NASA Conference Publication 2130

Validation Methods Research for Fault-Tolerant Avionics and Control Systems - Working Group Meeting II

Proceedings of a working group meeting
held at Langley Research Center
Hampton, Virginia
October 3-4, 1979

NASA

NASA Conference Publication 2130

Validation Methods Research for Fault-Tolerant Avionics and Control Systems - Working Group Meeting II

James W. Gault, Kishor S. Trivedi,
and James B. Clary, Editors
Research Triangle Institute

Proceedings of a working group meeting
held at Langley Research Center
Hampton, Virginia
October 3-4, 1979

NASA
National Aeronautics
and Space Administration

**Scientific and Technical
Information Office**

1980

PREFACE

To effectively address the problems of fault-tolerant avionics and control system validation, NASA-Langley Research Center has conceived and sponsored a series of working group meetings with the objective of identifying and addressing critical issues related to the validation process. The first working group meeting in this series, Working Group I, was held in March 1979.

Working Group I provided a forum for the exchange of ideas on fault-tolerant avionics and control system validation. The state of the art in fault-tolerant computer validation was examined in order to begin the establishment of a framework for future discussions of validation research for fault-tolerant avionics and flight control systems. The results of Working Group I and the evolution of the Avionics Integrated Research Laboratory (AIRLAB) by NASA-Langley Research Center provided impetus for a second working group meeting.

The objective of Working Group II was to identify, beginning with the ideas provided by Working Group I, specific validation tasks which could benefit substantially from the existence of AIRLAB. To provide an initial focus, validation issues specifically related to two fault-tolerant computers currently being designed and developed under the sponsorship of NASA-Langley Research Center, namely, SIFT and FTMP, were considered. Particular validation tasks for these computers were identified at a preliminary Working Group II meeting held at the Research Triangle Institute (RTI) in September. The tasks generated at this meeting served as a starting point for the larger Working Group II meeting held at NASA-Langley Research Center in October.

The activities of Working Group II during the two-day session in October centered around the previously defined validation tasks. These tasks were partitioned into three major categories:

1. Confirmation of System Reliability
2. Fault Processing Verification
3. Fault Processing Characterization

Working Group II attendees evaluated the preliminary proposed tasks in each of these areas and proposed additional tasks.

The Working Group II meeting was conceived and sponsored by personnel at NASA-Langley Research Center, in particular Billy L. Dove and A. O. Lupton.

TABLE OF CONTENTS

PREFACE iii

1.0 INTRODUCTION AND OVERVIEW 1
1.1 Motivation for the Problem 1
1.2 Current Status of the Validation Process 1
1.3 Fault-Tolerant Systems Technology Development 1
1.4 Fault-Tolerant Systems Validation Technology Development 2
1.4.1 Logical Proofs 3
1.4.2 Analytical Models 3
1.4.3 Experimental Testing 5
1.4.4 Fault-Tolerant Systems Validation Technology Development Summary 6
1.5 Report Scope and Organization 7

2.0 TOWARDS A VALIDATION METHODOLOGY FOR FAULT-TOLERANT AVIONICS COMPUTERS 8
2.1 Traditional Methods 8
2.1.1 Reliability Validation of a Simplex System 8
2.1.2 Reliability Validation of Redundant Systems 9
2.1.3 Inadequacy of Traditional Methods 9
2.2 Proposed Validation Methodology 10
2.2.1 Discussion Leading to the Proposed Methodology 10
2.2.2 Details of the Proposed Methodology 13
2.3 Future Work 15

3.0 PRELIMINARY TASKS FOR SIFT/FTMP RELIABILITY VALIDATION 16
3.1 Confirmation of System Reliability 16
3.2 Fault Processing Verification 17
3.3 Fault Processing Characterization 19
3.4 Other Tasks 19

4.0 SUMMARY 20

5.0 REFERENCES 22

APPENDIX I - DEFINITIONS AND REFERENCE CODE 24

APPENDIX II - WORKING GROUP II TASK DESCRIPTIONS 28

APPENDIX III - TASK RATING RESULTS FROM WORKING GROUP II 76

APPENDIX IV - WORKING GROUP II ATTENDEES 77

TABLES 80

FIGURES 92

1.0 INTRODUCTION AND OVERVIEW

1.1 Motivation for the Problem

In 1979 commercial air carriers in the United States paid, on the average, an increase of 70% in their annual fuel bill over the previous year. This statistic highlights the importance of developing energy efficient aircraft. Under the sponsorship of NASA-Langley, the ACEE Energy Efficient Transport Technology effort has been established to develop the technology required to support this effort.

New low-drag aerodynamic structures show great promise for lowering fuel consumption. However, as a consequence of their structure, wing and tail designs show significant increases in loading and decreases in stability that must be alleviated by the introduction of sensors, actuators, and digital electronics to dynamically alter control surfaces in flight. In certain future designs this active control system will be crucial to flight; its use will not be optional and no manual override or backup will be employed. Thus, the development of high integrity, active control systems technology is an integral part of the present direction of the Energy Efficient Transport Technology effort. The validation process must, of necessity, be a part of this effort.

1.2 Current Status of the Validation Process

While a great deal of work has gone on in the area of fault-tolerant system design, the problem of validating ultra-reliable systems is just beginning to be addressed. The current state of the art is given in summary form in Sections 1.4.1 through 1.4.3 of this report.

1.3 Fault-Tolerant Systems Technology Development

The development of active control systems has many facets, and the scope of this report is limited to the development of the technology required to support the fault-tolerant system's aspect of active controls. Much of the work reported takes the even narrower view of technology development for fault-tolerant computer systems. This is a reasonable limitation initially, and it is anticipated that many useful extensions can readily be made later.

The development of fault-tolerant systems technology involves design, assessment, validation, and maintenance of an experience base for candidate systems. NASA-Langley has, for several years, supported the design of two representative fault-tolerant computer systems - SIFT and FTMP. These systems represent distinct approaches to the implementation of a system which must "provide flight crucial functions with a failure probability of less than 10^{-9} at 10 hours."

A prototype version of each of these systems will be delivered to NASA-Langley in 1980 for assessment and validation. The knowledge gained from these systems will be used as an experience base to be retained and disseminated for understanding in the development of the next generation of systems.

It is envisioned that the assessment, validation, and experience base activities will be supported by a specialized facility, the Avionics Integrated Research Laboratory, which has been defined by NASA-Langley and will provide the capability to:

1) develop the technology and methodology required to integrate avionic and control functions for aircraft of the 1990's and beyond,

2) evaluate and study candidate system architectures,

3) validate implementation technologies, and

4) establish a data base of performance, reliability, and experiment statistics.

1.4 Fault-Tolerant Systems Validation Technology Development

One of the most important and challenging aspects of fault-tolerant systems development is the validation process. The validation process comprises the activities required to insure the agreement of the system realization with the system specification. This effort is significant and requires the development of technology in its own right.

Validation is not a new problem. Validation techniques exist and have been applied to many digital electronic avionics and control systems presently in use, such as the F-III redundant Mark II avionics, the B-I redundant avionics, the F-18 quad redundant digital flight control, the F-8 triplex digital fly-by-wire system, and the Space Shuttle quad redundant avionics and control backup system.

Validation activities have traditionally been a part of the digital system life cycle shown in Figure 1.1. Applied to avionics systems (as shown in Figure 1.2), this process includes some very familiar, concrete, and trusted tasks, such as bench tests, "hot-bench" tests, ground tests in the aircraft, and flight tests in experimental configurations.

None of the tasks of the past can be summarily dismissed as inappropriate to the problem at hand, but rather a more precise, systematic, disciplined, and extensive approach must be evolved to incorporate and augment existing techniques. The properties that distinguish ultra-reliable systems validation are summarized in Table 1.1.

The state of the art of validation was discussed as a portion of the agenda of Working Group I (ref. 1), an earlier NASA-Langley-sponsored effort.

The results of that discussion are summarized here as preliminary to the framework of validation presented in Section 2.0 of this report.

This recap is presented in a form consistent with the models of Section 2.0 and, in particular, the discussion which follows is keyed to Figure 2.5 (The Proposed Validation Taxonomy). Table 1.2 shows the three primary categories used in Working Group I to discuss the state of the art of validation and relates them to the three categories used in this report.

1.4.1 Logical Proofs

The theory of proving is being adequately addressed and is not the present limitation. Accurate and formal statements of specifications and environmental assumptions suitable for proof techniques may take weeks to write. Proof techniques are most often applied to the validation of software at the upper level of its hierarchical description. However, some use of proving has been made with hardware when appropriate formal language descriptions exist.

Automatic proof generating methods exist for certain restricted classes of problems. More powerful interactive proving techniques are also available but require highly skilled personnel and a large commitment of computer resources. Work on proof techniques is expected to continue and will be useful as a tool in the validation process.

1.4.2 Analytical Models

The term "analytical models" is used here to define that category of activities in the validation process concerned with analyzing or predicting system performance or reliability. Proof techniques and simulation/emulation may rightfully fit within this definition, but are categorized separately and are specifically excluded from this discussion. Of interest here is the definition of faults (a fault model) and the analysis or estimation of a system's response (a system model) to this fault model.

Work is progressing to unify performance and reliability considerations into a single model, creating what is called a "performability" model. The objective is to estimate a system's ability to perform in the presence of faults. The major enhancement provided by this model is the definition of "ability to perform" in terms of acceptable levels of degradation, rather than the more common, restrictive, and by now unreasonable, pass and fail levels.

The CARE III reliability estimation program takes a more traditional point of view:

> "The object of CARE III is the estimation of reliability for fault-tolerant avionic systems with failure probabilities of less than 10^{-9} at 10 hours."

In general, the problem of estimating system failure rate and confidence interval (stated as: the system has a failure rate of no more than X with a confidence interval of Y) from component failure rates and confidence intervals is unsolved. In addition, the failure probability of less than 10^{-9} is so stringent that everything is important. Assumptions, model approximations, and computational round-off errors all seriously impact the credibility of the results obtained.

CARE III proposes to accommodate a much more realistic fault model than previous programs, including:

1) time-dependent failure rates,
2) design errors, as well as,
3) intermittent and transient faults.

The system model also accommodates a more meaningful set of fault handling mechanisms, including the discrimination of:

1) time from fault occurrence to error occurrence,
2) time from error occurrence to error detection,
3) time from error detection to fault isolation, and
4) time from fault isolation to system recovery.

While recent improvements in system modeling are significant, many difficulties remain. Some of the most important issues in the modeling of fault-tolerant systems with ultra-reliability requirements are:

1. Handling large state spaces which result from the complex systems and fault models.

2. Fault latency - this is an important issue since combinations of faults may be much more damaging than any fault alone. A system's response to faults occurring before recovery from the last fault is complete must be studied.

3. Coverage - it is important to be able to assess the degree to which the fault model is in agreement with observed reality.

4. Unexpected events - some strategy is required for dealing with events not predicted. In these systems unpredicted events are not insignificant.

5. Failure statistics - a continued effort is required to obtain statistics concerning actual component and system failure mechanisms and rates.

6. Fault descriptions - the current understanding and description of faults is at the circuit and logic level. It is important to develop and understand how these low-level fault mechanisms can be faithfully modeled in terms of higher level system behavior.

1.4.3 Experimental Testing

In this subsection, the testing of a physical device and the testing of a simulated or emulated version of the same device is considered. Testing is the single most frequently applied tool of validation for both hardware and software. It is defined as the process of applying a set of inputs (selected to reveal faults belonging to a predefined fault model) to a unit and then comparing the results produced to a reference of the good response. There is no coherent theory of testing. That is, in general, we cannot, for an arbitrarily defined unit and fault model, define precisely how to generate or evaluate test data to insure a fault-free unit. We are left with a great many ad hoc approaches for treating restricted cases. Even when a unit passes a test, we can only make weak inferences concerning the unit's true condition.

The cost of testing depends upon the cost of: 1) generating input test patterns, 2) storing these patterns as references, and 3) the length of time required to run the test. The practical limitation of cost is one of the primary deficiencies of testing, since even relatively simple networks may require enormous numbers of test patterns. "Standard tests for commercial LSI devices rarely result in greater than 95% coverage (with coverage here meaning the percentage of nodes in the circuit that truly change state during the course of the test), because the cost of higher coverage is prohibitive. Similar statements can be made with respect to software testing procedures." Current industrial practice is to acceptance-test components at 3% acceptance quality level (AQL) at 95% confidence for stuck-at faults.

Another serious limitation to testing is our inability to describe faults at an abstract level, while retaining a proper abstraction of their true physical behavior. At the present time, the most frequently applied fault model is the permanent stuck node model. This is not always a useful or accurate model and there is a need to consider a more realistic fault model. This includes the need to identify equivalent classes of faults in order to reduce the large number of cases which need to be considered. The limitation of solid failures is no longer practical. There is a pressing need for a meaningful and tractable model for intermittent and transient behavior.

Physical fault insertion has long been used to provide information and calibration for input test pattern and diagnostic program coverage evaluation. If the trend continues toward higher levels of integration and if physical fault insertion is to continue as an applicable technique, then understanding low-level fault mechanisms at the interface of higher level models is imperative. Simulation/emulation have long been used as tools to attack the problem of evaluating fault coverage capability. Simulation/emulation models typically take much longer to run than similar testing on physical units and, therefore, suffer more severely from the practical limitation of long execution times than other forms of testing.

The most potentially fruitful work is in the area of design for testability. The idea is that the testing problem, in the light of increased complexity and integration, cannot be solved without incorporating testing features into the original design. There are no theoretical results which significantly

impact design and the same is true for testability design. However, in the literature there are a fair number of detailed suggestions and "tricks" that can ease problems if they are conscientiously applied.

In summary, there are some useful techniques for finding test input patterns for reasonably small networks when the fault model used is the solid stuck node type. Testing is widely used, primarily because we have some intuition in its use and because it is a very concrete activity. Testing alone, however, provides a very weak basis for systems validation when ultra-reliability is required.

1.4.4 Fault-Tolerant Systems Validation Technology Development Summary

The preceding discussion presents more questions than it answers, and rightly so, since many of the real issues are being clarified or defined for the very first time. There is little work available that deals with defining faults in a meaningful way. No treatment is offered for verifying what specifications are fault-free, and very little help is available for dealing with design faults. In addition, it is clear how very important software failures are, and yet there is only the smallest beginning being made to develop a model for these failures. Designing with verification in mind may be one of the most fruitful avenues for work, and results have begun to appear in the literature. Effort will be required on many different fronts if one is to field the systems foreseen. The presentation in Section 2.0 creates a model that can be used to plan future developments. This model unifies many of the disjoint concerns briefly discussed here and gives a framework for better understanding.

If the validation process for the next generation of systems is not different in its basic definition or intent, then how will it be different? The answer can be framed at two levels. At the most primary level, the distinction is a consequence of the significant increase in the reliability requirement specification. A typical reliability requirement for a present generation system is:

> a probability of catastrophic failure
> of 10^{-6} at 90 minutes

A typical reliability requirement for the next generation of systems is:

> a probability of catastrophic failure
> of 10^{-10} at 10 hours

A second level of distinction in the complexity of the validation process is a consequence of this increase in the reliability requirement; that is, the realizations of the next generation of systems will be significantly more complex than existing systems. Present systems typically employ from one to four processors in a basically static configuration. The next generation systems

will employ many more processors and dynamic reconfiguration strategies, allowing a wide variety of operational configurations for normal, as well as faulty, conditions.

These two attributes, a demanding reliability specification and the attendant system realization complexity, have a tremendous impact and compound the importance of the validation process. The use of lifetesting is out of the question, since any real failure is an extremely rare event. The use of testing for induced failures as a strategy is also inadequate because no test can be designed for events that are unforeseen. In addition, the criticality of the intended application, passenger carrying commercial aviation, creates a strong desire for system validation at a very high level of confidence. This desire, coupled with the difficulty of the validation task, illuminates the importance of developing technology for fault-tolerant flight crucial digital electronic systems.

1.5 Report Scope and Organization

Working Group I identified general fault-tolerant avionics and control systems validation issues. Working Group II focused on specific validation tasks and established a framework for further research. This document draws upon the raw information produced at these working groups and sets as objectives the presentation of:

1. a general framework for the validation of ultra-reliable fault-tolerant digital electronic systems,

2. a set of specific tasks for the validation of the first representative ultra-reliable fault-tolerant computer systems - SIFT and FTMP, and

3. a set of research tasks to support an ongoing effort in the development of technology for fault-tolerant systems validation.

Section 2.0 reviews the evaluation of the validation process and presents a general framework for validation technology resulting from the Working Group II meeting. This general model is the framework within which the specific tasks for SIFT and FTMP validation are summarized in Section 3.0. Throughout the process of defining and ordering the validation tasks, it was clear that a number of important research projects are required to support this technology. Section 4.0 identifies research tasks in two major categories - those in support of validation and those in support of fault-tolerant computing, and recommends specific efforts for the future development of technology required to support validation of fault-tolerant systems which will be part of the active control structure of energy efficient aircraft of the future. The Appendices of this report include: (I) definitions of terms, (II) task descriptions generated by Working Group II, (III) the results of task ratings made by the working group participants, and (IV) a list of Working Group II attendees.

2.0 TOWARDS A VALIDATION METHODOLOGY FOR FAULT-TOLERANT AVIONICS COMPUTERS

2.1 Traditional Methods

A traditional approach to reliability validation is the lifetesting method in which one takes n statistically identical copies of the system under test (SUT) and terminates the test after r ($1 \leq r \leq n$) systems have failed. Using the accumulated time on test T_r, one can derive a point estimate and confidence intervals for the mean life, or MTTF, of the system. These statistical techniques also allow one to calculate confidence intervals for system reliability for any given mission time. For details of the statistical techniques used in lifetesting, see ref. 2.

It should be clear that the accumulated time on test T_r increases as the reliability, or MTTF, of the SUT increases. For fixed r and n, this implies that the width of the confidence intervals for MTTF, or the reliability of SUT, increases with T_r. In other words, if one desires a fixed width of the confidence interval, one has to increase the number n of systems under test. It follows that the number of systems required to be put under test increases monotonically with the reliability of the system being tested. Furthermore, the validation problem is compounded because the cost of an individual copy of the system also increases remarkably with its reliability. Thus, the cost of validation increases more than proportionately with the reliability of the system under test (see Figure 2.1).

2.1.1 Reliability Validation of a Simplex System

With this information in mind, let us consider the reliability validation of conventional simplex (nonredundant) systems. Such systems are characterized by relatively low levels of reliability and, hence, the cost of validation is within reason. If one assumes that the time to failure of the system is exponentially distributed with the failure rate λ (or MTTF = $1/\lambda$), the system can be modeled as a two-state Markov chain as shown in Figure 2.2. Note that the state labeled 1 implies that the system is working properly, while state 0 indicates the system has malfunctioned. The system is initialized to state 1 and after a random interval of time, ends up in the failure state 0.

For such a conventional simplex system, the validation process consists of only two steps:

1) obtaining a point estimate and confidence intervals for the failure rate λ (or the MTTF/or the system reliability for a specified duration), and

2) testing the assumption of exponentially distributed lifetimes.

Both of these steps are well within the realm of statistical lifetesting techniques.

2.1.2 Reliability Validation of Redundant Systems

Next, let us consider a redundant system designed to provide greater levels of reliability, e.g., a two-unit standby sparing system. One unit is placed into operation, while the other unit is kept in a standby status. For the purpose of reliability analysis, the system can be modeled by a three-state Markov chain as shown in Figure 2.3. State i implies that i (= 0,1,2) units are in proper working order. λ is the failure rate of an individual unit and c is the coverage parameter associated with the reconfiguration mechanism. The starting state of the system is 2 and the failure state is 0. If a fault occurs in state 2 and it is covered, then the system goes to state 1. If a failure occurs in state 1, then the system as a whole ultimately fails. It should be noted that the failure of a single unit does not necessarily imply system failure, and that for a covered fault, the system goes through two states before reaching the failure state. This standby unit feature of a redundant system increases system reliability over a simplex system consisting of only one unit.

Two methods for reliability estimation (or validation) are presently available. One method uses conventional lifetesting techniques which treat the system as a black box, disregarding its internal structure. In particular, the reliability model of Figure 2.3 is not used. Because of the increased reliability of the system, the cost of validation increases (see Figure 2.1).

The second validation method utilizes the reliability model of Figure 2.3. To evaluate system reliability using this model, one needs the values of the failure rate λ of an individual unit and coverage c of the reconfiguration mechanism. These parameters (and their confidence intervals) are to be estimated by conducting a lifetest on these units. (Note the distinction between the lifetest of a unit and the lifetest of a standby sparing system composed of these units. In particular, the cost of the former is likely to be much less than that of the latter.) To estimate the coverage parameter, fault-insertion experiments are used.

2.1.3 Inadequacy of Traditional Methods

To achieve ultra-high reliability, extensive use of standby sparing and automatic reconfiguration is made to increase the probability that a system will pass through many "good" states before finally reaching the failure state. Because of the high level of redundancy, system reliability is pushed to hitherto unachievable levels. However, applying traditional lifetesting techniques implies unreasonably high validation costs (see Figure 2.1). Due to the high cost of system design and construction, it is usually difficult to have available the number of copies needed to obtain statistically significant results from lifetesting. For example, only one copy of each of the SIFT and FTMP systems is under construction. If one were to use a lifetest for reliability validation, then the sample size is $n = 1$. Since the expected time until the first (and only) system failure would be rather long, on the order of 10^8 hours, it is clearly infeasible to conduct such a test from the standpoint of time needed. Furthermore, even if one completed such a test hypothetically, statistical confidence in the results of the test would be negligible due to the small sample size, $n = r = 1$.

2.2 Proposed Validation Methodology

2.2.1 Discussion Leading to the Proposed Methodology

The above discussion clearly implies that the traditional lifetesting approach must be abandoned when validating ultra-high reliability systems. The problem of validating such systems is relatively unexplored, as pointed out by Hillier and Lieberman (ref. 3):

> "Statistical estimation of component [or subsystem] reliability is well in hand, but estimation of system reliability from component data is virtually an unsolved problem."

In searching for an alternative technique to traditional lifetesting, we note that one shortcoming of the traditional method is its "black box" approach which ignores the internal structure of SUT, and is based on experimental testing followed by statistical analysis. With ultra-high reliability systems, one cannot afford to ignore the internal structure of the system. It is believed that a validation methodology of such systems must be based on a judicious combination of experimental testing, analytic modeling, and logical proofs.

Having decided that the internal structure of SUT must be an integral part of the validation process, one has to determine further the level at which the system structure will be considered. Green and Bourne (ref. 4) suggest that the system should be broken down hierarchically until a level is reached where all the necessary measurement data is either available or can be collected. It is felt that an analytical reliability model (e.g., a Markov model) provides just enough detail of the system structure for our purposes. Such models enable one to abstract the states and the state transitions of a complex system into a relatively small and, hence, manageable graph structure. It is also believed that data can be collected to estimate the parameters characterizing such a model. Thus, the selection of a Markov reliability model to drive the validation process is consistent with the suggestions by Green and Bourne. It should be noted that Markov models have been used for system reliability prediction for a long time; however, using such models for reliability validation is believed to be new.

For the purpose of this exposition, let us assume that the Markov model of Figure 2.3 is a proper abstraction of the reliability behavior of the system under test. This model can be characterized by two parameters: 1) the failure rate λ of an individual unit, and 2) the coverage parameter c of the dynamic reconfiguration mechanism. In order to predict system reliability, these two parameters must be known. Therefore, experimental data (presumably from an extensive lifetesting) on the failures of the individual unit must be obtained. Applying statistical techniques, point estimate and confidence intervals on the failure rate λ can then be obtained. Similarly, fault injection simulation experiments must be conducted to make inferences on the coverage parameter c.

More complex systems (and hence more complex models) require more parameters to be estimated. These parameter types can be placed into two classes:

1) Parameters associated with the fault-occurrence behavior of the subsystems, e.g., the failure rate λ in Figure 2.3.

2) Parameters associated with the fault-handling (or fault-processing) behavior of the system, e.g., the coverage parameter c in Figure 2.3.

If one examines a complex fault-tolerant system such as SIFT (ref. 5) or FTMP (ref. 6) (see Figure 2.4 for a reliability model), it appears that fault-handling of the system is characterized by more than one parameter. Fault-handling is composed of phases such as fault-detection, fault-location, and system reconfiguration. To predict system reliability, mean detection time, mean location time, and mean reconfiguration time must first be estimated from experimental data. In addition, coverage associated with each of these phases must also be inferred from experimental data.

Once the above parameters have been estimated within the desired confidence limits, the reliability of the SUT can be estimated using an available package such as, CAST, ARIES, CARSRA, SURF, CARE, CARE II, or CARE III. This, in essence, provides us with a three-step reliability estimation procedure as follows:

1. Estimate parameters (i.e., failure rates) associated with the fault-occurrence behavior of the subsystems.

2. Estimate parameters (e.g., coverage) associated with the fault-handling behavior of the system.

3. Estimate system reliability using the analytical reliability model.

Further examination of the reliability estimation procedure above, however, reveals a major weakness. It is assumed that the analytical reliability model <u>is</u> a proper abstraction of system behavior. This may not necessarily be true. Any validation methodology for ultra-high reliability systems should identify and critically examine all assumptions in the formulation and the solution of the reliability model. These assumptions should either be verified by a logical proof (if possible), or simulation/emulation/physical experiments must be defined to test the validity of the assumptions. In the event that some of these assumptions do not hold to be true in light of experimental evidence, preparations must be made to modify appropriately the reliability model.

In general, three classes of assumptions are made in the formulation and the solution of the reliability model.

1) <u>Structural Assumptions</u> - A reliability model is an abstraction of either a lower level model or the physical system itself. Such a model structure is usually a directed graph consisting of a set of nodes and a set of

arcs. (See Figures 2.2-2.4.) Each node in the model represents a set of states in a lower level model (i.e., a projection). Similarly, each arc in the model represents a set of state transitions in the lower level model. It must be proved that these abstractions are done correctly. Wensley et al. (ref. 5) have outlined a proof procedure to show that a given abstract model is a correct structural abstraction of another lower level model. This proof procedure is expected to become standard and available for use by the computing community as a whole. In addition, one may look toward automata theory for help. Should the answer of this proof procedure be negative, however, then the reliability model must be modified, resulting perhaps in a model with a larger number of states and state transitions.

2) Assumptions Regarding the Fault-Occurrence Behavior - A fault model consists of a postulated class of faults, the corresponding failure rates (which occur as labels of certain arcs in the reliability model), and the description of the stochastic process of each failure class. An attempt must be made not to miss any critical fault types in the fault model. This assumption appears to be neither testable (in a finite fashion), nor provable. More discussion is needed on this issue. Of related importance is whether two or more distinct fault types have been combined into a single fault class in the model. This assumption should be provable by structural methods described under (1) above. The result of expanding one fault class into several is an expansion in the state space and the number of state transitions. The reason for a finer classification may be due to a significant difference in detection, location and reconfiguration times associated with the faults in question.

The second type of assumption in a fault model is the average failure rate of each fault class. Such an assumption can be tested by standard reliability lifetests of the associated subsystem. However, it is presumed that this effort relies on MIL-STD-217B-type expressions, and assumption for failure rate computations and the experience that has gone into the formulation of such a standard. If a check reveals that the computation of failure rates is in error, the change can be easily absorbed into a reliability model since the failure rates appear as parameters.

The third type of assumption in a fault model is the nature of the stochastic failure process. This is usually assumed to be a Poisson process (or equivalently, the assumption of an exponential time-to-failure distribution). Since the failure here refers to the failure of a subsystem (such as a processor), the assumption may be statistically tested if results of a lifetest on the subsystem are available.

3) Assumptions on the Fault-Handling Behavior - As mentioned earlier, mean time and coverage of each phase of the fault-handling process must be estimated. This can be done by conducting fault-injection experiments on the simulation/emulation/physical version of the SUT. Since the value of the coverage parameter is known to have a significant effect on system reliability (ref. 7), coverage must be measured carefully and estimated within a small confidence interval.

Various phases of the recovery process (detection, location, reconfiguration, etc.) have associated distributions since the corresponding times are

random variables. The usual assumption is that of an exponential distribution. This assumption can be verified by means of measurements conducted on the prototype or on fault-injection-type simulation followed by statistical tests. If the above verification results in the unfortunate conclusion of a nonexponential distribution, the reliability model becomes a non-Markovian model.

In summation, the results of validation tests could be a more complex reliability model due to either a growth in the state space or a non-Markovian model. Research efforts must, therefore, be focused on how to deal with these two possibilities. A possible analytical approach to the state growth problem is to look into the method of state aggregation (ref. 8), or the technique of near-complete decomposition (ref. 9). A possible approach to the handling of nonexponential distributions is to use the Coxian methods of stages (ref. 10). The three techniques suggested here are used routinely in queueing theoretic models for computer system performance analysis (refs. 11,12,13,14).

2.2.2 Details of the Proposed Methodology

As discussed in the previous section, the ultra-high reliability requirements of fault-tolerant computers for digital flight control applications preclude the use of traditional lifetests for the purposes of validation. A validation methodology for such systems must be based on a judicious combination of logical proofs, analytical modeling, and experimental testing.

Analytical models enable us to abstract the states and state transitions of a complex system into a relatively small manageable graph structure. Such graph models can be used to predict the aspects of the behavior of the system under study. A logical proof may be used to show that the analytical model is indeed a proper abstraction of the real system. Both the logical proof and the tractable analytical model are based on certain atomic assumptions regarding system behavior. These assumptions must be tested by exercising a simulation/emulation of the system (or a physical prototype, if available).

Thus, logical proofs, analytical models, and experimental testing are three categories of activities that are integral parts of a validation methodology (see Figure 2.5). These three categories apply not only in the validation of reliability, but also in the validation of other system attributes, such as its performance, safety, etc. However, most of our discussion here is restricted to the validation of system reliability.

The logical proof procedures have been collected together under Task I-2 in the list of validation tasks given in table 3.1 of this report. As discussed in the previous section, it needs to be shown that the chosen analytical reliability model is a proper abstraction of the system under consideration. It is possible to give a logical proof of this fact as demonstrated in reference 5. This is described herein as Task I-2. Besides this proof, it is also proposed that a proof of correctness of system design (hardware/software), as well as the proof that the system scheduler performs according to its design, be presented.

Within the activity labeled analytical models, the development, refinement, and solution of reliability models is included and identified as Task I-1 in Section 3.0. Currently, Markov models are in use, but alternative model types, such as Petri nets should be investigated.

The third major category of activity refers to experimental testing on the simulation/emulation/physical version of the fault-tolerant system to be validated. This category is further divided into four subcategories. The first category refers to the validation of the fault-occurrence behavior of the subsystems comprising the system under test. Examples of such subsystems are the individual processors, memories, devices, etc. If adequate lifetests can be conducted, then the failure rate and the distribution of the times to failure of a subsystem may be inferred using statistical techniques. Equivalently, reliable standards such as MIL-STD-217B may be used. No task has been defined along these lines in Section 3.0.

The next subcategory under experimental testing refers to the validation of the fault-handling behavior of the system. This has been adequately covered by Tasks II-7 to II-13 in Section 3.0 of this report. Since such validation experiments require statistical methods in designing such experiments (that is, preprocessing) and in analyzing data collected from the experiment (that is, postprocessing), a separate Task I-3 has been defined to support such statistical activities. Task I-3 provides a bridge between the analytical reliability model (Task I-2) and experimental testing (Task Group II).

The third subcategory under experimental testing refers to validation of the fault-free behavior of the system (Tasks II-1 to II-6). These tests, together with the proof of design correctness (Task I-2), increase our confidence that the system does not suffer from any design/documentation/implementation/realization flaws.

Since we are considering ultra-high reliability systems regarding which almost no practical experience exists, we should expect many "surprises." Exploratory testing (Task Group III) is, therefore, proposed to uncover future surprises.

The set of steps needed in the validation of system reliability are presented in flowchart form in Figure 2.6.

As pointed out earlier, the validation procedure is driven by the reliability model. The reliability model structure is obtained from the system description, which itself is the output of the system design process. Fault occurrence behavior of the subsystems is the second set of inputs needed for the solution of the reliability model. A characterization of this fault occurrence behavior may be inferred from lifetests conducted on the subsystems. Alternatively, standards, such as MIL-STD-217B or past experience, may be used to characterize the fault occurrence behavior. A third set of inputs needed to exercise the reliability model is the characterization of the fault handling behavior of the system. This involves conducting fault-injection type experiments and analysis of resulting data using statistical techniques. The charac-

terization of the fault-handling behavior encompasses estimation of coverage and mean times associated with various phases of the fault-handling process (e.g., detection, isolation, reconfiguration, etc.).

Since the reliability model is an abstraction of the dynamic behavior of the system, it is extremely desirable to present a proof that the model is a proper abstraction of the system behavior.

The postulated fault model may not have covered all possible fault types. Exploratory testing is needed to uncover any surprises and to observe system response to inputs outside the design envelope.

Once the three sets of inputs are available for the reliability model, it can be evaluated numerically using available reliability analysis packages, such as CARE III, when it becomes available. The resulting reliability prediction needs to be checked against the design requirements. If the predicted reliability is found working, a system redesign needs to be undertaken.

2.3 Future Work

From the discussion in Section 2.2, three topics on which future work is needed are clearly evident.

(A) Investigation of reliability models other than the classical Markovian models.

(B) Solution of reliability models with a large state space.

(C) Dealing with nonexponential distribution of times to failure and similarly nonexponential distributions in the phases of fault-handling process (e.g., detection time, location time, etc.).

In addition, the following topics need further research.

(D) Most Markovian models assume that successive events are independent. Methods of testing this assumption and methods of modifying the reliability model, in case of dependence, are needed.

(E) Quantitative methods of software reliability estimation and prediction are needed.

(F) A validation methodology that not only addresses the question of reliability validation, but also encompasses the validation of safety, performance, and economics of the system is needed (refs. 15,16).

Further discussion of these topics is presented in Section 4.0 of this report.

3.0 PRELIMINARY TASKS FOR SIFT/FTMP RELIABILITY VALIDATION

A primary objective of Working Group II was to identify specific tasks which should be conducted in support of fault-tolerant computer validation research. In order to provide a focus for these efforts, SIFT and FTMP were used as example fault-tolerant computers (refs. 17,18). This section describes the set of specific tasks which were identified by working group participants. No claims are made as to the sufficiency of these tasks. However, the need for these tasks was affirmed by one or more participants of Working Group II.

The proposed tasks have been classified into four major categories. These are:

1. Confirmation of System Reliability
2. Fault Processing Verification
3. Fault Processing Characterization
4. Other Tasks

The following sections of this report describe the particular objectives of the proposed tasks in these categories and briefly describe the tasks proposed. These tasks are related to the validation techniques discussed in Section 2.0, Figure 2.5.

3.1 Confirmation of System Reliability

The ultra-high reliability requirements of fault-tolerant computers for digital flight control applications preclude the use of traditional lifetests for the purposes of validation. The method of validation must be based on a critical examination of models for reliability prediction. In particular, an attempt should be made to examine all the assumptions made in the formulation and the solution of the model, experiments must be designed to test the validity of the assumptions wherever possible (or a proof must be given, if possible), and finally, preparations must be made to change the reliability model if the initial assumptions are negated by experimental results.

Three major aspects of a reliability model are identified and treated separately:

1) Reliability Model Structure

2) Fault Model (types of failure, associated failure rates and distributions)

3) Fault Handling Behavior (coverage, detection location, reconfiguration times, etc.)

Verification that the reliability model structure is a proper abstraction of the actual system may be performed using a combination of logical proof and experiment proof. The fault processing verification subgroup concentrates on experiments to verify and parameterize the assumptions in the fault handling behavior for a fixed (given) fault model. The fault characterization group attempts to identify new fault classes and, hence, extend the fault model.

Thus, all the three classes of tasks (represented by three groups) are strongly related. Figure 3.1 summarizes the proposed verification process.

As shown in Figure 3.1, the reliability prediction is based on a reliability model (e.g., a Markov model). The formulation and solution of this model is clearly the first step, identified as Task I-1 in Table 3.1. A reliability theorist for model formulation and a numerical analyst for model solution should be available.

Since model predictions are based on assumptions regarding system behavior, these assumptions must be critically examined for validation.

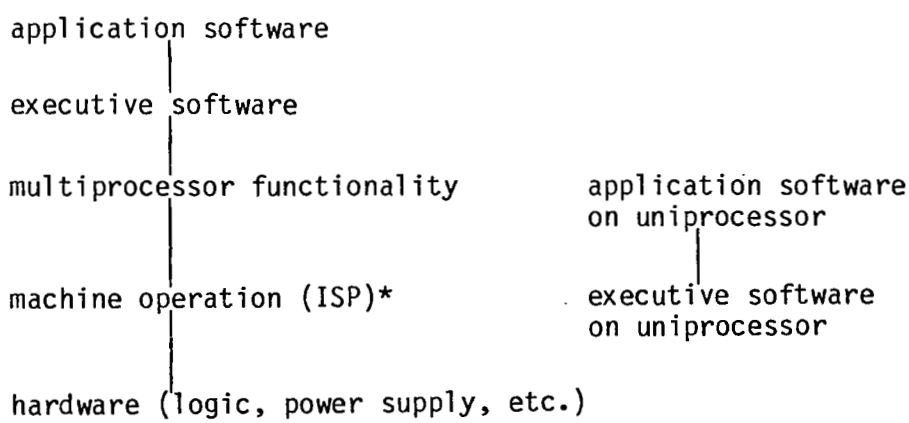
The structure of the reliability model will be confirmed by a proof which will show that the actual system is a proper refinement of the reliability model (Task I-2). The complexity of the system under consideration will dictate that the above proof be structured in the form of a hierarchy of proofs.

The assumptions regarding the fault-handling behavior will be validated by conducting experiments on either the actual system or on an emulation of the system. Statistical design of the experiments and analysis of the resulting experimental data will be performed in Task I-3. The resulting data may require a change in the reliability model and subsequently a new prediction of system reliability.

The tasks summarized in Table 3.1 have been recommended by Working Group II attendees for validating ultra-reliable fault-tolerant computers, such as SIFT and FTMP. These particular tasks address confirmation of system reliability through: 1) reliability modeling structures, 2) fault modeling, and 3) fault handling behavior analysis.

3.2 Fault Processing Verification

This section describes the experiments designed for fault processing verification (table 3.2). The experiments identified are based on the following system classifications:



* Instruction Set Processor

The correctness of each of the sections mentioned above (e.g., application software, executive, . . .) can be investigated by testing and formal procedures. An ideal combination of testing and formal procedure is to prove the design and test the implementation. At the present time, proof of correctness of a design is in its infancy; testing for design and implementation verification and also specification measures is suggested.

There are thirteen experiments which have been identified to test the system. These thirteen experiments fall into two classes:

1. Functional Testing: The idea behind this set of tests is to exercise each integral part of the system (e.g., single processor, single processor executive routine, etc.). This will provide a certain degree of confidence that the system is performing its basic functional operations correctly. This set of tests is structured to reveal design errors (e.g., wrong specification), physical faults (e.g., short pins), etc.

2. Fault Processing Tests: This set of tests is designed to exercise the fault handling capability of the system within its design objectives. There are two sources that can expose the system (or different parts of the system) to error:

   a. hardware faults - It is proposed that faults be injected at the following hardware levels:

      - single stuck pins
      - stuck logic
      - whole chip
      - common mode
      - power supply
      - clock

      The type of faults injected in the first three levels are:

      - solid
      - intermittent
      - transient
      - design error

   b. software design errors - These types of errors include:

      - stress
      - time
      - inconsistent data
      - wrong data

The first six experiments explained in this section belong to the functionality testing and the next experiments belong to the fault processing verification area.

3.3 Fault Processing Characterization

Exploratory testing is required in order to establish the bounds within which the system can possibly operate. It is not expected that the system will tolerate all of the test conditions of this class. On the contrary, the system will be frequently driven into anomalous conditions. It is characteristic of the tasks enumerated in this section (table 3.3) that provision must be made for the machine or system under test to be reinitialized after a nonsurvivable event, and for a data file to be off-loaded from its memory into a facility data bank. The data files would contain sequences of syndromes perceived by the system during the course of this testing, and the concomitant configuration changes. Their value would be to exhibit the consequences of the test conditions as seen at the various abstract levels at which syndrome data is available.

Another characteristic of these tasks is the ability to modulate the intensity or severity of the tests in some respect. This is desirable in order to permit some quantification of the vulnerability of the unit under test. Metaphorically, this would yield a logical shmoo plot for the probable operating region.

The test results must be carefully scrutinized in order to detect lapses in the design, either because the design may be too fragile, or because the design intent is not met. The latter situation corresponds to what has been referred to as a "surprise." It is easier to find surprises when the system does not otherwise fail. Here instead, the system consistently fails, and surprises must be searched for by winnowing postmortem data. Excessive fragility, meanwhile, may feed back to the design of future systems.

It may be noted that most of the tasks of this class closely resemble other tasks in which the system recovery hypothesis is being substantiated. It is believed, however, that the diversity in objectives between such other tasks and these tasks sufficiently warrants the separate categories. The probing nature of tasks in this category allows degrees of freedom not useful in the other, and which might therefore be overlooked if the two categories were to be combined for convenience. It is also true that these tests need not be as exhaustive or as accurate as the others, for a rough characterization of the shmoo is quite adequate.

3.4 Other Tasks

During the course of Working Group II, several tasks were proposed which do not clearly fit into either of the first three categories. In some cases, these tasks spanned more than one of the previously defined categories. In other cases, the recommended tasks deal with indirectly related, but highly relevant, areas such as instrumentation. These tasks are summarized in Table 3.4.

4.0 SUMMARY

The preceding chapters of this report establish the importance, breadth, and difficulty of the systems validation process. For several years, NASA-Langley Research Center has provided leadership and funding in an ongoing effort to develop the technology required to support this process. The working group meetings reported here are the most recent steps in this development effort. The challenge of validating the next generation of ultra-reliable systems can best be met if those with relevant experience and understanding of validation of present systems can be used as resources to address the questions:

1) What is the state of the art in systems validation?
2) What are the important unanswered questions in systems validation?
3) How can we most effectively proceed?

While it is clear that this diverse community of present experts is a powerful asset, it is not clear how they can best be directed to produce a useful product. The working group format was selected as the vehicle most likely to utilize this asset effectively.

The working group activity, along with individual follow-up effort, has produced significant contributions to the development of systems validation technology. The most important of these contributions are reported in this document. They are:

1. The identification, criticism, review, and refinement of a validation framework and reliability validation procedure (Section 2.0).

2. The identification of tasks and facilities required to validate the ultra-reliable specimen, fault-tolerant computers, SIFT and FTMP (Section 3.0 and Appendix II). A significant number of these tasks were reviewed by the working group attendees (Appendix III).

The taxonomy of the validation process with its three primary classes of activity, proof methods; analytic methods; and experimental methods (Figure 2.5), along with the comprehensive view of the reliability validation process (Figure 2.6) are important contributions. These models are a basis for both present understanding and future planning efforts. It is significant that a diverse group of professionals has had the opportunity to see and react to this model, thus providing an important form of peer review. This review process is one of many required to substantiate the validity of these models. In addition, a candidate set of specific experiments (23 in number) have been reviewed and evaluated; and while no formal vote of confidence was taken, the credibility and importance of these experiments is elevated by their exposure to review and informal assessment. The candidate experiments stimulated additional tasks identified by the participants. The total collection of tasks provides a substantial and reasonable basis for future planning. They are the primary basis for the recommendations made in the next subsection.

The development of a validation technology is an ongoing, never ending, activity and concepts requiring future development surfaced in abundance during the working group. It became apparent that even the most mundane and well-understood validation tasks often depend upon assumptions and estimations that need further research effort. Special attention was given by many people to the identification of specific research projects required to support future validation technology development. The proposed efforts vary a great deal in their importance and level of effort. Many of the tasks focus on fault-tolerant computing technology, while others address validation technology; still others are a mixture of the two. Any editing activity by necessity applies the bias of the editor. The original, unedited task recommendations are, therefore, given in Appendix II so that they may be preserved beyond the abstractions necessary for this report.

5.0 REFERENCES

1. "Validation Methods for Fault-Tolerant Avionics and Control Systems," Working Group Meeting No. 1, 12-14 March 1979, Sponsored by NASA-Langley Research Center, Hampton, VA.

2. Bain, L. J., Statistical Analysis of Reliability and Life-Testing Models. New York, NY: Marcel Dekker, Inc., 1978.

3. Hillier, F. S. and G. J. Lieberman, Operations Research. San Francisco, CA: Holden-Day, Inc., 1974.

4. Green, A. E. and A. J. Bourne, Reliability Technology. New York, NY: Wiley-Interscience, 1972.

5. Wensley, J. et al., "SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control," Proceedings of the IEEE, October 1978, pp. 1240-1255.

6. Hopkins, A L. et al., "FTMP - A Highly Reliable Fault-Tolerant Multiprocess for Aircraft," Proceedings of the IEEE, October 1978, pp. 1221-1239.

7. Arnold, T. F., "The Concept of Coverage and Its Effect on the Reliability Model of a Repairable System," IEEE Transactions on Computers. Vol. C-22, March 1973, pp. 251-254.

8. Chandy, K. M. and C. H. Sauer, "Approximate Methods for Analyzing Queueing Network Models of Computer Systems," ACM Computing Surveys, Vol. 10, No. 3 (September 1978), pp. 281-318.

9. Courtois, P. J., Decomposability: Queueing and Computer System Applications. New York, NY: Academic Press, 1977.

10. Cox, D. R., "The Use of Complex Probabilities in the Theory of Stochastic Processes," Proc. Cambridge Philosophical Society, Vol. 51 (1955), pp. 313-319.

11. Chandy, K. M. and R. T. Yeh, Current Trends in Programming Methodology, Vol. III: Software Modeling. Englewood Cliffs, New Jersey: Prentice-Hall, 1978.

12. Kleinrock, L., Queueing Systems. New York, NY: Wiley-Interscience, 2 volumes (1975, 1976).

13. Kobayashi, H., Modeling and Analysis. Reading, MA: Addisson-Wesley, 1978.

14. Trivedi, K. S., "Analytic Modeling of Computer Systems," IEEE Computer, October 1978.

15. Trivedi, K. S., "Designing Linear Storage Hierarchies So As To Maximize Reliability Subject to Cost and Performance Constraints," Proceedings 1980 International Symposium on Computer Architecture, La Baule, France.

16. Meyer, J., "Computation-Based Reliability Analysis, IEEE Transactions on Computers, Vol. C-25, No. 6, June 1976, pp. 578-584.

17. Hopkins, A. L., et al., "FTMP - A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft," Proceedings of the IEEE, pp. 1221-1239, October 1978.

18. Wensley, J., et al., "SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control," Proceedings of the IEEE, pp. 1240-1255, October 1978.

APPENDIX I - DEFINITIONS AND REFERENCE CODE

REFERENCE CODE

Note: The following is a preliminary, nonexhaustive collection of terms which relate to fault-tolerant computer validation. Both hardware-oriented and software-oriented definitions are included for the present time.

IEEE Std. 100-1972	IEEE Standard Dictionary of Electrical and Electronic Terms.
IEEE/FTC	Interim IEEE Technical Committee on Fault-Tolerant Computing Dictionary of Terms.
SET	"Software Engineering Terminology" - Draft, 23 March 1978 by R. Poston and H. Hecht - Terminology Task Group Subcommittee on Software Engineering Standards - Technical Committee on Software Engineering, IEEE Computer Society.
DACS	"Data and Analysis Center for Software (DACS) Glossary - A Bibliography of Software Engineering Terms," Compiled by Ms. Shirley Gloss-Soler, Rome Air Development Center/ISISI, Griffiss AFB, N.Y.
SPS	Structured Programming Series, Vol. 15, Validation and Verification Study, R. L. Smith, May 1975, RADC-TR-74-300.

DEFINITIONS

Correctness Proof	Technique of proving mathematically that a given program is correct with a given set of specifications. The process can be accomplished by manual methods or by program verifiers requiring manual interaction. (SPS)
Error (Hardware Genesis)	Any discrepancy between a computed, observed, or measured quantity and the time specified, or theoretically correct value or condition. (IEEE Std. 100-1972)

Error (Software Genesis)	An error is an action which results in software containing a fault. The act of making an error includes omission or misinterpretation of user requirements in the software subsystem specification. Incorrect translation or omission of a requirement on the design specification and programming errors. (SET)
Failure	The termination of the ability of an item to perform its required function. (IEEE Std. 100-1972)
Fault (Hardware Genesis)	A physical condition that causes a device, component, or element to fail to perform in a required manner; for example, a short-circuit or a broken wire. (IEEE Std. 100-1972)
Fault (Software Genesis)	A fault is a manifestation of an error in program code. The fault is evident when entry of some input data results in the program failing to perform the required function. Note: fault and bug are the same thing. (SET)
Fault Tolerance	The capacity of a computer, subsystem, or program to withstand the effects of internal faults; the number of errors producing faults a computer, subsystem, or program can endure before normal functional capability is impaired. (IEEE/FTC)
Intermittent Fault	A temporary fault. (IEEE Std. 100-1972)
Stuck Fault/Stuck Failure	A failure in which a digital signal is permanently held in one of its binary states. (IEEE Std. 100-1972)

Validation	The process of determining whether executing the system (i.e., software, hardware, user procedures, personnel in a user environment) causes any operational difficulties. The process includes ensuring that specific program functions meet their requirements and specifications. Validation also includes the prevention, detection, diagnosis recovery and correction of errors. Editorial Comment: Validation is more difficult than the verification process since it involves questions of the completeness of the specification and environment information. There are both manual- and computer-based validation techniques. (SET)
Verification	Computer program verification is the iterative process of determining whether or not the product of each step of the computer program acquisition process fulfills all requirements levied by the previous step. These steps are system specification verification, requirements verification, specification verification and code verification (SET)

NOTE TO READER: There is a lack of consistency in interpretations of the terms "failure," "fault," and "error." Consider the following:

(1) that a failure is an event,
(2) that a fault is a condition (or state), and
(3) that an error is a datum.

Then the following statements apply to both hardware and software:

A failure is the event when something causes a device, component, system, algorithm, etc. to change its state from one in which it performs its intended function to one in which it does not. The something which causes the change may or may not be known. After the failure, the device, component, etc. is called a failed or faulty device, component, etc. Any higher level system of devices, etc., which cannot perform its function because a subdevice, etc. is failed, is also called failed or faulty.

A fault is the particular condition or flaw in a failed device, etc. which differentiates it from its unfailed state.

> When the function or output of a device, etc. differs from its intended function or output, that difference is called the error. In data processing systems, error means bad or wrong data. An error is all that can be detected internally to a computing system. A higher level system, which contains a failed device, etc. emitting errors yet continues to perform its function, is said to be fault tolerant. An accumulation of errors may well be the cause of a failure of a higher level system.

Thus, a physical device fails when it "breaks down." Thereafter it contains a fault. A system designer or software programmer can create a design or software containing a fault. In this sense the designer or programmer, not the design or software, failed. A fault may or may not be active; when it is, one or more errors result. A fault is latent, transient, intermittent or permanent dependent upon the manner in which it generates errors. A software bug may not surface until some time after a system has been in operation; i.e., it may be latent. A bug may cause a data error only occasionally in response to specific, infrequent input data patterns and, thus, may appear intermittent. Customarily, a software bug is regarded as a permanent fault, remaining in the system even after the moment of its creation by a programmer. However, it is possible for a bug, having given rise to a data error, to disappear from an operational system, in which case it appears as a transient. The resulting bad data may or may not be attenuated in further processing.

APPENDIX II - WORKING GROUP II TASK DESCRIPTIONS

CONFIRMATION OF SYSTEM RELIABILITY

Tasks I-1 through I-3

Proposed by
Preliminary Working Group II Participants

J. Goldberg (Chairman)
W. Carter
K. Trivedi
R. Alberts

Tasks I-4 through I-11

Proposed by
Working Group II Participants As Indicated

TASK I-1

Title: Reliability Analysis

Objectives:

1. Develop and refine mathematical models of system reliability.
2. Estimate reliability characteristics using assumed and experimentally-derived failure statistics.

Procedure:

A mathematical model of system reliability for a subject computer will be acquired or developed. The model should comprehend all intended fault-handling behavior at an appropriate level of abstraction, and it should be tractable for numerical evaluation. The faults assumed should be modeled by type (e.g., permanent, transient, intermittent), distribution and rate. The fault handling behavior should include fault detection time, fault location time and reconfiguration time, and other appropriate characteristics, such as the coverages associated with the three phases. A set of system models and fault models may be needed to reflect varying trade-offs of fidelity and computational tractability.

The primary purpose of the model is to predict the reliability of the given system for a range of assumed configurations (e.g., initial number of processors) and for a range of assumed fault levels, in other words, to support a claim for mission reliability. As a means to this end, the model will be used for two secondary purposes:

1. specification of data to be obtained from experiments on the actual computer (or an emulation of it) which will be used to improve the quality of assumptions on fault occurrences and fault-handling characteristics of the system,

2. definition of requirements on the design of the computer, in a form that allows proof that the design is consistent with the reliability model.

The results of the tasks on Proof of Correctness and Experimental Validation of System Fault Handling will be analyzed to determine if the structure of the model is a true representation of the system. If it is not, then the model may need to be revised. The results of the experimental tests will be analyzed by the Data Analysis Task (Task I-3) in order to obtain more realistic characterizations of faults and system responses. New reliability predictions will be generated on the basis of these characterizations.

Facilities:

1. Computer Support
   a. Interactive computer system to support model development, including specification and programming (e.g., TOPS20).
   b. Powerful scientific computer to support model evaluation (with high speed and high accuracy) (e.g., CYBER, DEC 10).
2. Software Support
   a. Existing reliability analysis packages, e.g., CARE III.
   b. Program development environment.
   c. Statistical analysis/design.
   d. Logical analysis (proof checker, theorem proving).
   e. Logic design analysis (simulator, test generator).

Personnel:

Reliability Theorist
Numerical Analyst-Programmer

Level of Effort:

1 @ 1/2-time continuing
1 @ full-time continuing

Priority: Highest

TASK I-2

Title: Confirmation and Use of Design Proofs

Objective:

1. Maintain the integrity of design-correctness proofs through system modifications and proof revision.
2. Use results of experimental test on the subject computer to confirm assumptions made in the proof.
3. Use the proof to guide experimental tests.

Procedure:

A formal proof of the correctness of the design of the subject computer will be acquired. This proof will likely have the form of a hierarchy, the top level of the hierarchy being the validation of the reliability model structure. Efforts will be applied, to the extent practical, to complete the proof within

its intended scope. The proof will be revised to reflect changes in the system design.

Test results will be analyzed to confirm assumptions made in the proof, such as the functions performed at the lowest level of the proof (e.g., machine instructions) and timing characteristics of scheduling, synchronizations and reconfiguration.

The proof will be analyzed to help plan experimental tests. For example, classes of faults will be distinguished that are equivalent with respect to some system state, in order to economize on the number of tests needed to cover system fault behavior.

Facilities:

1. Computer Support
   a. A powerful symbol-manipulation computer (e.g., DEC 10).

2. Software Support
   a. Proof-checking tools (high priority).
   b. Proof-of-correctness tools (lower priority).

Personnel:

Logician - Program-Correctness Theorists
- One Senior
- One Junior

Level of Effort:

Full-time - two years each

Priority: Highest

TASK I-3

Title: Design of Experiments and Analysis of Experimental Data

Objective:

Estimate parameters and distributions of various aspects of the fault-handling behavior. The parameters of interest are:

1. Estimate (for those faults that are covered)
   a. mean fault detection time,
   b. mean fault location time,
   c. mean reconfiguration time.

2. Distributions of the above.

3. Estimate
   a. fault detection coverage,
   b. fault location coverage,
   c. reconfiguration coverage.

4. Suggest changes to the structure on the parameterization of the reliability model according to the results of the data analysis.

Procedure:

1. Supply experimental designs to the fault processing verification group, including the number and type (factorial, block, etc.) of experiments needed, and the set of data to be obtained from each experiment.

2. After the data is received from the fault processing verification group, use statistical procedures to perform the above three objectives. Objective 1 is a mean-value estimation problem. Confidence intervals should be obtained using standard statistical methods. Objective 3 is the problem of estimating proportions. Once again, standard statistical methods are available for this purpose. Objective 2 is the problem of functional estimation. The usual hypothesis that the respective distributions are exponential can be tested using statistical methods such as the Kolmogorov-Smirnov test.

3. Evaluate validity of assumptions made in the construction of the model, based on the results of the statistical tests.

Facilities:

1. Hardware - A mainframe computer is needed for carrying out the data analysis.

2. Software - Standard statistical packages are needed:
   - Statistical tables of chi-square, normal, student-t and other distributions are needed.

Personnel:

One Statistician
One Team Member of the fault-processing verification group

Level of Effort:

Statistician at 1/2-time for one year
Fault-tolerant System Designer at full-time for one to two years

Priority: Highest

TASK DESCRIPTION WORK SHEET

Participant's Name John M. Myers

Task Number: I-4

Task Title: Alternative Modeling Techniques

Problem: Present modeling of computer reliability relies primarily on state models (Markov for stochastic or combination for deterministic behavior). State models have the following shortcomings:

a) No provision for concurrent operation of spatially separated subsystems; and hence, an artificial large increase in a "state space."

b) Lack of clear exposition of the tie between a link between states and identifiable structural features.

Discussion: Petri-net models can portray concurrent operation and relate function to structure. Until recently, they have not been mathematically tractable. Recent advances in tractability are demonstrated for deterministic modeling in the analysis of the FTMP clock network. The use of nets as a structure on which to calculate probabilities also appears promising.

Proposal: Survey participants for candidate alternatives to (and views on) Markov models; develop Petri-net-based modeling for the analysis of computer reliability.

Personnel: Senior Analyst

Level of Effort: 1 man-year

Priority: TBD

TASK DEFINITION WORK SHEET

Participant's Name J. F. Meyer

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: I-5

Task Title: Model Solution Methods

Category (check): Reliability confirmation X
Fault processing verification ____
Fault processing characterization ____
Other (Specify) ____

Objective:
Determine methods for solving stochastic performance, reliability, and performability model (e.g., state bumping, decomposition of solutions in time and space, approx. solutions, etc.).

Procedure:
1) Identify problem areas.
2) Classification of solution techniques.
3) Investigation of particular methods.

Facilities:
Interactive computer system.
Program development environment.

Personnel:
At least 2 reliability theorists (interaction among personnel is necessary here).

Level of Effort:
2 @ 1/4 - 1/2 time

Priority:
High

TASK DEFINITION WORK SHEET

Participant's Name Herb Hecht

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: I-6

Task Title: Evaluation of Reliability Requirements

Category (check):

Reliability confirmation	X
Fault processing verification	
Fault processing characterization	
Other (Specify)	

Objective:
To keep the objective of the reliability confirmation in line with (1) current technology, (2) current regulations, (3) observed incidents (aircraft accidents and component failure patterns).

Procedure:
Establish a function for keeping track of (1) - (3) above, translate these into AIRLAB tasks or modifications of these.

Facilities:
Office and literature.

Personnel:
1 Senior System Engineer

Level of Effort:
Full-time (this individual may also be able to make formal evaluations of AIRLAB against current requirements).

Priority:
Highest

TASK DEFINITION WORK SHEET

Participant's Name Nicholas D. Murray

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: I-7
Task Title: Performance Confirmation

Category (check): Reliability confirmation X
Fault processing verification ____
Fault processing characterization ____
Other (Specify) Performance confirmation

Objective:
Using a state model for reliability analysis, a definition must be made between "good" states and "failed" states. There are two drivers for this definition:

1) The operational behavior of the system under fault conditions (i.e., voting, comparing, etc.).

2) Sufficient resources available to service the flight-critical applications.

The reliability model needs to be augmented to reflect sufficient performance (or lack of performance). For instance, the SIFT has a model of the schedule/allocation routine that supports the reliability model.

Also, it would appear that AIRLAB could be a tool to find the performance threshold.

TASK DEFINITION WORK SHEET

Participant's Name A. Hopkins

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: I-8

Task Title: Performance Analysis

Category (check): Reliability confirmation ____
Fault processing verification ____
Fault processing characterization ____
Other (Specify) Performance X

Objective:
Develop a structural model of performance capability in context with applications requirements.

Procedure:
Analysis of sample application and the fault-tolerant computer's scheduler.

Facilities:
Computation

Personnel:
Computer Scientist - 6 man-months
Flight Control Engineer - 1 man-month

Level of Effort:
Computer Scientist - 6 man-months
Flight Control Engineer - 1 man-month

Priority:
High

TASK DEFINITION WORK SHEET

Participant's Name Melliar-Smith

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: I-9

Task Title: Executive Implementation Proof

Category (check): Reliability confirmation X
Fault processing verification ____
Fault processing characterization ____
Other (Specify) ____

Objective:
To establish at AIRLAB the capability to formally verify the implementation of the executive (and associated) programs against their specification. Proof of high-level language and machine instruction will be required.

Procedure:
Collaboration with academic and research laboratories to obtain a proof system and to become familiar with its use. Development of guidelines for industrial implementers to facilitate proof.

Facilities:
Large multi-access computer, preferably DEC system 20.

Personnel:
1 plus Computer Scientist

Level of Effort:
Continuing

Priority:
Urgent because of very limited current capability of NASA and because of difficulty of recruiting staff.

TASK DEFINITION WORK SHEET

Participant's Name Melliar-Smith

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: I-10

Task Title: Application Requirements Analysis

Category (check):

Reliability confirmation	X
Fault processing verification	
Fault processing characterization	
Other (Specify)	

Objective:

To develop methods for formally verifying the specifications of the application tasks against the underlying aerodynamic and structural requirements.

Procedure:

Collaboration with academic and research labs and with NASA and industrial design teams to develop capability within a few years.

Facilities:

Large multi-access computer.

Personnel:

1 plus Computer Scientist
1 plus Mathematician with background in aerodynamics and structures.

Level of Effort:

Continuing

Priority:

Unless already underway elsewhere (at required level of formality), urgent because of long lead time, political delays, and difficulty of recruiting staff.

TASK DEFINITION WORK SHEET

Participant's Name Melliar-Smith

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: I-11

Task Title: Application Program Proof

Category (check):

Reliability confirmation	X
Fault processing verification	

Fault processing characterization ____
Other (Specify) _____

Objective:
To develop, or support the development of, or to become familiar with research programs aiming at verification of the correctness of application programs by mathematical analysis.

Procedure:
Collaboration with academic and research labs to develop capability within a few years. Develop programming standards to allow proof of production flight control programs.

Facilities:
Large multi-access computer, preferably DEC system 20

Personnel:
2 plus Computer Scientist

Level of Effort:
Continuing

Priority:
Urgent because of long lead time and because of difficulty of recruiting staff.

FAULT PROCESSING VERIFICATION

Tasks II-1 through II-13

Proposed by
Preliminary Working Group II Participants

D. Siewiorek (Chairman)
J. Abraham
J. Clary
R. Joobbani

Tasks II-14 through II-21

Proposed by
Working Group II Participants As Indicated

TASK II-1

Title: Initial Check-Up (Diagnostic)

Objective:
Verify that the single processor performs the basic functions.

Procedure:
Run the standard diagnostic supplied by the manufacturer.

Facilities:
1. Hardware - single subject processor.
2. Software - single subject processor diagnostic programs.

Personnel:
Software Technician

Level of Effort:
Assume the diagnostic to be run
5 times a day for 15 days,
1/2 man-month

Priority: High

TASK II-2

Title: Programmer's Manual Validation

Objective:
1. To look for design errors. To make sure the machine performs the functions according to its specification as documented in programmer's manual.

2. To investigate incompletely described machine features and fully characterize those features (e.g., I/O, Interface, Interrupts).

Procedure:
1. Perform the functions documented in programmer's manual and validate their correctness.

2. Investigate the system response to situations not completely specified in the programmer's manual and record the responses.

Facilities:
1. Hardware - single subject processor (e.g., BDX-930)
   - any peripheral or devices interfaced to the system that might be used later (e.g., sensors, actuators)
   - test/monitor computer

2. Software - programmer's manual
   - program development environment

- on-line debugger
- experimental results recording and assessment software

Personnel:
Chief Experimenter
Software Technician

Level of Effort:
Assume 100 instructions, each instruction checked with 10 different test cases or values (e.g., for different address mode valid and/or invalid).

If there are about 10 instructions per test and 20 instructions per day can be executed, then a total of 2 tests per day are done.

The total manpower for instruction testing is

$$\frac{100 \text{ instr.} * 10 \text{ test cases}}{2 \text{ tests/day}} = 500 \text{ man-days}$$

$\approx$ 1.5 man-years.

About 6 man-months are also required for investigating the system response to situations not completely specified in the Programmer's Manual.

The total required manpower then will be about 2 man-years.

Priority: High

TASK II-3

Title: Executive Routine, Including Error Mangement (Reconfiguration) Routines, Validation (Design Errors)

Objective:
1. To validate that executive routines respond as specified.
2. Search for design errors (or lack of specification) in executive routines.

Procedure:
1. Treat executive routines as black boxes with only inputs and outputs. Check response to

   a. expected data,
   b. out-of-bounds data,
   c. inconsistent data.

   } individual routines

   Data may be generated from examining the software, from specification of the software module, or randomly. Make sure every path through individual routines is exercised at least once, including error return paths. Validate consistency of response by multiple experiments.

2. Check executive routine interaction (pipeline). Feed a "string" of executive routines with

   a. expected data,
   b. out-of-bounds data,
   c. inconsistent data.

   } valid routine sequences

3. Determine (measure) routine response time.

4. Check system error return-reporting paths (i.e., through system hierarchy).

Facilities:

1. Hardware - single subject processor
   - test monitor computer

2. Software - program development environment

   a. on-line debugger like ODT, 6-12
   b. experimental results recorder and assessment software
   c. automated executive routine exercises that only need to know input/output area

Personnel:

Chief Experimenter
Software Technician

Level of Effort:

Assume 5 inputs, 5 outputs per module - each module 100 assembly language instr ⟶ 20 PASCAL lines
1000 lines PASCAL/20 ⟶ 50 routines
250 inputs x 20 experiments each ⟶ 5000 experiments
4 routines/day ⟶ 3 man-weeks
cross product - each routine talks to 2 others
100 routine combinations ⟶ 6 man-weeks
3 man-months

Priority: High

TASK II-4

Title: Multiprocessor Interconnection Validation

Objective:

1. To validate the interconnections between the processors.

2. To validate the functionality of the system with respect to the interconnection (i.e., communication, protocol handling and related processor effects).

Procedure:

1. Design and run diagnostic routines that check the interconnections and the protocol (obviously, there is no need to duplicate diagnostics provided by the manufacturer).

2. Run single processor diagnostic on each and/or all of the processors and observe the effects on other processors.

3. Make all the processors talk to each other and observe the behavior (such as bus contention, missing messages, protocol handling, priority conflicts, etc.).

4. Determine (measure) response time to communication setup, messages and interrupts for each processor.

Facilities:

1. Hardware - subject multiprocessor
   - bus monitor hardware
   - test monitor computer

2. Software - program development environment
   - on-line debugger
   - experimental results recorder and assessment software

Personnel:

Chief Experimenter
Software Technician
Hardware Technician

Level of Effort:

Assume there are six processors connected to each other (fully connected); running the multiprocessor diagnostic and observing the interconnection on one process takes half a man-month, so six processors and cross product takes 1/2 * 6 = 3 man-months.

TASK II-5

Title: Multiprocessor Executive Routine, Including Error Management (Reconfiguration) Routines, Validation

Objective:

1. To validate that the executive routines (multiprocessor system executive and single processor executive) respond as specified.

2. To search for design errors (or lack of specification) in executive routines.

Procedure:

1. Treat executive routines as black boxes with only inputs and outputs. Check response to:

   a. expected data,

b. out-of-bound data, and
c. inconsistent data.

2. Make sure every path through individual routines is exercised at least once, including error return paths. Validate consistency of response by multiple experiments.

3. Check executive routines interaction.

   a. feed a string of executive subroutines with
      - expected data
      - out-of-bound data
      - inconsistent data

4. Check the side effects of executive subroutines when running in different processors at the same time.

5. Check the scheduling and task assignment.

6. Determine (measure) routine response time.

7. Check system error return-reporting paths (i.e., through system hierarchy).

Facilities:

1. Hardware - subject multiprocessor
   - test monitor computer

2. Software - software development environment
   - on-line debugger
   - experimental results recorder and assessment software

Personnel:

Chief Experimenter
Software Technician

Level of Effort:

Same as Task II-3 except that since the executive runs on different processors (assume 6 processors) and the system executive is added, then we need 6*3 man-months = 18 man-months.

Priority: High

TASK II-6

Title: Application Program Validation (Design Errors) and Performance Baseline

Objective:

1. To verify that application software responses are as specified.

2. Search for design errors.

3. Measure system response parameters (time constants) with time varying inputs.

Procedure:

1. Treat system as a black box with only inputs and outputs. Check response to

   a. expected data,
   b. out-of-bounds data,
   c. inconsistent data,
   d. random data at boundaries, i.e., on the edge of control, where flight transitions (or transitions to other software),
   e. sequences of data,
      - expected
      - out-of-bounds
      - inconsistent
      - random.

2. Make sure every path through software is exercised (whole system).

3. Check system error return and reporting.

Facilities:

1. Hardware - total system under test

2. Software - same as Task II-3
   - executive error reporting

Personnel:

Chief Experimenter
Software Technician

Level of Effort:

Approximately 4 times Task II-3, assuming application described in Task II-13 is 4 times more complex than executive.
1 man-year

Priority: High

TASK II-7

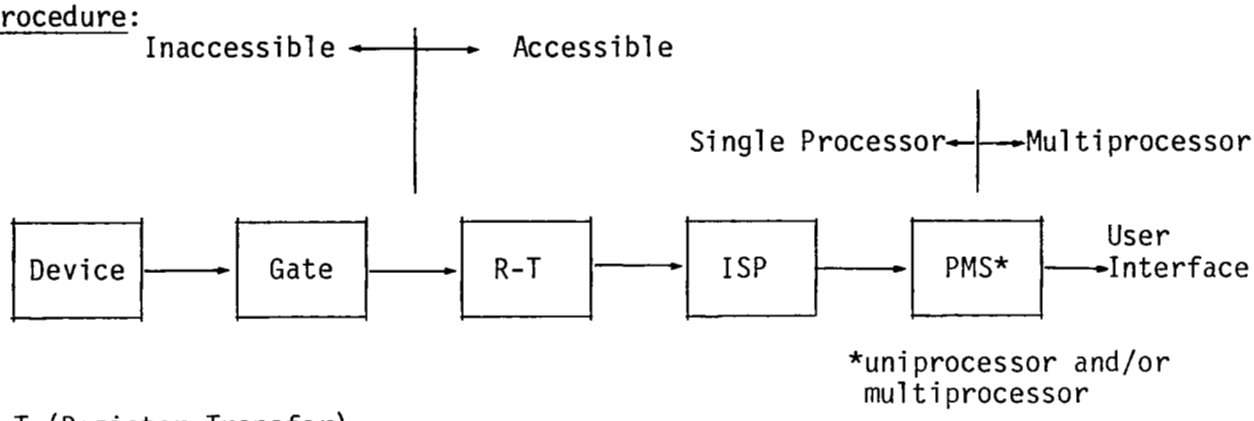
Title: Simulation of Inaccessible Physical Failures

Objectives:

1. To enhance understanding of the relationship between physical faults and their error manifestations.

2. To provide a data base which may be used to supplement/support physical fault injection experiments.

Procedure:



R-T (Register-Transfer)
ISP (Instruction Set Processor)
PMS (Processor Memory Switch)

1. Develop/obtain required simulation software with fault-injection capability at each of above levels.

2. Provide interfaces between the packages so that parts of system can be simulated at different levels.

3. Simulate fault-free system at varying (appropriate) levels of complexity to validate the simulation software.

4. Simulate system with injected faults.

5. Observe relationship between injected faults, input data and fault manifestation from Step 4.

6. Characterize relationships between injected faults and their manifestation and attempt to abstract information into equivalence classes.

7. Repeat above for multiple processor system.

Facilities:

1. Hardware - 1 or more computers (dependent on whether simulations are specially developed) - (e.g., Nanodata QM-1 plus DEC-10)

2. Software - simulation software
   - description of target machine at appropriate level
   - diagnostic software for target machine to validate the simulation software
   - simulation executive with evaluation software
   - monitor and application software of target machine

Personnel:

4-8 man-years
1 Engineer, 3 Programmers - (1-2 years)*

Priority: High

*NOTE: Once required simulation software exists, additional experiments will require much less time.

TASK II-8

Title: Physical Fault Insertion: Single Processor Manifestation Understanding and Preliminary Characterization Histograms

Objective:

1. Establish fault classes (e.g., manifestation number) to cut down complexity of fault injection at higher system levels.

2. Generate "representative" system level histogram of detection, isolation and reconfiguration times.

Procedure:

1. Physically inject faults on a single processor implementation:

"solid"	pins
"intermittent"	gate manifestation
"transient"	whole chip common mode power

2. Use diagnostics to see what portion of the machine (as defined by the diagnostic) does not work.

3. Map physical faults into "memory" or higher level manifestation wherever possible.

4. Automatically log each experiment and its results for single processor.

Facilities:

1. Hardware - monitoring computer
   - mag tape for records
   - high-speed data logger (if desperate)
   - test jig for inserting physical faults

2. Software - diagnostics
   - instruction to executive software that processor is available
   - broken diagnostic analyzer plus state dump
   - modify executive (if necessary) to report errors to monitor computer
   - some support from executive to coordinate fault injection with system state

Personnel:

1-2 Engineers - 3 shifts of technicians
10000/50/day = 200 days ($\approx$ 1 year)
$\approx$ 4-5 man-years
worst case $\longrightarrow$ 2 times longer

Priority: High

TASK II-9

Title: Physical Fault Insertion (Multiprocessor)

Objective:

Establish fault classes (map physical faults into manifestations). This will reduce the number of fault injections required at higher system levels.

Procedure:

1. Use the memory manifestation of physical faults classified in Task II-8 as a basis for injecting faults at the multiprocessor level. Insert physical faults for other cases in each of the single processors (i.e., unclassified faults).

2. Insert physical faults at the interconnection between each two processors

"solid"	pins
"intermittent"	gate manifestation
"transient"	whole chip
	common mode
	power

3. Map the manifestation of the fault insertion in steps 1. and 2. into the "memory" manifestation whenever possible.

4. Automatically log each experiment and its results (verify that all the other processors are not affected by the fault propagation).

5. Repeat the experiment a small number of times (e.g., < 100) to create histograms for:
   - fault propagation time
   - detection time
   - isolation time
   - reconfiguration time.

Facilities:

1. Hardware - monitoring computer
   - mag tape for records
   - high-speed data logger (if desperate)
2. Software - diagnostics
   - instruction to target system executive software that new processor is available
   - broken diagnostic analyzer plus state dump
   - modify executive (if necessary) to report errors to monitor computer
   - some support from executive to coordinate fault injection with system state

Personnel:

1-2 Engineers - 3 shifts of Technicians

10000/50/day = 200 days ($\approx$ 1 year) for single processor

Assume 6 processors
≈25 man-years
worst case ⟶ 2 times longer

Priority: High

TASK II-10

Title: Characterization of Executive Routine (Single Processor) Responses Under Single Physical Failure Conditions

Objective:

1. To classify executive routine response to hardware failures (i.e., manifestations).

2. These results will be used to abstract failure manifestations for higher system levels.

3. Cut down the number of and/or simplify system level failure experiments.

Procedure:

1. Inject failures into operating software. Failures can be manifested by:

   a. failure classes identified by lower levels (e.g., Task II-8) and injected by DMA,
   b. physical fault insertion for the uncharacterizable failures,

2. Failures injected into:

   a. data input,
   b. temporary data during execution (if fault manifestation is processor sensitive),
   c. code at various positions selected by
      - exhaustion
      - randomly
      - selectively (e.g., at program control points since data instruction changes can be directly mapped to memory alterations).

3. Collect data automatically and charcterize into classes

   a. unclassified physical fault insertion
   b. memory alteration
      - specific areas of memory
      - randomly
   c. undetected or unrecoverable

Facilities:

1. Hardware - single object processor
   - DMA
   - monitoring computer

- mag tape for records
- high-speed data logger (if desperate)
- test jig for inserting physical faults

2. Software - program development environment
- on-line debugger like ODT, 6-12
- experimental results recorder and assessment software
- automated executive routine exercises that only need to know input/output area

Personnel:
Chief Experimenter
Software Technician

Level of Effort:
Assuming (.9) x 10,000 maps 10 to a class ⟶ 900 classes
900 classes per each instruction
- 5 x 1000 lines ⟶ 4.5×10^6 experiments
- if automated and experiment every 10 seconds
⟶ 10,000 hours or 5 man-years

Priority: High

TASK II-11

Title: Multiprocessor System Executive, Including Error Management (Reconfiguration) Routines, Fault Handling Capabilities

Objective:
1. To classify the multiprocessor system executive response to hardware failures.

2. To measure the system time constant and scheduling algorithm

Procedure:
1. Inject failures into the running software. Failures can be manifested by:

   a. failure classes identified by lower levels (e.g., Task II-8 and Task II-9) and injected by DMA,
   b. physical fault insertion for the uncharacterizable failures.

2. Failures injected into:

   a. input data,
   b. temporary data during execution (if fault manifestation is fault sensitive),
   c. code at various positions selected by
      - exhaustion,
      - randomly,
      - selectively (e.g., at program control points since data instruction changes can be directly mapped to memory alterations).

3. Insert the faults characterized in Task II-10.

4. Collect data automatically and characterize the faults into classes:

   a. physical fault insert
   b. memory alteration
   c. system crash

5. Collect data for system response:

   a. detection time,
   b. isolation time,
   c. recovery time.

Facilities:

1. Hardware - single object processor
   - DMA
   - monitoring computer
   - mag tape for records
   - high-speed data logger (if desperate)
   - test jig for inserting physical faults

2. Software - program development environment
   - on-line debugger like ODT, 6-12
   - experimental results recorder and assessment software
   - automated executive routine exercises that only need to know input/output area

Level of Effort:

Assuming (.9) x 10,000 maps 10 to a class ⟶ 900 classes
900 classes per each instruction
- 5 x 1000 lines ⟶ 4.5×10^6 experiments
  Assume 6 processors ⟶ $6 \times 4.5 \times 10^6 = 27 \times 10^6$ experiments
- if automated and experiment every 10 seconds
  ⟶ 75,000 hours or 25 man-years

Personnel:

Chief Experimenter
Software Technician

TASK II-12

Title: Application Program Fault Handling on Multiprocessors

Objective:

1. Classify application software response to hardware failures.

2. Measure system response parameters (time constant) in failure situations.

Procedure:

1. Repeat Task II-11 for error handling.

Facilities:
Same as Tasks II-6 and II-11

Task II-13

Title: Multiple Application Programs Fault Handling on Multiprocessor

Objective:
1. Characterize system scheduling and response parameters in failure situations.

2. Classify application program and system software response to hardware failures.

Procedure:
Repeat Task II-12 when multiple application programs are running.

Facilities:
Same as Task II-12

TASK DEFINITION WORK SHEET

Participant's Name Herb Hecht

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: II-14

Task Title: Software Reliability Research

Category (check):

Reliability confirmation	X
Fault processing verification	X
Fault processing characterization	
Other (Specify)	

Objective:
1. To identify severity of manifestations of software failures.

2. To measure software future labs as a function of stress (interrupt rate, improper data, hardware features).

Procedure:
To implement measurement of pertinent parameters in AIRLAB.

Facilities:
Normal software development facilities SIFT and/or FTMP.

Personnel:
1. Software Engineer

2. Computer Technician

Level of Effort:

1. 1 man-year

2. 1/2 man-year over 2 years
   (these levels are based on running these tests concurrent with hardware reliability tests)

Priority:
High

TASK DEFINITION WORK SHEET

Participant's Name John M. Myers

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: II-15

Task Title: Measurement of Synchronization of Clocks

Category (check):		
	Reliability confirmation	X
	Fault processing verification	X
	Fault processing characterization	
	Other (Specify)	

Objective:
Make a preliminary experimental determination of the behavior of functioning clocks; develop instrumentation capable of distinguishing rare clock failure from instrumentation failure.

Explanation:
The synchronization of a clock network is essential to fault-tolerant computers. The only experimental results (available for atomic clocks) cast doubt on the assumption that functioning clocks have either a gaussian distribution or a fixed bound of phase separation. This assumption is critical to the proofs of fault-tolerance of the clock networks.

Procedure:
Measure and record phase separation of clocks in a four-clock network, using redundant phase measurement, with A/D converters that are offset one from another to avoid confusing clock faults with A/D glitch; use guard indicators to rule out effects due to power supply irregularities. Collect frequency distributions of phase separation, and see what indication there is of

a) bounded,
b) gaussian, or
c) broader (e.g., unbounded) distribution.

Facilities:
Clock network from SIFT and/or FTMP, counters, analog/digital converters, magnetic tape recorders, phase measuring circuits (timers), i.e., electronics laboratory with recording equipment.

Personnel:
Engineer - 2 man-months
Technician - 3 man-months
Analyst - 2 man-months

Level of Effort:
1 month to set up; then low-level of monitoring for two years.

Priority:
High

TASK DEFINITION WORK SHEET

Participant's Name Melliar-Smith

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: II-16

Task Title: Failure Severity Analysis

Category (check):

Reliability confirmation	
Fault processing verification	X
Fault processing characterization	
Other (Specify)	

Objective:
To obtain from industry, or develop, a categorization of failure syndromes against the severity of their consequences. This is intended to provide input to the reliability analysis.

Procedure:

Facilities:

Personnel:

Level of Effort:

Priority:
Needed as input to reliability analysis.

TASK DEFINITION WORK SHEET

Participant's Name Kurt Moses

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: II-17

Task Title: Application Program Validation and Performance Baseline on a System (as distinct from Task II-6)

Category (check):

Reliability confirmation	
Fault processing verification	X
Fault processing characterization	
Other (Specify)	

Objective:
Validate application software for the system. (Task II-6, as written, can only validate the computer - a part of the system).

Procedure:
Interface SIFT/FTMP with a real-time simulation of a suitable aircraft, actuator configuration, and sensor configuration. Perform the tasks of II-6 for different aircraft and environmental conditions (winds, gusts, etc.)

Facilities:
a) Hardware - SIFT or FTMP computer; host computer for (b).

b) Aircraft/sensor/actuator simulation, including computer interface.

Personnel:
Cobol System Engineer
Software Technician
Chief Experimenter

Level of Effort:
3 man-years, min.

Priority:
High

TASK DEFINITION WORK SHEET

Participant's Name Basil Smith

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: II-18

Task Title: Software Fault - Containment Verification

Category (check):

Reliability confirmation	
Fault processing verification	X
Fault processing characterization	

Other (Specify) _____

Objective:
Test should stress the system's ability to contain software errors to affected application programs.

Procedure:
Induce failure of applications code via random changes and generate most malignant behavior "test" applications that are designed to stress its containment mechanisms.

Facilities:
Object F-T computer system
Monitoring system
Software support facility

Personnel:
Analyst/Clever Programmer

Level of Effort:
6 man-months

Priority:
High

TASK DEFINITION WORK SHEET

Participant's Name Melliar-Smith

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: II-19

Task Title: Logical Analysis of Design to Reduce Size of Test Data Sets

Category (check):
Reliability confirmation ____
Fault processing verification X
Fault processing characterization ____
Other (Specify) _____

Objective:
The very large number of test cases necessary for coverage can be reduced by logical analysis of the design, in fault free and fault present cases.

Procedure:
Collaboration with academic and industrial researchers already working on systematic testing.

Facilities:
Large multi-access computer.

Personnel:
1 Computer Scientist

Level of Effort:
Continuing

Priority:
Must precede testing tasks.

TASK DEFINITION WORK SHEET

Participant's Name D. B. Mulcare

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: II-20

Task Title: Definition and Exercising of Non-Application-Dependent Applications Software Benchmark Package*

Category (check):

Reliability confirmation	
Fault processing verification	X
Fault processing characterization	
Other (Specify)	

Objective:
To define, evaluate, and employ a benchmark applications software program for integrated avionics/flight control to use in other tasks.

Procedure:
Develop an appropriate mix of software modules (of varying size, criticality, time constrained, rep rate, etc.) which realistically exercise the multiprocessors in a representative manner. Many modules might only be dummy modules occupying a prescribed time slice.

Facilities:
Compiler

Personnel:
System Engineer
Software Specialist

Level of Effort:
6 mos.

Priority:
Moderate

*NOTE: This task might well be carried out as part of already designed Task II or III experiments.

TASK DEFINITION WORK SHEET

Participant's Name Jacob Abraham

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: II-21

Task Title: Verify that the Operational System Diagnostic has a High Coverage

Category (check):

Reliability confirmation	
Fault processing verification	X
Fault processing characterization	
Other (Specify)	

Objective:
Periodic diagnosis is necessary to flush out latent faults so that assumptions made in reliability model are valid.

Procedure:
(a) Research into fault classes, prove coverage, and/or

(b) Simulation of gate level - get coverage number.

Facilities:
Fault simulator if second avenue (b) is used.

Personnel:
(a) Experienced Researcher

(b) Technician

Level of Effort:
1 man-year

Priority:
High

FAULT PROCESSING CHARACTERIZATION

Tasks III-1 through III-7

Proposed by
Preliminary Working Group II Participants

A. Hopkins (Chairman)
D. Rennels
J. Gault
M. Smith

Tasks III-8 through III-13

Proposed by
Working Group II Participants As Indicated

TASK III-1

Title: False Input Information

Objective:

To determine the fault-tolerant computer system's response to input information which is plausible (passes bound checks) but incorrect. This task will help to characterize the damage produced in output responses dependent upon the erroneous inputs. In addition, careful attention must be paid to detect unexpected disturbances in system behavior which would not normally be affected by the input errors. This experiment will help to establish the system's vulnerability to input data corruption. By causing input failures which are half levels, the ability of the system to accurately detect and isolate single source data errors is characterized.

Approach:

Three classes of data corruption will be used: subtle inconsistencies, interference, and half values. For input sensor points, inject data values which are not consistent with the simulated environment. These values will be subtle inconsistencies which are difficult to detect by bounds checks or analytic redundancy. Faulty data injection will be varied as to its duration, severity (degree of inconsistency), and input source. The results of this task will help to identify what input discrepancies are difficult to detect and characterize the system's response to this class of failures.

1. For various input points, inject noise of varying intensity and duration. This will help to characterize the impact of variable length external disturbances on system behavior. In addition, criticality of inputs may be detected, i.e., noise at one locality may be more damaging than at another.

2. For various input sensors, inject half level signals. This test will help characterize the system's ability to detect and isolate faults which may appear as inconsistent values at different points.

Facilities:
AIRLAB facilities for simulation of data inputs to the fault-tolerant computing system.

Level of Effort:
TBD

TASK III-2

Task: Memory Alteration

Objective:
Selective memory alteration can be used to characterize the robustness of a fault-tolerant system when confronted with a) software and design errors, b) latent faults, c) correlated faults, and d) confusion by divergence.

Approach:
Inject changes at one or more locations within various computer memories, either simultaneously or sequentially. Examples are tests for the effects of:

1. Simple Software Errors - Place an (identical) incorrect value in identical memory locations within various redundant computers. Tabulate system failures as a function of number of fault insertions and area of memory disturbed.

2. Latency - Injecting faults in various memories at uncorrelated internal locations. Tabulate system failures as a function of the rate and domain of fault injection.

3. Confusion by Divergence - Injecting differing values in identical locations in the memories of the various redundant computers. Tabulate system failures as a function of fault insertions and areas of memory disturbed.

Facilities:
1. Multiple DMA or equivalent
2. Revised local executives
3. Special interfaces and cables

Level of Effort:
TBD

TASK III-3

Task: Configuration Control Manipulation

Objectives:
Find the limits of the system when resources are arbitrarily removed/replaced.

1. See what load shedding is done when resources are removed.

2. Is minimum system response time met?

3. Are critical functions (on a priority level) done?

4. How quickly does system recover when resources are replaced?

5. What is system response to any syndrome?

Approach:

1. Arbitrarily remove system resources (i.e., buses, memory) until system response drops below a minimum time; critical functions are not done or system dies.

   a. these should be done several times with different resources removed at different "times/rates."
   b. view the load shedding, if any, to see if vital functions are still being carried out; who is shedded first; when is decision made (configuration at that time).
   c. see if decision maker works "consistently."
   d. what inputs are ignored as functionality degrades; what alternatives are chosen?

2. Restore resources arbitrarily.

   a. what functions restored first (configuration at that time)?
   b. are resources "restarted/initialized" correctly?

Outcome:

1. "Remove/replace actions", i.e., configuration should be tabulated against

   a. system response time.
   b. "critical" functions still in operation.
   c. system inputs should be driven at a constant maximum rate.
   d. the approaches need to be adjusted to each individual system and how their reconfiguration management is done.

Facilities:

1. Modified executive

2. A way to measure system response

3. Means to remove/replace resources arbitrarily in any order desired

Level of Effort:

TBD

TASK III-4

Task: Cold and Warm Start Manipulation

Objective:
Explore the behavior of the system when its "instinctive" initial start capabilities are stressed. For example, what happens when some modules perceive a power-on transient while the rest do not? More generally, wherever modules possess special operational modes that can be stimulated apart from a system consensus, the consequences of various configurations of such modes is to be explored.

Approach:
The approach must be detailed in view of the specific properties of the units under test. In general, the special modes will be directly stimulated in various subsets of modules by the test coordination unit, which is presumed here to be a minicomputer. Derive a generalized behavioral description from the resultant configuration data.

Facilities:
1. Special interfaces and cables

2. Revised local executives

Level of Effort:
TBD

TASK III-5

Task: Common Mode Noise and Margin Test

Objective:
Characterize the responses of the intercommunication buses, power buses and input/output buses to common mode noise and signals approaching and/or exceeding their boundaries, in either the time or amplitude domain.

Approach:
1. Common Mode Noise

   a. inject noise on the buses (power, intercommunication, I/O) either singly or in multiples and view the system's responses; i.e., when does it die; what are its precursors to death?
      - noise can be raised/lowered in frequency.
      - noise can be raised/lowered in duration.
      - noise can be raised/lowered in amplitude.
      - crosstalk within a bus.

   Examples - with what frequency and duration can noise appear on an intercommunication bus before the system gets confused and can't reconfigure?
   - what size glitches can power supply support?
   - at what amplitude does noise start masking the signal?

2. Margin Testing

a. vary the signals with respect to either time or amplitude and view the system's response; ascertain "who" failed first and why.

Examples - vary voltage until some "units" are unable to respond; insufficient power to drive line driver.
- what are responses of buffers/drivers to signals that are almost 'I/O'?

Facilities:

1. Special interfaces and cables
2. Noise generators for EMI and RFI
3. Off-board data recorders
4. Voltage, current generators; logic analyzers
5. Revised local executives

Level of Effort:
TBD

TASK III-6

Title: Clock Modification Tests

Objectives:
To control the system clock so that the specifications of frequency, shape, and synchronization are stressed until the system fails. After a failure has occurred, syndrome data can be analyzed to determine which factors of performance degraded and what ultimately caused the failure. This test will typify the clock operating characteristics which can be tolerated and, hence, will indicate the margins in the clock system specification.

Approach:
The clock modifications which are specifically employed are implementation-dependent. There are a number of alterations to the clock signal, beyond specified values, which are to be applied.

clock rate: The clock rate is increased and performance data collected until a system failure occurs. The behavior of the system as it approaches the critical clock rate will characterize the system failures as the specified clock rate is exceeded.

synchronization (skew): The synchronization (skew) between processors is increased until system failure occurs. As in the clock rate experiment, the behavior of the system as the failure state is approached will be analyzed in addition to determining the critical value of skew.

clock system noise: The clock, for single or multiple processors, is perturbed by noise of various durations and levels (nondestructive). This

activity will help to characterize the system's vulnerability to disturbance of this type.

Facilities:
The capability to manipulate the existing clock and to obtain system syndromes. There are no unique facilities required for this experiment.

TASK III-7

Title: Multiple Fault Injections

Objective:
This test is intended to characterize the "breaking point" of a fault-tolerant system under multiple faults. Parameters of interest are the number of insertion points, the fault rates and durations, and effects when points are excited with faults simultaneously.

Approach:
Each of N sets of tests examines the system performance with a different number of faults being inserted ($1 \leq i \leq N$). The time of insertion is generated randomly (Poisson) for each point at a rate R. For each set of tests system failures are tabulated as a function of i and R. Sensitive points are identified at which system failures occur at below-average fault arrival rates.

Additional tests may be included which vary the fault duration, or which insert faults simultaneously at selected points.

Facilities: Same as for Fault Injection (multiple copies)

Level of Effort: Arduous

TASK DEFINITION WORK SHEET

Participant's Name G. Masson

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: III-8

Task Title: Multiple, but sequential, injection of faults at potentially vulnerable times

Category (check):
Reliability confirmation ____
Fault processing verification ____
Fault processing characterization X
Other (Specify) ____

Objective:
What if in "SIFT or FTMP" types of systems, faults were injected, for example, in reconfiguration modes of operation? For example, a fault is

injected, the system responds; but as it responds, another fault is injected, and so forth.

Procedure:
Task III-7

Facilities:
Task III-7

Personnel:
Task III-7

Level of Effort:
Task III-7

Priority:
10

TASK DEFINITION WORK SHEET

Participant's Name John M. Myers

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: III-9

Task Title: Surprise Assessment Methodology and Examples

Category (check): Reliability confirmation X
Fault processing verification X
Fault processing characterization X (cuts across all three)
Other (Specify) _____

Objective:
Avoid some catastrophes by early appreciation of surprises, and promote new approaches by active "harvesting" of surprises.

Procedure:
Seek (at NASA, SRI, Draper, Bendix, Collins, etc.) surprise incidents. Develop Petri-nets to show system as perceived before and after surprise. Show alternative. Develop concert of mapping from one net to another as a measure of the impact of a surprise. Disseminate.

Facilities:
Access to computer with graphics desirable.

Personnel:
Two Senior Analysts

Level of Effort:
Quarter- to half-time - two years each

Priority:
TBD

TASK DEFINITION WORK SHEET

Participant's Name A. H. Lindler

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: III-10

Task Title: Evaluation of Strategy for Handling Transient and/or Intermittent Faults

Category (check):

Reliability confirmation	
Fault processing verification	X
Fault processing characterization	X
Other (Specify)	

Objective:
Verify predicted system response.
Characterize system performance.

Procedure:
Insert transients and intermittents and record system response.

Facilities:
Same as Task II-9.

Personnel:
TBD

Level of Effort:
TBD

Priority:
8

TASK DEFINITION WORK SHEET

Participant's Name C. L. Seacord

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: III-11

Task Title: Analysis/testing to determine the tolerance or sensitivity hardware parameter variation

Category (check): Reliability confirmation X
Fault processing verification
Fault processing characterization X
Other (Specify) _____

Objective:
Determine the change in performance or (more likely) fault reaction that will accompany the differences in hardware characteristics due either to piece part tolerance or environment.

Procedure:

Facilities:

Personnel:
1 Computer Designer/Analyst
1 Technician
1/2-time Software Engineer

Level of Effort:
3-6 months elapsed time and about 25% hardware availability

Priority:
High to Medium

TASK DEFINITION WORK SHEET

Participant's Name A. H. Lindler

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: III-12

Task Title: Sensitivity of system to phase lag

Category (check): Reliability confirmation
Fault processing verification
Fault processing characterization X
Other (Specify) _____

Objective:
Determine maximum phase lag the system will tolerate (phase lag at which given function is not performed satisfactorily).

Procedure:
Select a set of "high" frequency tasks, force greater than normal phase lag.

Facilities:
TBD

Personnel:
TBD

Level of Effort:
TBD

Priority:
8

TASK DEFINITION WORK SHEET

Participant's Name Pio De Feo

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: III-13

Task Title: Effects of Massive Failures

Category (check):
- Reliability confirmation ____
- Fault processing verification ____
- Fault processing characterization X
- Other (Specify) ____

Objective:
Analyze effects of massive failures like cracked PC boards, failures which might be generated by lightning strikes, etc.

Procedure:
Physically insert solid massive failures which involve several chips, connectors, etc.

Facilities:
Flexible test jig.

Personnel:
1 Engineer
1-2 Technicians

Level of Effort:
TBD

Priority:
High

OTHER TASKS

Tasks IV-1 through IV-11

Proposed by

Working Group II Participants As Indicated

TASK DEFINITION WORK SHEET

Participant's Name John M. Myers

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: IV-1

Task Title: Theoretical Limitations of Fault Tolerance

Category (check): Reliability confirmation ____
Fault processing verification ____
Fault processing characterization ____
Other (Specify) Theory of fault tolerance

Objective:
Show theoretical limits to what faults can be tolerated by a fault-tolerant computer -- in the spirit of thermodynamic limits or efficiency.

Procedure:
Fault detection is analogous to recognition of non-theorems of a formal system. From the mathematical results of Godel; Turing; Jauski and Church, we know that there are fundamental limitations to the ability of any realizable procedure to detect non-theorems. In this task one would attempt to transform their formal results so as to illuminate fundamental bounds on what fault-tolerant computers can mask and/or detect.

Facilities:
Pencil and Paper

Personnel:
Senior Analyst

Level of Effort:
Half-time for one year

Priority:
High

TASK DEFINITION WORK SHEET

Participant's Name John M. Myers

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: IV-2

Task Title: Proving Timing Correctness

Category (check): Reliability confirmation ____
Fault processing verification ____
Fault processing characterization ____
Other (Specify) Reliability theory ____

Objective:
Provide programming and checking disciplines to guarantee correct timing of program execution.

Procedure:
Although proving the correctness of software is extremely difficult, one significant aspect is more tractable. Petri-net analysis similar to that used to analyze the FTMP clock network is feasible and can show the minimal restrictions on programming such that both systems and applications programs can be guaranteed to run on time. The same analysis will produce a checking procedure to guarantee that programs are written subject to the necessary restrictions.

Personnel:
Senior Analyst

Level of Effort:
1 man-year

Priority:
TBD

TASK DEFINITION WORK SHEET

Participant's Name A. H. Lindler

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: IV-3

Task Title: Trade-off of Diagnostic/Maintenance/Failure Low-Level Isolation Versus Decreased Reliability

Category (check): Reliability confirmation ____
Fault processing verification ____
Fault processing characterization ____
Other (Specify) Research Task ____

Objective:
Provide a general idea of system reliability degradation resulting from increased visibility of details of failures (sensing requires additional complexity in terms of test points which increase vulnerability of system to faults in addition to extra hardware and software).

Procedure:
Analysis

Facilities:
None

Personnel:
Systems Analyst and/or Reliability Analyst

Level of Effort:
TBD

Priority:
8

TASK DEFINITION WORK SHEET

Participant's Name Hopkins

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: IV-4

Task Title: Instrumentation for Timing Tests

Category (check): Reliability confirmation ____
Fault processing verification ____
Fault processing characterization ____
Other (Specify) Facility development

Objective:
Develop methodology to observe timing relationships on a non-interference basis (Heisenberg notwithstanding).

Procedure:
Evolve physical principles.
Design hardware experiments.
Design instrumentation.
Build and test.

Facilities:
TBD

Personnel:
Physicist/Engineer

Electronic Technician

Level of Effort:
5-10 man-months

Priority:
Highest

TASK DEFINITION WORK SHEET

Participant's Name G. Masson

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: IV-5

Task Title: Sensitivity Analysis/Measurement of a System to Faults

Category (check): Reliability confirmation ____
Fault processing verification ____
Fault processing characterization ____
Other (Specify) Basic research

Objective:
What are the measures of fault manifestation such that we can describe how sensitive a system is to faults? In other words, can I measure a minimal injection for system failure?

Procedure:
Tasks III-5 to 7

Facilities:
Tasks III-5 to 7

Personnel:
Tasks II-5 to 7

Level of Effort:
Tasks III-5 to 7

Priority:
10

TASK DEFINITION WORK SHEET

Participant's Name G. Masson

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: IV-6

Task Title: Fault-Test Interrelationship Characterization

Category (check): Reliability confirmation ____
Fault processing verification ____
Fault processing characterization ____
Other (Specify) Basic research ____

Objective:
In order to perform diagnosticability analysis on "SIFT/FTMP" types of systems, it is necessary that the complex interrelationships among faults and tests be mentioned. In other words, what faults do tests cover/detect, and what faults invalidate tests?

Procedure:
Injection of multiple faults (both permanent and intermittent) in subsystems being tested and subsystems doing the testing. Generation of an extended outcome matrix which describes the probability that a fault's subsystem detects faults in a fault's sybsystem.

Facilities:
Tasks III-5 to 7

Personnel:
Tasks III-5 to 7

Level of Effort:
Tasks III-5 to 7

Priority:
10

TASK DEFINITION WORK SHEET

Participant's Name G. Masson

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: IV-7

Task Title: Composite Validation

Category (check): Reliability confirmation ____
Fault processing verification ____
Fault processing characterization ____
Other (Specify) Basic research ____

Objective:
First, everyone should accept the fact that there will never be a technique which will yield a single number (i.e., 10^{-9}) with which to evaluate the reliability of a system. Instead, the validation will have to be based on

many contributions which when collected together provide strong (irrefutable) data/evidence regarding reliability.

Procedure:
What data bases can be generated by AIRLAB experiments and can theoretically be developed which utilize this data to form a composite element for the validation process?

Facilities:
Exposure to AIRLAB proposals, ongoing experiments, growth plans of AIRLAB.

Personnel:
1 Ph.D. Operations Research/Statistician
1 Assisting Analyst/Programmer

Level of Effort:
Continuous

Priority:
10

TASK DEFINITION WORK SHEET

Participant's Name G. Masson

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: IV-8

Task Title: Error Pattern Classification of Generic Analog/Physical Fault Mechanisms

Category (check):
Reliability confirmation ____
Fault processing verification ____
Fault processing characterization ____
Other (Specify) Basic research ____

Objective:
Given III-5, an important question is the following: What do various subsystems actually do when so distributed? Does a subsystem do anything each time or does it fall into certain modes of faulty operation that over many repetitions of the experiment are seen many times? Can these modes be classified by error patterns as the bus (address, data, control)?

Procedure:
Tasks III-5 to 7

Facilities:
Tasks III-5 to 7

Personnel:
Same

Level of Effort:
Same

Priority:
10

TASK DEFINITION WORK SHEET

Participant's Name G. Masson

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: IV-9

Task Title: Establishing Generic Fault Classes

Category (check): Reliability confirmation ____
Fault processing verification ____
Fault processing characterization ____
Other (Specify) Basic research

Objective:
To what level do we collapse faults operation and how do we describe this faults operation? For example, memory errors a bus patterns.

Procedure:
Inject ranges of analog faults and observe, for example, bus error patterns. Develop generic categories.

Facilities:
Task II-7 and Tasks III-5 to 7

Personnel:
Same

Level of Effort:
Same

Priority:
10

TASK DEFINITION WORK SHEET

Name: John Myers

Task Number: IV-10

Research Topics: Instrumenting Fault-Tolerant Computers

A. What specifications are needed for SIFT and FTMP so that they can be effectively monitored?

1) Can monitoring be done without decreasing reliability?
2) What needs to be monitored?

B. Instrumentation Requirements

1) What (from item 2 above) needs to be monitored?
2) What bandwidth is required?
3) How much parallelism? Concurrency?
4) How to provide for flexibly-controlled extraction of significant features, to avoid being delayed with uninteresting data?

Personnel:
Engineer, Technician, Analyst

Level of Effort:
Six man-months each

Priority: High

TASK DEFINITION WORK SHEET

Participant's Name G. Masson

Tasks which you feel are important and not included in the preliminary report may be proposed on this work sheet.

Task Number: IV-11

Task Title: Error Patterns on Buses as a High-Level Fault Manifestation

Category (check): Reliability confirmation ____
Fault processing verification ____
Fault processing characterization ____
Other (Specify) Basic research ____

Objective:
Above fault/changed memory locations in a high-level manifestation we could consider error patterns on buses in the sense that even if memory words are bad, when I read/write, I see the faults as an error pattern on the bus lines.

Procedure:
Tasks II-8 and 9, Tasks III-5 to 7

Facilities:
Bus injector hardware

Personnel:
Tasks III-5 to 7

Level of Effort:
Tasks III-5 to 7

Priority:
10

APPENDIX III - TASK RATING RESULTS FROM WORKING GROUP II

RELATIVE ASSESSMENT OF INITIAL PROPOSED VALIDATION TASKS

FOR FAULT-TOLERANT COMPUTER SYSTEMS

Working Group II participants were asked to assess the relative importance of the validation tasks identified in the preliminary working group meeting held in September. The result of these assessments is summarized in Figure D.1.

Participants in Working Group II were told to consider each task presented and rate them on a scale of 1 to 10. They were further told that a score of 10 was the highest rating a proposed task could receive and that 1 indicated the lowest rated tasks.

It should be noted that Task I-3 received the highest mean score of 9.4. Task II-10 received the lowest score of 5.7. Of the 23 proposed tasks, only 4 received a 1 - the lowest possible score. These were Tasks I-2, II-6, II-8 and II-10.

The task evaluation results should be taken only as an indication of the desirability of the proposed tasks. It is clear that a much more in-depth and precise description of objectives and methods of these tasks will be required.

APPENDIX IV - WORKING GROUP II ATTENDEES

* Indicates attendance at Preliminary Session

Dr. Jacob Abraham
Department of Electrical Engine
University of Illinois
Urbana, IL 61801
Phone: (217) 333-0750

Mr. Richard Alberts
Research Triangle Institute
Systems and Measurements Division
P.O. Box 12194
Research Triangle Park, NC 27709
Phone: (919) 541-5878

Mr. Salvatore Bavuso
NASA-Langley Research Center
Mail Stop 477
Hampton, VA 23665
Phone: (804) 827-3681

*Dr. William Carter
IBM, T. J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
Phone: (914) 945-1092

Mr. Jim Clary
Research Triangle Institute
Systems and Measurements Division
P.O. Box 12194
Research Triangle Park, NC 27709
Phone: (919) 541-6951

Dr. Steve Crocker
USC/Information Sciences Institute
4676 Admiralty Way
Marina Del Rey, CA 90291
Phone: (213) 822-1511

Dr. Pio De Feo
Mail Code: 210
NASA-Ames Research Center
Moffett Field, CA 94035
Phone: (415) 965-5048

Mr. Billy L. Dove
NASA-Langley Research Center
Mail Stop 477
Hampton, VA 23665
Phone: (804) 827-3681

Mr. Robert G. Flynn
The Boeing Company
P.O. Box 3707 Mail Stop 3N-43
Seattle, WA 98124
Phone: (206) 773-1786

Dr. James W. Gault
Department of Electrical Engineering
North Carolina State University
P.O. Box 5275
Raleigh, NC 27650
Phone: (919) 737-2376

*Mr. Jack Goldberg
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
Phone: (415) 326-6200, Ext. 2784

Dr. Herb Hecht
SoHaR, Incorporated
1040 S. La Jolla Avenue
Los Angeles, CA 90035
Phone: (213) 935-7039

Mr. Dale Holden
NASA-Langley Research Center
Mail Stop 477
Hampton, VA 23665
Phone: (804) 827-3681

Mr. Milt Holt
NASA-Langley Research Center
Mail Stop 477
Hampton, VA 23665
Phone: (804) 827-3681

Dr. Albert L. Hopkins, Jr.
C. S. Draper Laboratory, Inc.
Mail Station 35
555 Technology Square
Cambridge, MA 02139
Phone: (617) 258-1451

*Mr. Rostam Joobbani
Research Triangle Institute
Systems and Measurements Division
P.O. Box 12194
Research Triangle Park, NC 27709
Phone: (919) 541-6830

Dr. William Kautz
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
Phone: (415) 326-6200

Mr. Albert H. Lindler
NASA-Langley Research Center
Mail Stop 477
Hampton, VA 23665
Phone: (804) 827-3681

Mr. Brian Lupton
NASA-Langley Research Center
Mail Stop 477
Hampton, VA 23665
Phone: (804) 827-3681

Dr. Gerald Masson
Electrical Engineering Department
Johns Hopkins University
Barton Hall
Charles & 34th Streets
Baltimore, MD 21218
Phone: (301) 338-7013

Mr. Jack D. McDonnell
McDonnell Douglas Corporation
3855 Lakewood Boulevard
Mail Code 36-49
Long Beach, CA 90846
Phone: (213) 593-5616

Mr. Peter Michael Melliar-Smith
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
Phone: (415) 326-6200 Ext. 2336

Dr. J. F. Meyer
2523 Electrical Engineering Building
The University of Michigan
Ann Arbor, MI 48109
Phone: (313) 763-0037

Mr. G. Earl Migneault
NASA-Langley Research Center
Mail Stop 477
Hampton, VA 23665
Phone: (804) 827-3681

Mr. Kurt Moses
Bendix Corporation
Flight Systems Division, Dept. 7211
Teterboro, NJ 07608
Phone: (201) 288-2000 Ext. 1584

Mr. Dennis B. Mulcare
Lockheed-Georgia Company
Department 72-42, Zone 415
Marietta, GA 30063
Phone: (404) 424-4886

Mr. Nicholas D. Murray
NASA-Langley Research Center
Mail Stop 477
Hampton, VA 23665
Phone: (804) 827-3681

Dr. John M. Myers
77 North Washington Street
Boston, MA 02114
Phone: (617) 277-5301

*Dr. Dave Rennels
The Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91103
Phone: (213) 354-4321

Mr. Chuck O. Schulz
Rockwell International
400 Collins Road, N.E.
Cedar Rapids, IA 52406
Phone: (319) 395-3057

Mr. C. L. Seacord
Honeywell, Incorporated
Mail Station MN 15-2644
1625 Zarthan Avenue
St. Louis Park, MN 55416
Phone: (612) 542-5758

Dr. Dan P. Siewiorek
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA 15213
Phone: (412) 578-2570

Dr. Basil Smith
C. S. Draper Laboratory, Inc.
555 Technology Square
Cambridge, MA 02139
Phone: (617) 839-1321

*Mr. Myke Smith
Research Triangle Institute
Systems and Measurements Division
P.O. Box 12194
Research Triangle Park, NC 27709
Phone: (919) 541-6829

Dr. Richard K. Smyth
Milco International, Incorporated
15628 Graham Street
Huntington Beach, CA 92649
Phone: (714) 894-1717

Mr. J. Larry Spencer
NASA-Langley Research Center
Mail Stop 477
Hampton, VA 23665
Phone: (804) 827-3681

Mr. Robert M. Thomas, Jr.
NASA-Langley Research Center
Mail Stop 477
Hampton, VA 23665
Phone: (804) 827-3681

Dr. Kishor Trivedi
Computer Science Department
Duke University
Durham, NC 27710
Phone: (919) 684-3048

Dr. John G. Weber
Simulation Technology, Inc.
4124 Linden Avenue
Dayton, OH 45432
Phone: (513) 252-5623

Mr. John Wensley
August Systems, Incorporated
1940 McGilchrist Street, S.E.
Salem, OR 97302
Phone: (503) 364-1000

Mr. Allan White
Kentron International, Inc.
3221 N. Armistead
Hampton, VA 23665
Phone: (804) 838-1010

Table 1.1 Properties Which Make Ultra-Reliable Systems Difficult to Validate

1. Lifetesting is inappropriate.

2. System design complexity makes it difficult to:

   - perform failure effects analysis,
   - instrument and measure all relevant parameters,
   - use testing approaches since so many states and failure modes are possible.

3. Large-scale integration technology realization makes it difficult to:

   - access important control and observation points,
   - inject faults at a level where fault models are best understood, and
   - determine a confidence level for fault coverage.

Table 1.2 Categories for State of the Art in Validation Methods

Category in Working Group II	Category in Working Group I
1. Logical Proofs	Formal Proofs
2. Analytical Models	Analysis Reliability Modeling
3. Experimental Testing	Testing Simulation/Emulation

Table 3.1 Proposed Validation Task Summary - Category I: Confirmation of System Reliability

Task No.	WG-II Rating	Task Title	Task Objective
I-1	8.4	Reliability Analysis	1- To develop improved mathematical models of system reliability. 2- To estimate reliability characteristics based upon assumed and experimentally-derived failure statistics.
I-2	8.0	Confirmation and Use of Design Proofs	1- To maintain the integrity of design correctness proofs. 2- To use experimental results to confirm proof assumptions. 3- To use proofs to guide experimental tests.
I-3	9.4	Design of Experiments and Analysis of Experimental Data	Estimate parameters and distributions of various aspects for fault handling behavior.
I-4 (Myers)	(*)	Alternative Modeling Techniques	Identify and assess alternative models for computer reliability assessment, such as Petri nets.
I-5 (Meyer)	(*)	Model Solution Methods	To determine methods for solving stochastic performance, reliability and performability models.
I-6 (Hecht)	(*)	Evaluation of Reliability Requirements	To insure compatibility of reliability confirmation with: a) current technology, b) current regulations, and c) observed incidents.

Table 3.1 Proposed Validation Task Summary - Category I: Confirmation of System Reliability (concluded)

Task No.	WG-II Rating	Task Title	Task Objective
I-7 (Murray)	(*)	Performance Confirmation	To develop a state model for reliability analysis which can distinguish between "good" state and "failed" state.
I-8 (Hopkins)	(*)	Performance Analysis	To develop a structural model of performance capability.
I-9 (Melliar-Smith)	(*)	Executive Implementation Proof	To establish at AIRLAB the capability to formally verify the implementation of the executive (and associated programs) against their specification.
I-10 (Melliar-Smith)	(*)	Application Requirements Analysis	To develop methods for formally verifying the specifications of the application tasks against the underlying aerodynamic and structural requirements.
I-11 (Melliar-Smith)	(*)	Application Program Proof	Develop methods for verifying the correctness of application programs by mathematical analysis.

(*) Indicates task not rated by Working Group II.

Table 3.2 Proposed Validation Tasks Summary - Category II: Fault Processing Verification

Task No.	WG-II Rating	Task Title	Task Objective
II-1	8.2	Initial System Check-Out (diagnostic)	Verify that the single processor performs basic functions.
II-2	8.0	Programmer's Manual Validation	1- To identify design errors. To ensure the machine performs functions according to specifications in programmer's manual. 2- To identify and fully characterize incompletely described machine features.
II-3	8.3	Single Processor Executive Validation	1- To determine if executive routines respond as specified. 2- To identify design errors and incompletely specified executive routines.
II-4	8.7	Multiprocessor Interconnection Validation	1- To validate the interconnections between processors. 2- To validate multiprocessor functionality.
II-5	8.7	Multiprocessor Executive and Error Management Validation	1- To determine if multiprocessor and single processor executives respond as specified. 2- To identify design errors or incomplete specification.
II-6	7.6	Application Program Validation	1- To verify that application software responds as specified. 2- To identify design errors. 3- Measure system parameters with time varying inputs.

Table 3.2 Proposed Validation Tasks Summary - Category II: Fault Processing Verification (continued)

Task No.	WG-II Rating	Task Title	Task Objective
II-7	7.1	Simulation of Inaccessible Physical Failures	1- To enhance understanding of relationship between physical faults and their error manifestations. 2- To provide a data base to supplement/ support physical fault injection experiments.
II-8	8.0	Physical Fault Insertion: Single Processor Manifestation Understanding and Preliminary Characterization Histograms	1- To establish functional fault classes to reduce number of faults injected at higher system levels. 2- Generate "representative" system level histograms of detection and isolation times.
II-9	8.4	Physical Fault Insertion: Multiprocessor	Establish fault classes for multiprocessors.
II-10	5.7	Executive Routine Response Characterization Under Single Physical Failure Conditions	1- To classify executive routine responses to hardware failures. 2- To abstract failure manifestations for higher system levels in order to reduce number of fault insertion experiments.
II-11	8.2	Multiprocessor Executive Fault Handling Capabilities	1- To classify multiprocessing response to hardware failures. 2- To measure reconfiguration scheduling algorithm time constant.

Table 3.2 Proposed Validation Tasks Summary - Category II: Fault Processing Verification (continued)

Task No.	WG-II Rating	Task Title	Task Objective
II-12	7.3	Multiprocessor Application Program Fault Handling	1- To classify application software response to hardware failures. 2- To measure system response parameters for failure situations.
II-13	7.1	Multiprocessor Multiapplication Program Fault Handling	1- To characterize system scheduler in failure situations. 2- To classify application programs and system software to hardware failures.
II-14 (Hecht)	(*)	Software Reliability Research	1- To identify severity of manifestations of software failures. 2- To measure software failures as a function of stress.
II-15 (Myers)	(*)	Measurement of Synchronization of Clocks	1- To make an experimental determination of the behavior of functioning clocks. 2- To develop instrumentation capable of distinguishing rare clock failures from instrumentation failures.
II-16 (Melliar-Smith)	(*)	Failure Severity Analysis	To obtain or develop a categorization of failure syndromes against the severity of their consequences.
II-17 (Moses)	(*)	Application Program Validation and Performance Baseline for a System	To validate application software for the system. (Note: Task II-6 addresses computer-only validation.)

Table 3.2 Proposed Validation Tasks Summary - Category II: Fault Processing Verification (concluded)

Task No.	WG-II Rating	Task Title	Task Objective
II-18 (B. Smith)	(*)	Software Fault-Containment Verification	To stress system's ability to contain software errors to affected application programs.
II-19 (Melliar-Smith)	(*)	Logical Analysis of Design to Reduce Size of Test Data Sets	To reduce by logical analysis of the design, the large number of test cases necessary for coverage verification.
II-20 (Mulcare)	(*)	Definition and Use of Application Benchmark Programs	To define, evaluate and employ benchmark applications software for integrated avionics and flight control.
II-21 (Abraham)	(*)	Verification of Operational System Diagnostic Coverage	To verify that faults are detected with periodic diagnosis.

(*) Indicates task not rated by Working Group II.

Table 3.3 Proposed Validation Task Summary - Category III: Fault Processing Characterization

Task No.	WG-II Rating	Task Title	Task Objective
III-1	7.3	False Input Information	To determine the fault-tolerant computer system's response to input information which is plausible (e.g., passes bound checks) but incorrect.
III-2	6.7	Memory Alteration	Selective memory alteration can be used to characterize the robustness of a fault-tolerant system when confronted with a) software and design errors, b) latent faults, c) correlated faults, and d) confusion by divergence.
III-3	8.0	Configuration Control Manipulation	To find the limits of the system when responses are arbitrarily removed.
III-4	6.6	Cold and Warm Start Manipulation	To explore the behavior of the system when its "instinctive" initial start capabilities are stressed.
III-5	7.9	Common Mode Noise and Margin Tests	Characterize the responses of interconnection buses and input/output buses to common mode noise and signals approaching and/or exceeding their boundaries in either the time or amplitude domain.
III-6	7.5	Clock Modification Tests	To determine the clock frequency, shape and synchronization points at which the system fails.

Table 3.3 Proposed Validation Task Summary - Category III: Fault Processing Characterization (continued)

Task No.	WG-II Rating	Task Title	Task Objective
III-7	7.6	Multiple Fault Injections	To characterize the "breaking point" of a fault-tolerant system under multiple faults.
III-8 (Masson)	(*)	Multiple, Sequential Injection of Faults at Potentially Vulnerable Times	To determine the vulnerability of fault-tolerant systems to sequential faults at critical times, such as during reconfiguration.
III-9 (Myers)	(*)	Surprise Assessment Methodology	To avoid some catastrophies by early appreciation of surprises and to promote new approaches by active "harvesting" of surprises.
III-10 (Lindler)	(*)	Evaluation of Strategy for Handling Transient and/or Intermittent Faults	1- Verify predicted system response. 2- Characterize system performance.
III-11 (Seacord)	(*)	Tolerance of Sensitivity to Hardware Parameter Variation	Determine the change in performance or (more likely) fault reaction that will accompany differences in hardware characteristics due to either piece-part tolerance or environment.

Table 3.3 Proposed Validation Task Summary - Category III: Fault Processing Characterization (concluded)

Task No.	WG-II Rating	Task Title	Task Objective
III-12 (Lindler)	(*)	Sensitivity of System to Phase Lag	Determine maximum phase lag the system will tolerant (i.e., phase lag at which given function is not performed satisfactorily).
III-13 (De Feo)	(*)	Effects of Massive Failures	To analyze effects of massive failure like cracked PC boards, lightning strikes, etc.

(*) Indicates tasks not rated by Working Group II.

Table 3.4 Proposed Validation Task Summary - Category IV: Other Tasks

Task No.	WG-II Rating	Task Title	Task Objective
IV-1 (Myers)	(*)	Theoretical Limits of Fault Tolerance	Demonstrate theoretical limits to what faults can be tolerated by a fault-tolerant computer.
IV-2 (Myers)	(*)	Proving Timing Correctness	Provide programming and checking disciplines to guarantee correct timing of program execution.
IV-3 (Lindler)	(*)	Trade-Off of Diagnostic/ Maintenance/Failure Low-Level Isolation Versus Reliability	To provide a general idea of system reliability degradation resulting from increased visibility of details of failures.
IV-4 (Hopkins)	(*)	Instrumentation for Timing Tests	Develop methodology to observe timing relationships on a non-interfering basis.
IV-5 (Masson)	(*)	Sensitivity Analysis/Measurement of a System to Faults	To determine if there are measures of fault manifestation that can be used to describe how sensitive a system is to faults.
IV-6 (Masson)	(*)	Fault-Test Interrelationship Characterization	To understand complex interrelationships among faults and tests. To determine what faults tests cover/detect and what faults invalidate tests.

Table 3.4 Proposed Validation Task Summary - Category IV: Other Tasks (concluded)

Task No.	WG-II Rating	Task Title	Task Objective
IV-7 (Masson)	(*)	Composite Validation	To identify and assess validation techniques which when collected together provide strong (irrefutable) data/evidence regarding reliability.
IV-8 (Masson)	(*)	Error Pattern Classification of Generic Analog/Physical Fault Mechanisms	To determine what various subsystems actually do when disturbed. To determine if certain modes of faulty operation are consistent.
IV-9 (Masson)	(*)	Establishing Generic Fault Classes	To determine the level to which we collapse faults operation. To describe this faults operation.
IV-10 (Myers)	(*)	Instrumenting Fault-Tolerant Computers	To determine specifications and instrumentation requirements so that fault-tolerant computers can be effectively monitored.
IV-11 (Masson)	(*)	Error Patterns on Buses as a High-Level Fault Manifestation	To determine high-level fault manifestations, such as error patterns on buses.

(*) Indicates task not rated by Working Group II.

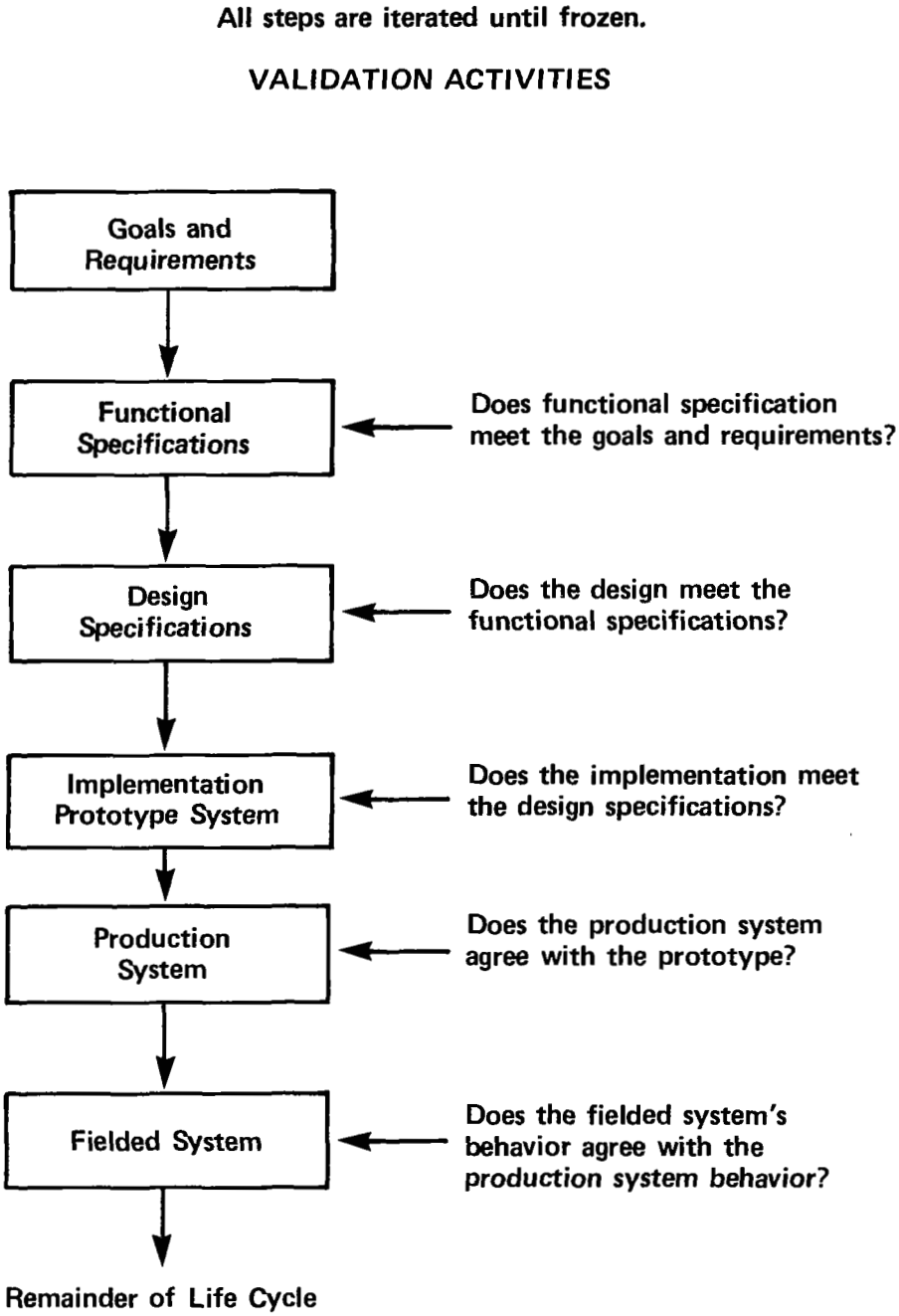


Figure 1.1.- Digital system life cycle.

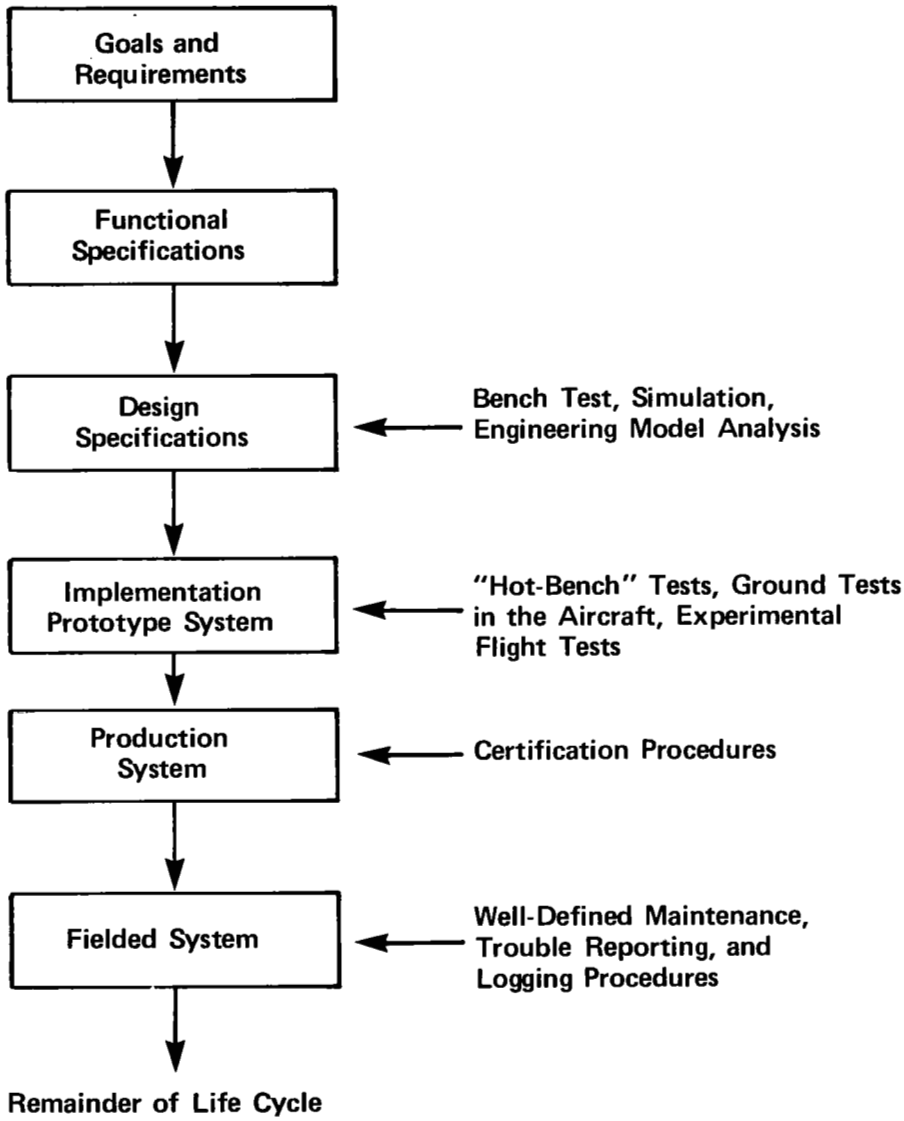


Figure 1.2.- Digital system life cycle applied to aircraft systems.

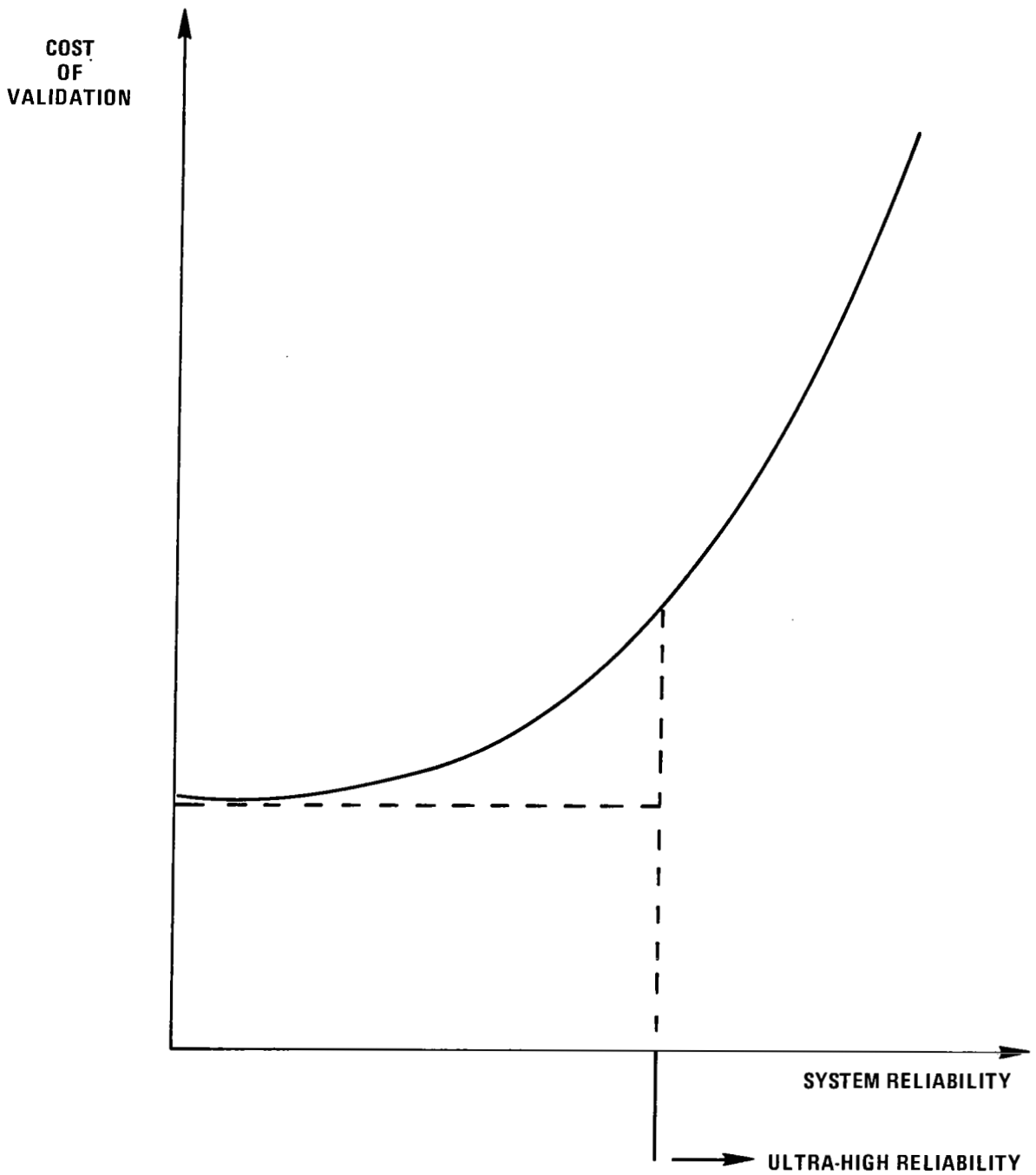


Figure 2.1.- Cost of validation as a function of system reliability assuming a conventional lifetesting approach.

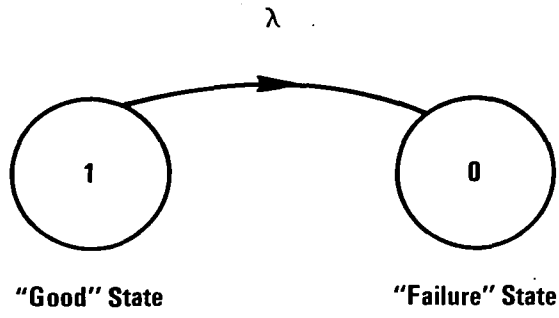


Figure 2.2.- A two-state Markov model of reliability.

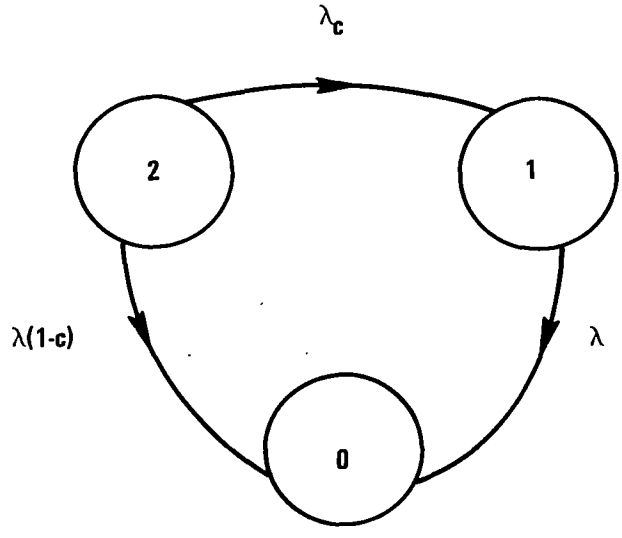


Figure 2.3.- Markov model of a two-unit standby sparing system.

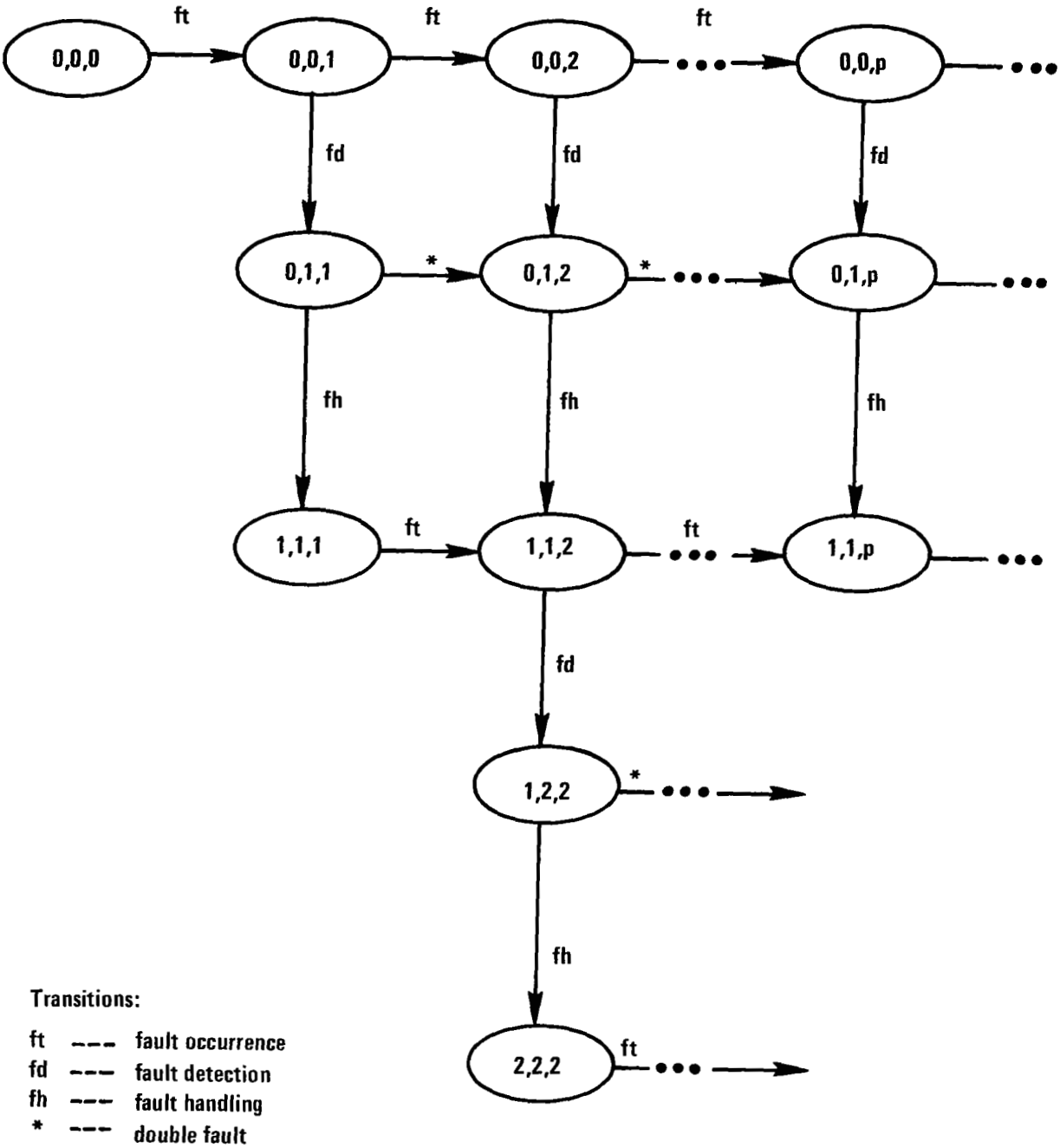
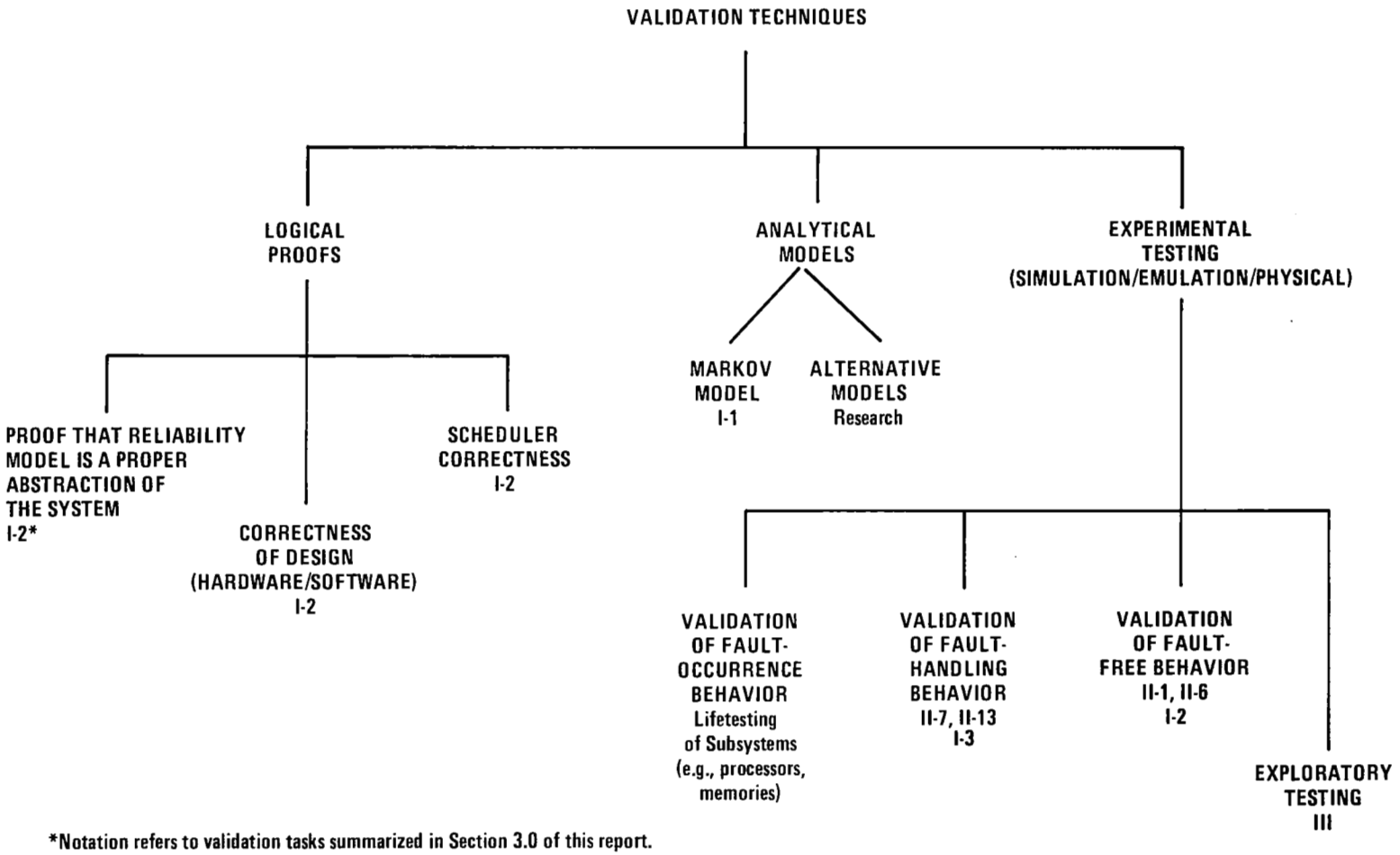


Figure 2.4.- A Markov reliability model of SIFT system.



*Notation refers to validation tasks summarized in Section 3.0 of this report.

Figure 2.5.- The proposed validation taxonomy: Tree form.

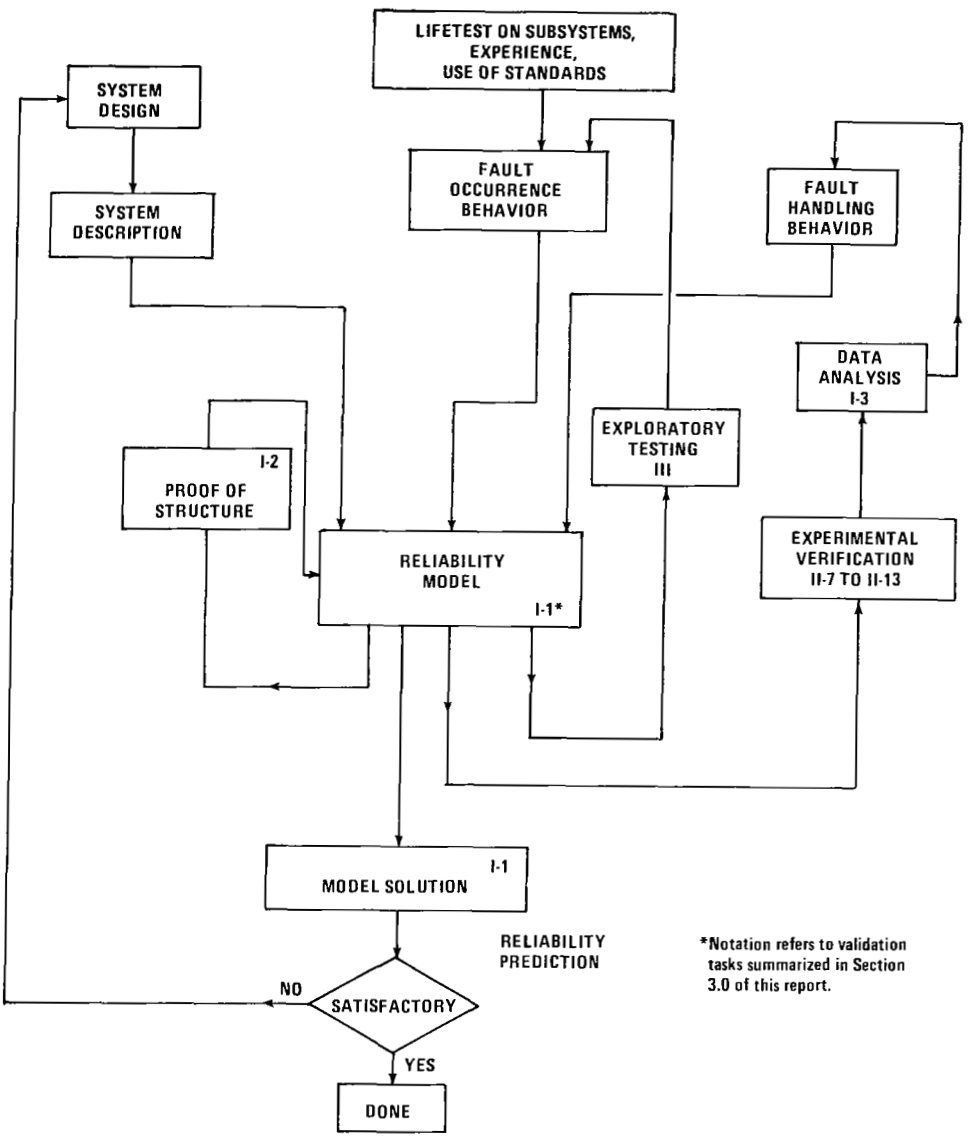


Figure 2.6.- Reliability validation procedure.

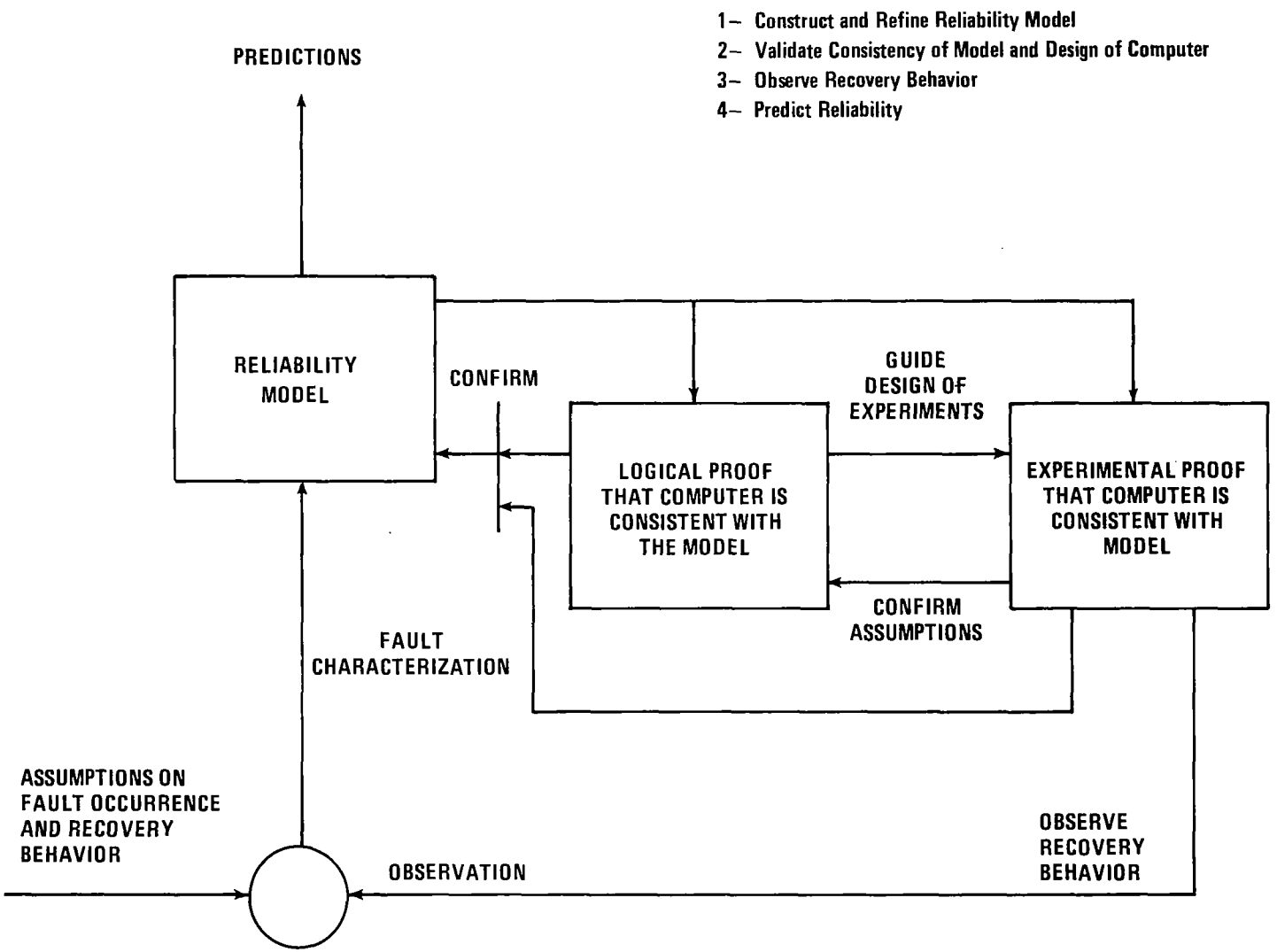


Figure 3.1.- General scheme for confirmation of fault tolerant computer reliability.

TASK NUMBER	IMPORTANCE TOTAL										MEAN
	10	9	8	7	6	5	4	3	2	1	
I-1	11	2	2	1	0	2	1	1	0	0	8.4
I-2	8	3	2	2	0	0	0	2	0	1	8
I-3	12	5	0	1	1	0	0	0	0	0	9.4
II-1	9	2	1	3	2	2	1	0	0	0	8.2
II-2	8	1	2	4	1	2	0	1	0	0	8
II-3	9	1	3	3	1	3	0	0	0	0	8.3
II-4	10	2	3	3	0	2	0	0	0	0	8.7
II-5	10	3	2	3	0	2	0	0	0	0	8.7
II-6	7	0	5	3	2	1	0	1	0	1	7.6
II-7	6	1	4	2	1	4	1	0	2	0	7.1
II-8	10	3	1	1	0	2	1	0	1	1	8
II-9	7	3	6	1	1	2	0	0	0	0	8.4
II-10	4	1	2	1	1	2	2	0	2	3	5.7
II-11	10	0	3	3	0	0	2	0	1	0	8.2
II-12	4	2	6	0	2	3	1	0	1	0	7.3
II-13	4	2	6	1	0	2	2	1	1	0	7.1
III-1	3	2	6	1	2	2	1	0	1	0	7.3
III-2	3	1	3	3	2	5	1	0	1	0	6.7
III-3	4	2	7	3	2	1	0	0	0	0	8
III-4	4	0	3	2	3	0	5	0	1	0	6.6
III-5	5	0	7	1	4	1	0	0	0	0	7.9
III-6	4	3	2	3	3	1	2	0	0	0	7.5
III-7	5	3	4	0	3	0	2	0	1	0	7.6

Figure D.1.- Working group II assessment of the preliminary validation tasks.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CP-2130		

4. Title and Subtitle	5. Report Date
VALIDATION METHODS RESEARCH FOR FAULT-TOLERANT COMPUTER SYSTEMS - WORKING GROUP MEETING II	May 1980
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
James W. Gault, Kishor S. Trivedi, and James B. Clary, editors	L-13716
	10. Work Unit No. 534-02-13-22
9. Performing Organization Name and Address	11. Contract or Grant No.
NASA Langley Research Center Hampton, VA 23665	
	13. Type of Report and Period Covered Conference Publication
12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Washington, DC 20546	

15. Supplementary Notes

James W. Gault: North Carolina State University, Raleigh, N.C.
Kishor S. Trivedi: Duke University, Durham, N.C.
James B. Clary: Research Triangle Institute, Research Triangle Park, N.C.

16. Abstract

The validation process comprises the activities required to insure the agreement of system realization with system specification. This process is particularly challenging when the system is fault-tolerant and intended for life-critical applications. This report documents a preliminary validation methodology for fault-tolerant systems begun by a working group held in the fall of 1979, and sponsored by NASA-Langley Research Center. A general framework for a validation methodology is presented along with a set of specific tasks intended for the validation of two specimen systems - SIFT and FTMP. These systems are being developed for the NASA ACEE Energy Efficient Transport Technology effort and are representative of the most advanced fault-tolerant computer systems. In addition to the description of both general and specific tasks of the validation process, this report identifies tasks in two major areas of research. First are those activities required to support the ongoing development of the validation process itself, and second are those activities required to support the design, development, and understanding of fault-tolerant systems.

17. Key Words (Suggested by Author(s))		18. Distribution Statement
Fault-tolerant computer	Design proof	Unclassified - Unlimited
Software reliability	Testing	
Avionic systems	Reliability models	
Validation process		
Systems reliability		Subject Category 59

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified	104	$6.50

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