rose, CARE III Model Overview and User's Guide, NASA Technical Memorandum 85810, June 1984.

10 Trivedi, K.S. and R.M. Geist, A Tutorial on the CARE III Approach to Reliability Modeling, NASA Contractor Report 3488, December 1981.

11 Geist, R.M., K.S. Trivedi, J.B. Dugan, and M.K. Smotherman, "Design of the Hybrid Automated Reliability Predictor," in Proceedings of the 5th IEEE/AIAA Digital Avionics Systems Conference, November 1983.

12 Dugan, J.B., K.S. Trivedi, M.K. Smotherman, and R.M. Geist, The Hybrid Automated Reliability Predictor, Duke University, 1985.

13 McGough, J., M.K. Smotherman, and K.S. Trivedi, "The Conservativeness of Reliability Estimates Based on Instantaneous Coverage," IEEE Transactions on Reliability, Vol. C-35, July 1985.

14 Trivedi, K.S., J.B. Dugan, R.M. Geist, and M.K. Smotherman, "Modeling Imperfect Coverage in Fault-Tolerant Systems," in Proceedings of the IEEE FTCS-14 Conference, 1984, pp. 77-82.

15 Butler, R.W., The Semi-Markov Unreliability Range Evaluator (SURE) Program, NASA Technical Memorandum 86261, July 1984.

16 White, A.L., "Upper and Lower Bounds for Semi-Markov Reliability Models of Reconfigurable System," NASACR-172340, April 1984.

17 Moler, C. and C. van Loan, "Nineteen Dubious Ways to Compute the Exponential of a Matrix," SIAM Review, Vol. 20, No. 4, October, 1978, pp. 801-836.

18 Lala, J.H., MARK 1 - Markov Modeling Package, The C.S. Draper Laboratory, Inc., Cambridge, Mass., March 1983.

19 Lala, J.H., "Interactive Reductions in the Number of States in Markov Reliability Analysis," AIAA Guidance and Control Conference, August 1983.

20 Shampine, L.F., and H.A.Watts, "Global Error Estimation for Ordinary Differential Equations," ACM Transactions on Mathematical Software, Vol. 2, No. 2, June, 1976, pp. 172-186.

21 Johnson, S.C., ASSIST User's Manual, NASA Technical Memorandum 87735, August 1986.

22 Babcock, P.S., An Introduction to Reliability Modeling of Fault-Tolerant Systems, CSDL-R-1899, The C.S. Draper Laboratory, Inc., Cambridge, Mass., September 1986.

23 IR&D Annual Report for DFY-87, The C.S. Draper Laboratory, Inc., Cambridge, Mass., November 1986.

24 Schabowsky, Jr., R.S., M.A. Luniewicz, and M.A. Radlauer, "Automated Reliability Model Construction," in Proceedings of AIAA Guidance, Navigation, and Control Conference., August 1986.

25 Schabowsky, Jr., R.S., M.A. Radlauer, and J. Brandner, "Automated Markov Reliability Model Formulation," in Proceedings of Robotics and Expert Systems Conference., June 1986.

26 Babcock, P.S., B. McCarragher, Predicting Numerical Error Propagation in the Solution Of Markov Models, in publication, The C.S. Draper Laboratory, Inc., Cambridge, Mass.

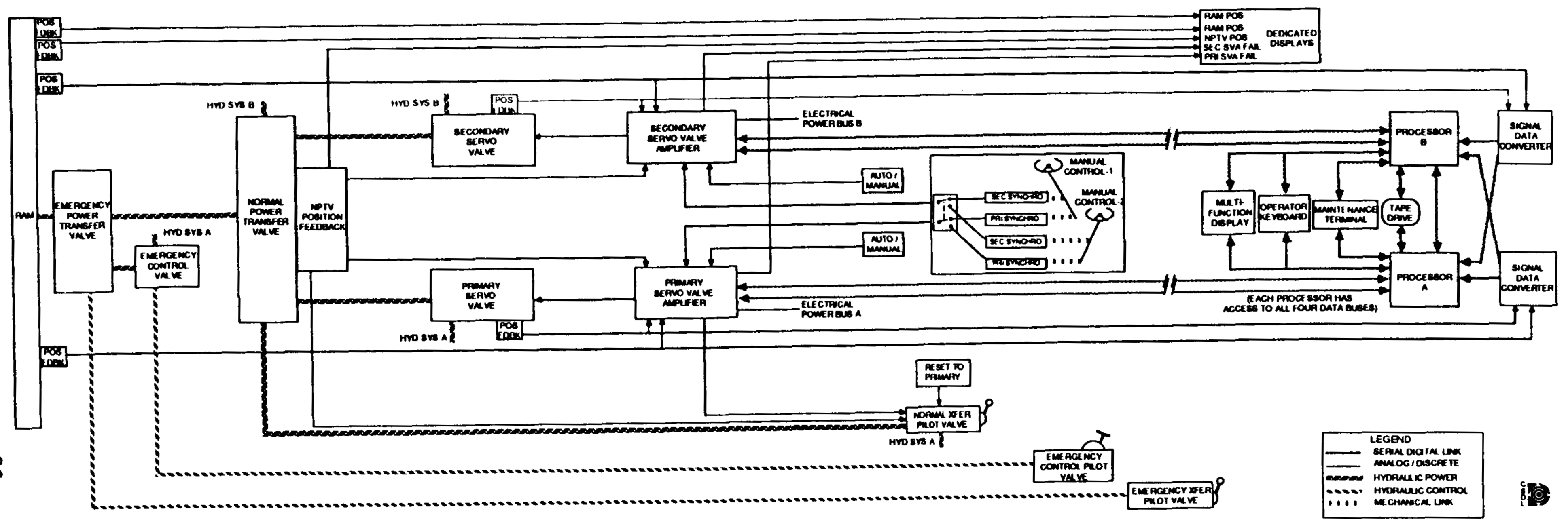

FIGURE 1: INTEGRATED CONTROL SYSTEM ARCHITECTURE

ORIGINAL PAGE IS
OF POOR QUALITY

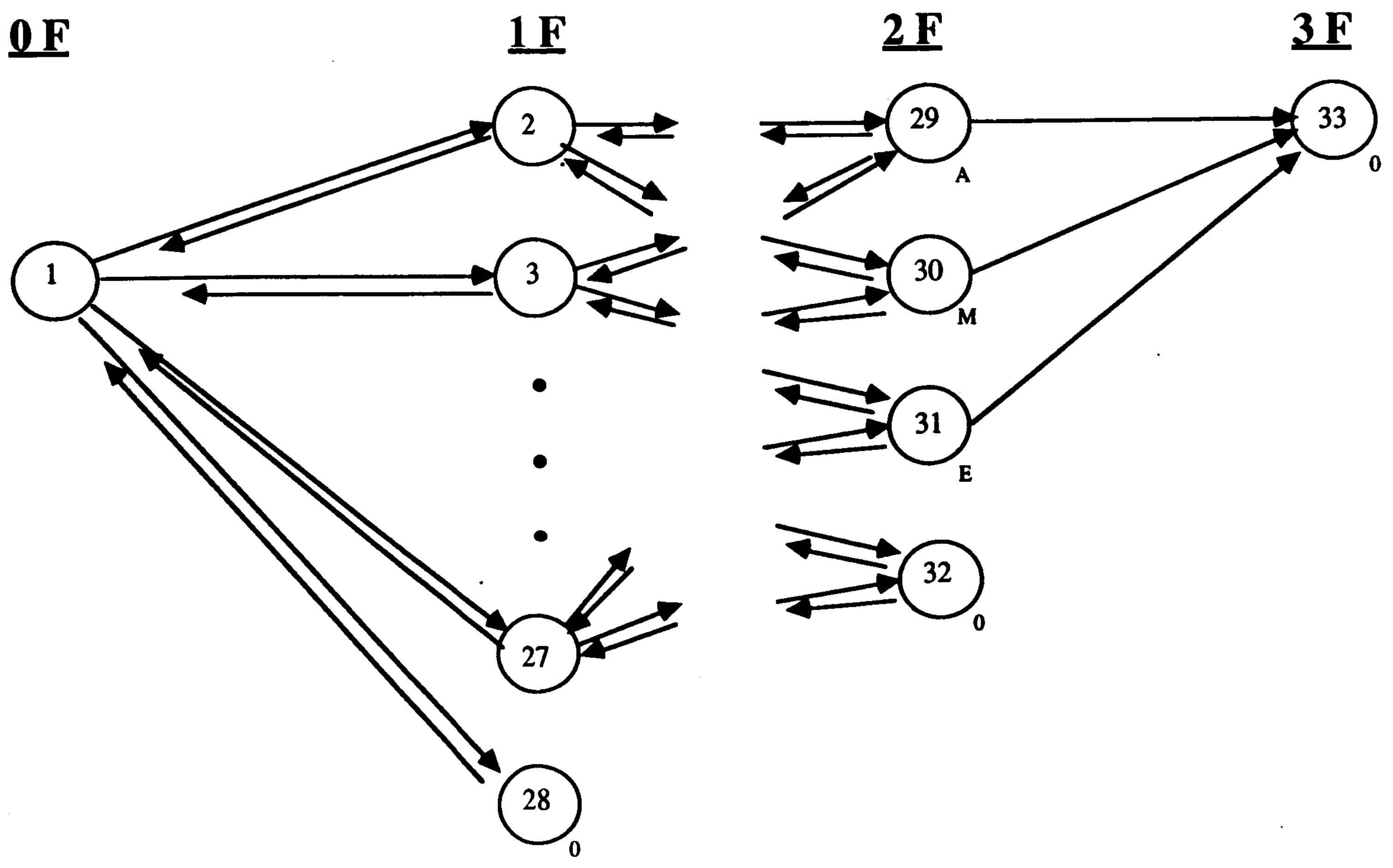

FIGURE 2: INTEGRATED CONTROL SYSTEM MARKOV MODEL

MODE	HARP	SURE (FIRST TERM)	SURE (PADE)	MARK 1
A	0.99774	N / A	N / A	0.998
M	4.6764E-4	N / A	N / A	4.68E-4
E	1.4801E-3	N / A	N / A	1.47E-3
0	2.7110E-4	2.71E-4 - 2.81E-4	2.7110E-4	2.71E-4
STATE 33	3.7359E-5	3.73E-5 - 3.92E-5	3.7359E-5	3.74E-5
SIGNIF. DIGITS	15	2	N / A	N / A

TABLE 1: MODE PROBABILITY RESULTS

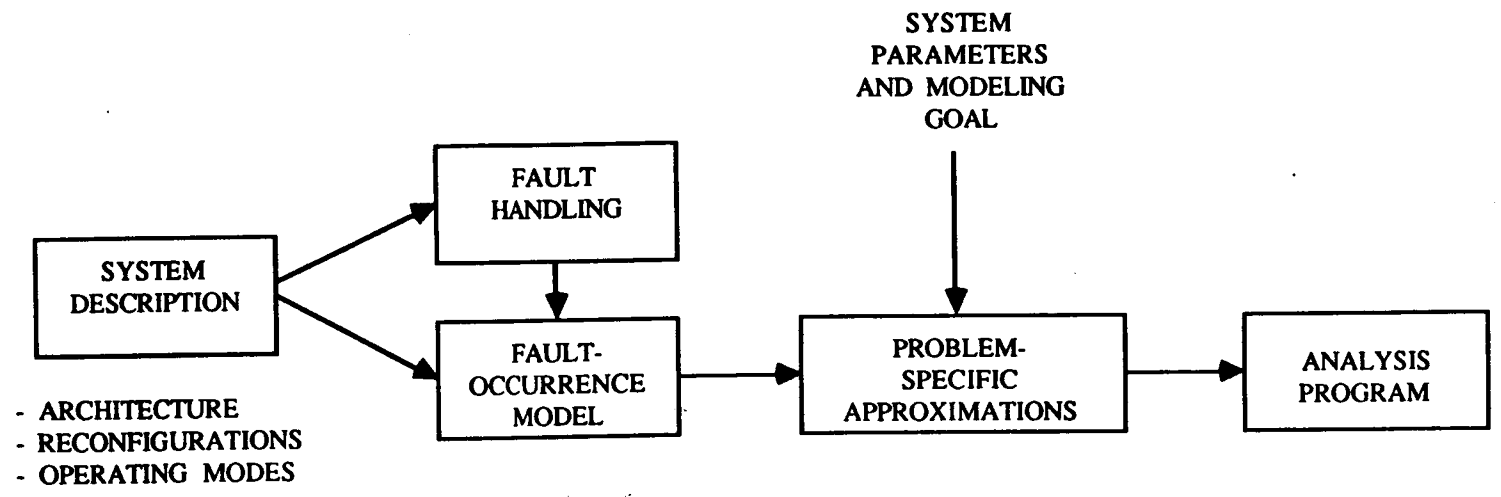

FIGURE 3: ELEMENTS OF RELIABILITY ANALYSIS

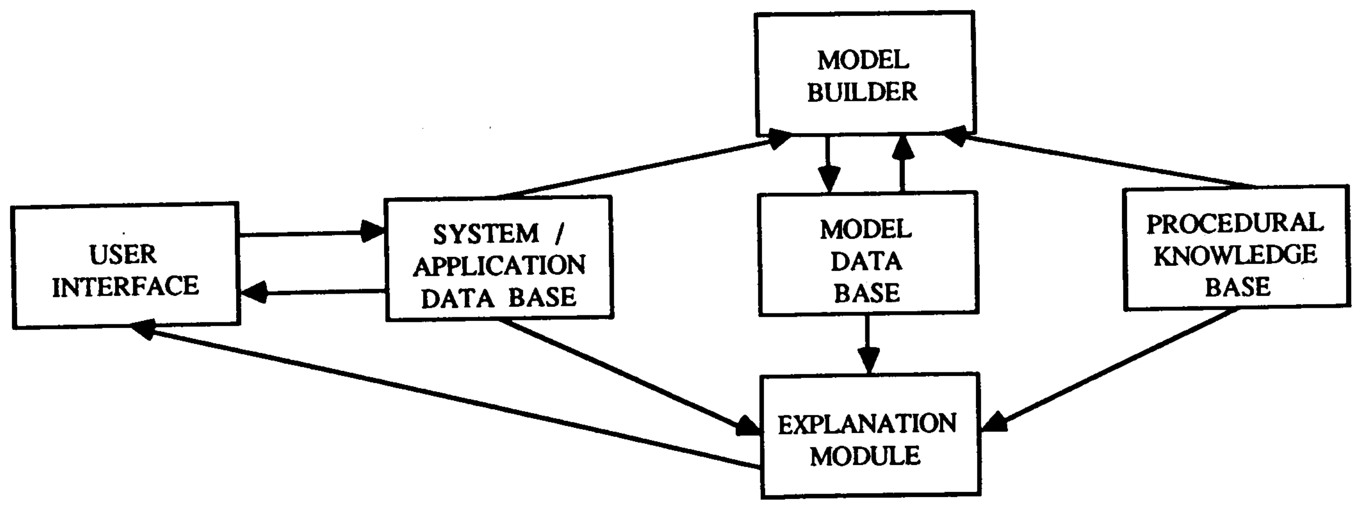

FIGURE 4: CAME PROGRAM BLOCK DIAGRAM

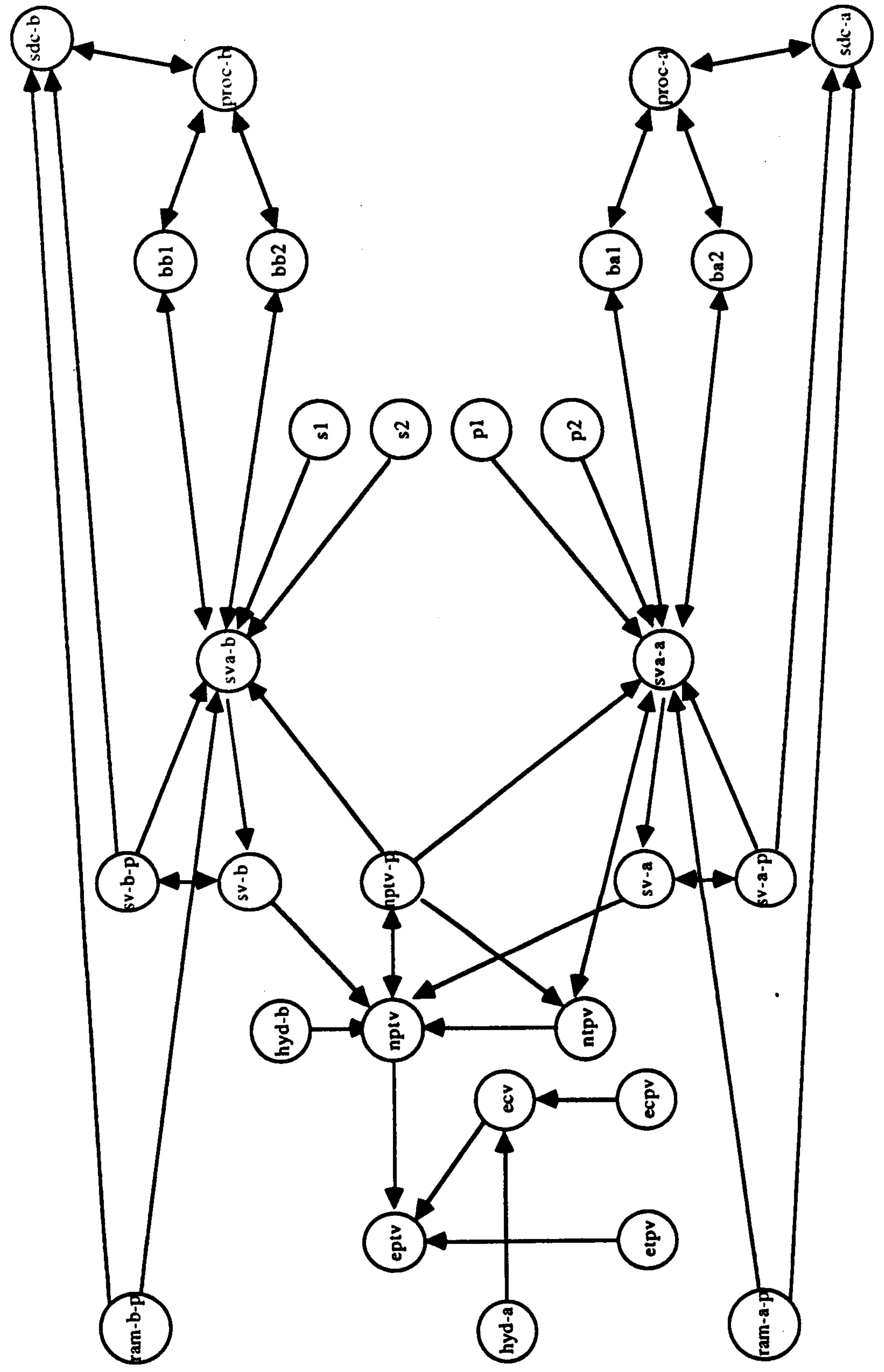

FIGURE 5: INTEGRATED CONTROL SYSTEM: ARCHITECTURE

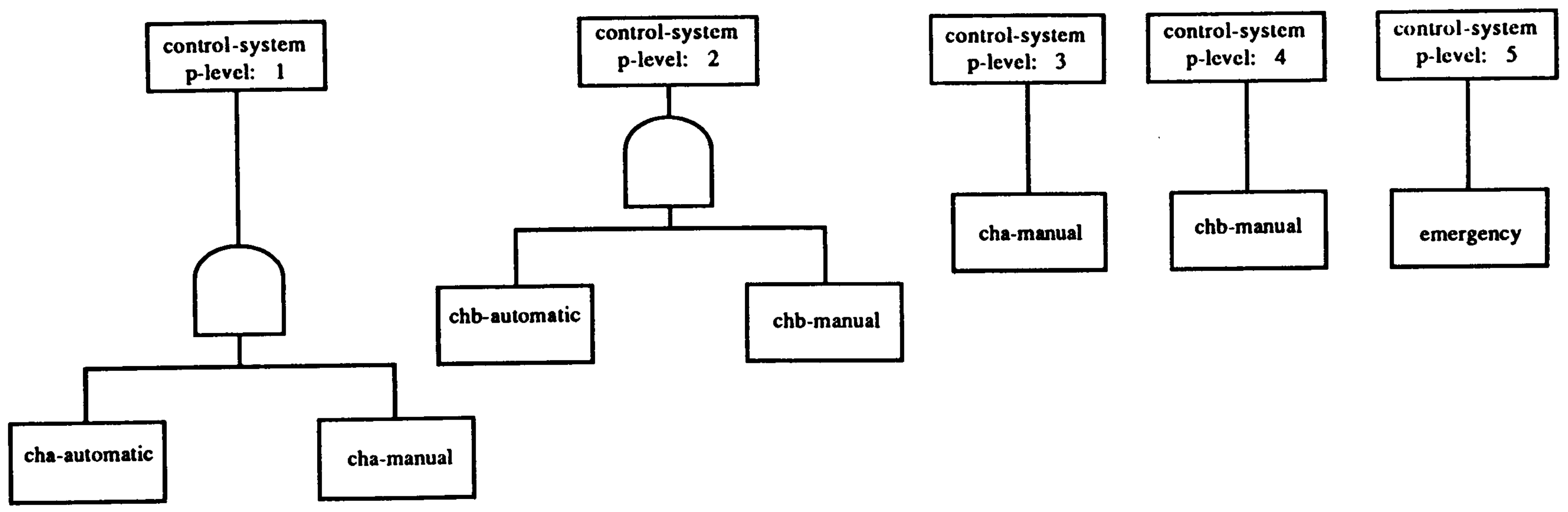
control-system p-level: 1
cha-automatic
cha-manual
control-system p-level: 2
chb-automatic
chb-manual
control-system p-level: 3
cha-manual
control-system p-level: 4
chb-manual
control-system p-level: 5
emergency

FIGURE 6: INTEGRATED CONTROL SYSTEM: OPERATING MODES

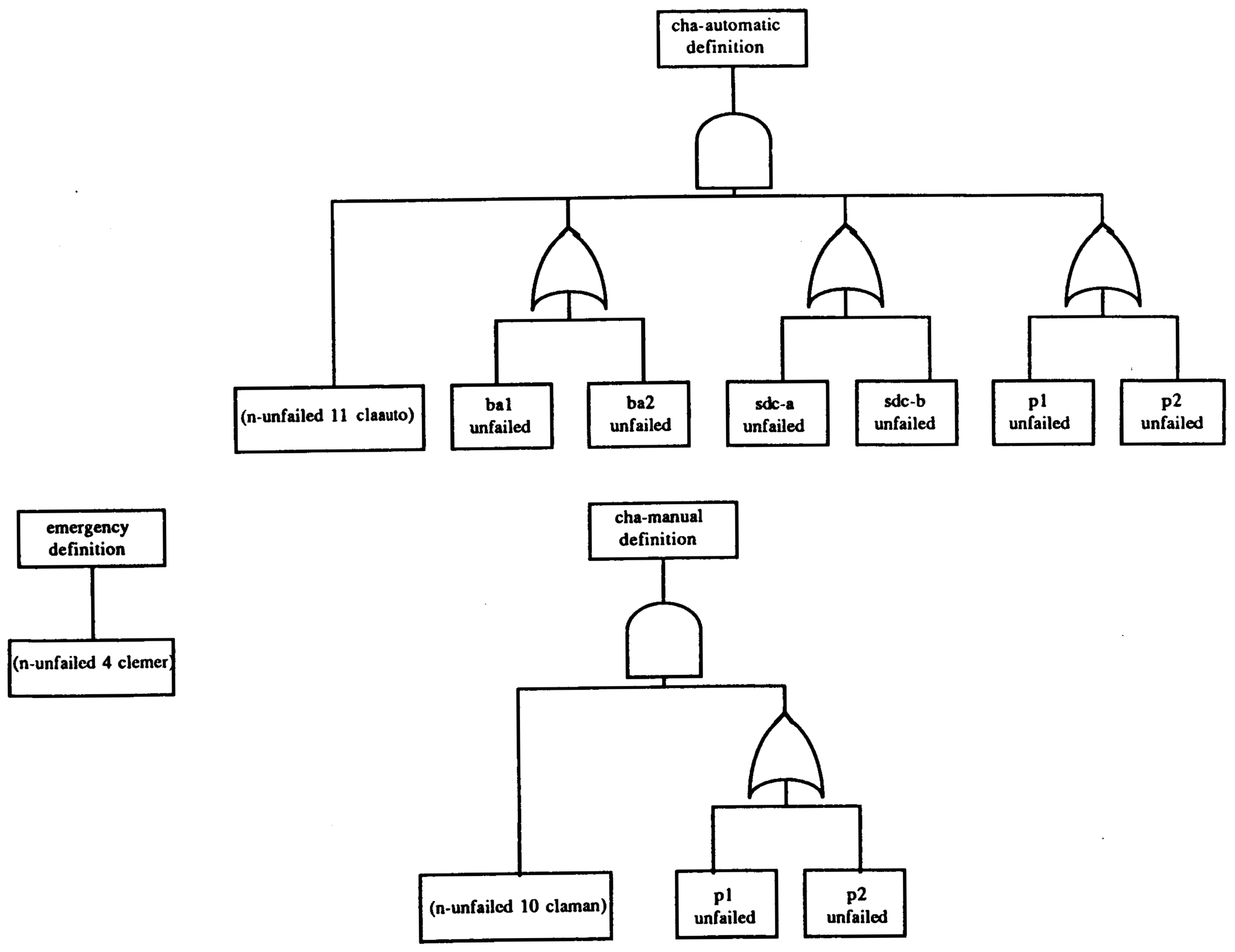

FIGURE 7: INTEGRATED CONTROL SYSTEM: OPERATING MODES (FURTHER SPECIFICATIONS)

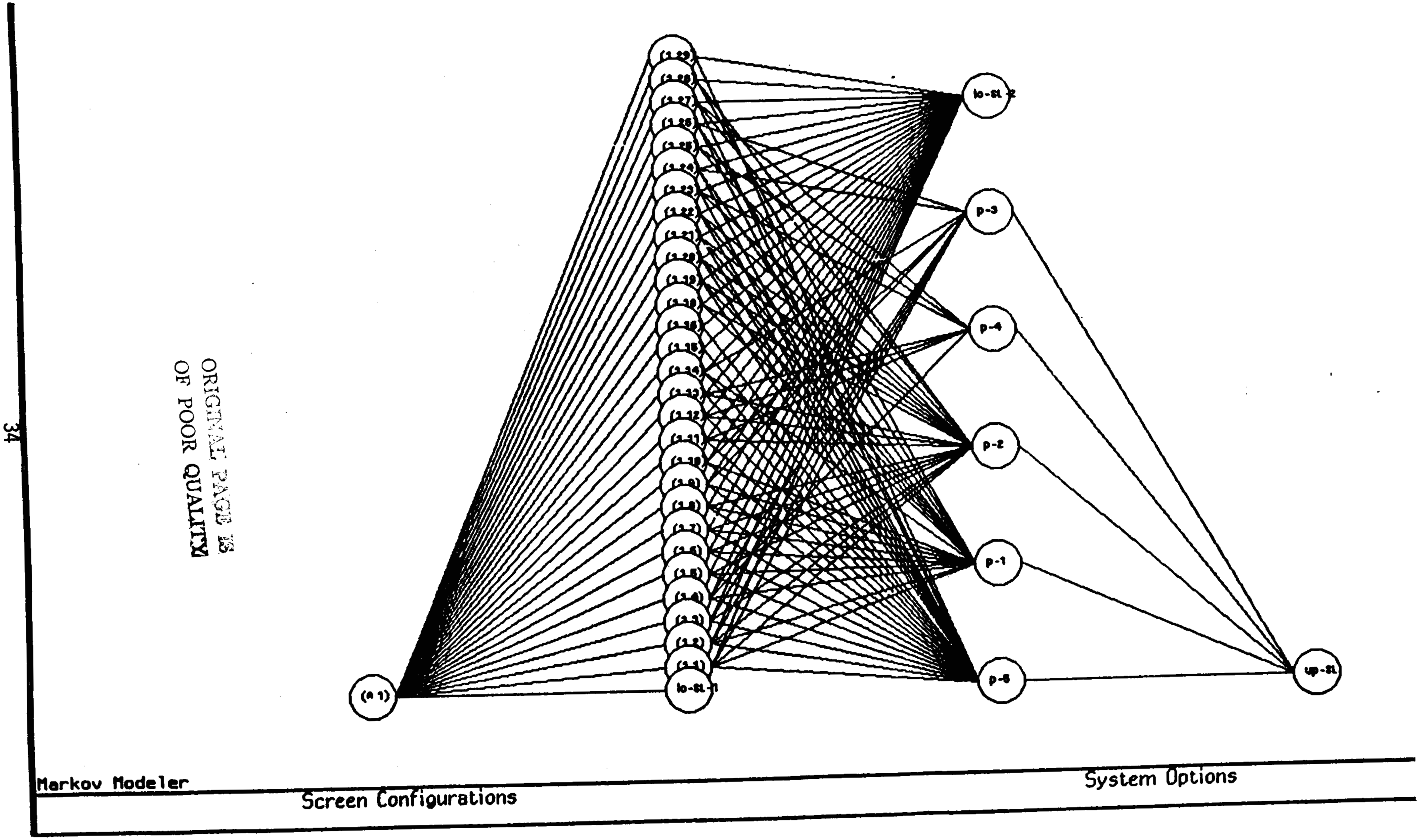

FIGURE 8: INTEGRATED CONTROL SYSTEM: MARKOV MODEL

```
state  (1  6)
SYSTEM:   control-system
system-performance-level:     1
failed components:    ba2
uncovered  failures:
objects at performance level 1:        sdc-a   sdc-b   proc-a   proc-b
        ba1   bb1   bb2   sva-a   sva-b   sv-a   sv-b
        sv-a-pf   sv-b-pf   nptv   nptv-pf   ntpv
        eptv   ecv   s1   s2   p1   p2
        hyd-a   hyd-b   ram-a-pf   ram-b-pf
        ecpv   etpv
objects that are in use:        sdc-a   sdc-b   proc-a   proc-b
        ba1   bb1   bb2   sva-a   sva-b   sv-a   sv-b   sv-a-pf   sv-b-pf   nptv   nptv-pf   ntpv
        eptv   ecv   s1   s2   p1   p2   hyd-a   hyd-b   ram-a-pf   ram-b-pf   ecpv   etpv
reconfigurations:
sources:
        State  (0  1)  because of failure event(s):
               ba2
destinations:
        State  p-1  because of failure event(s):
               sdc-a   sdc-b   proc-b   bb1   bb2   sva-b   sv-b
               sv-b-pf   s1   s2   p1   p2   hyd-b   ram-b-pf   ecpv
        State  p-2  because of failure event(s):
               proc-a   ba1   sva-a   sv-a   sv-a-pf   ntpv   hyd-a   ram-a-pf
        State  p-5  because of failure event(s):
               nptv   nptv-pf   etpv
        State  lo-SL-2  because of failure event(s):
               eptv
```

FIGURE 9: INTEGRATED CONTROL SYSTEM: STATE INSPECTOR EXAMPLE

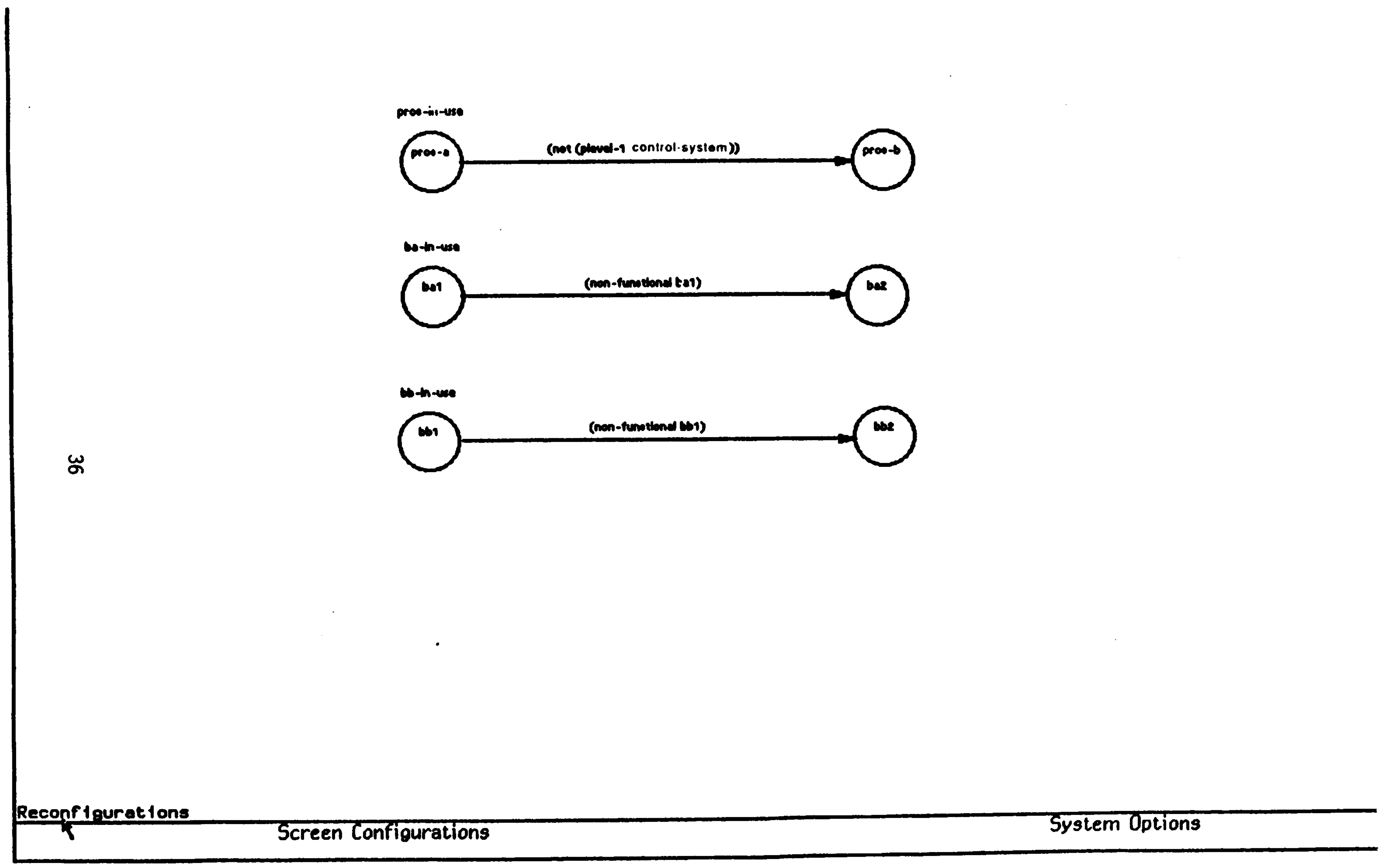

FIGURE 10: INTEGRATED CONTROL SYSTEM: RECONFIGURATIONS

Standard Bibliographic Page

1. Report No. NASA CR-178380	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle On the Next Generation of Reliability Analysis Tools		5. Report Date October 1987
		6. Performing Organization Code
7. Author(s) Philip S. Babcock IV, Frank Leong, and Eli Gai		8. Performing Organization Report No.
9. Performing Organization Name and Address The Charles Stark Draper Laboratory, Inc. 555 Technology Square Cambridge, MA 02139		10. Work Unit No. 505-66-21-01
		11. Contract or Grant No. NAS1-18061
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225		13. Type of Report and Period Covered Contractor Report 10/85 - 2/87
		14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor: C. W. Meissner, Jr.

16. Abstract The current generation of reliability analysis tools has concentrated on means of improving the efficiency of the description and solution of the fault-handling processes and providing a solution algorithm for the full system model. The tools have succeeded in improving user efficiency in these areas to the extent that the problem of constructing the fault-occurrence model is now the major analysis bottleneck.

For the next generation of reliability tools, we propose that techniques be developed to improve the efficiency of the fault-occurrence model generation and input. Further, the goal is to provide an environment that permits a user to provide a top-down design description of the system from which a Markov reliability model is automatically constructed. In this way the user is relieved of the tedious and error-prone process of model construction, permitting an efficient exploration of the design space, and an independent validation of the system's operation is obtained. An additional benefit of automating the model construction process is the opportunity for reducing the specialized knowledge required of a user. Hence, the user need only be an expert in the system he is analyzing; the expertise in reliability analysis techniques is supplied by the analysis tool.

IR&D research at CSDL has shown the feasibility of achieving these goals. Much work is left to do in further automating the fault-occurrence model generating ability and integrating it with the best features of the current generation tools.

17. Key Words Suggested by Author Reliability Analysis, Markov Modeling, Automated Reliability Tools	18. Distribution Statement Unclassified - Unlimited Subject Category 66

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	37	A03

For sale by the National Technical Information Service, Springfield, Virginia 22161