General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)
Performability Evaluation of the SIFT Computer

J. F. MEYER
D. G. FURCHTGOTT
L. T. WU

January 1979

Prepared for
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

G. E. Migneault, Technical Officer
NASA Grant NSG 1306

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
SYSTEMS ENGINEERING LABORATORY
THE UNIVERSITY OF MICHIGAN, ANN ARBOR
# TABLE OF CONTENTS

I. INTRODUCTION ........................................... 1

II. PERFORMABILITY MODELING ............................... .5

III. MODELING OF SIFT AND ITS ENVIRONMENT ............ 9

IV. SOLUTION METHODS AND RESULTS ...................... .23

V. REFERENCES ............................................... .36
PERFORMABILITY EVALUATION OF
THE SIFT COMPUTER
by
J.F. Meyer, D.G. Furchtgott and L.T. Wu

Systems Engineering Laboratory
The University of Michigan
Ann Arbor, MI 48109

Abstract - Performability modeling and evaluation techniques are applied to the SIFT computer as it might operate in the computational environment of an air transport mission. User-visible performance of the "total system" (SIFT plus its environment) is modeled as a random variable taking values in a set of "levels of accomplishment." These levels are defined in terms of four attributes of total system behavior: safety, no change in mission profile, no operational penalties, and no economic penalties. The "base model" of the total system is a stochastic process whose states describe the internal structure of SIFT as well as relevant conditions of the environment. Base model state trajectories are related to accomplishment levels via a "capability function" which is formulated in terms of a 3-level model hierarchy. Performability evaluation algorithms are then applied to determine the performability of the total system for various choices of computer and environment parameter values. Numerical results of those evaluations are presented and, in conclusion, some implications of this effort are discussed.

I. INTRODUCTION

Performability modeling and evaluation methods, as introduced in [1], provide a means for quantifying "ability to perform" when system performance is "degradable," that is, depending on the history of the computer's structure and environment during some specified utilization period T, the system can exhibit one of several worthwhile levels of performance (as viewed by the user throughout T). Of particular interest are systems where degraded levels of performance (in addition to
"full degradation" or "failure") are caused, at least in part, by changes in the computer's structure. Typically, such changes are due to faults which occur during utilization and to subsequent structural reconfigurations that are made in the process of fault recovery. Changes in structure may also be due to reconfigurations that are made to accommodate changes in the computer's environment and, particularly, its workload.

The growing interest in such systems is a consequence of the fact that computing systems with distributed hardware and software resources (e.g., multiprocessors, multicomputers, distributed operating systems, distributed databases, etc.) often exhibit this type of degradable performance. In particular, this is true of distributed, fault-tolerant computers that are designed to detect and locate a faulty resource and, through reconfiguration, eliminate its use.

If performance is degradable then, as observed in [1], traditional views of computer "performance" and computer "reliability" are no longer applicable. These views matured in the context of nondegradable performance where, in the presence of structural changes, a system either performs adequately (success) or does not (failure). In this context, "performance" is regarded as successful performance and "reliability" as the ability to perform successfully (probability of success). Accordingly, performance can be evaluated relative to a fixed, fault-free structure (since the system is presumed to perform successfully when it is fault-free) and reliability can be evaluated relative to a structure-based definition of success.
In particular, we see these views reflected by the analytic models that are typically used for computer performance and reliability evaluation. Probabilistic models for performance evaluation (see [2] - [4], for example) represent variations in internal state (e.g., the number of jobs being served or waiting for service in each resource) and workload (e.g., job arrivals) but assume that the structure of the system is fixed (time-invariant). On the other hand, probabilistic models for reliability evaluation (beginning with [5] and continuing through the recent work of [6] and [7]) represent variations in structure (e.g., for each type of resource, the number that remain fault-free) while ignoring the influence of internal state and workload. Although such modeling restrictions are appropriate in the case of nondegradable systems, as argued in [1], more general models are called for when performance is degradable.

As a consequence of these observations, a general modeling framework was introduced [1] which permits the definition, formulation, and evaluation of a unified performance-reliability measure referred to as "performability." The purpose of this paper is to describe the techniques of performability modeling and evaluation in more detail via their application to a specific computer and computational environment. The computer considered is the SIFT (Software-Implemented Fault-Tolerance) computer being developed for the NASA Langley Research Center by SRI International [8], [9]. Its environment is taken to be the control of an advanced (next-generation) commercial aircraft during a transoceanic flight, where such control includes
"active" controls (e.g., active flutter control) and automatic landing control, as well as other, more conventional aircraft control functions. Assumptions regarding the computational requirements of this environment are based on the study made by Ratner, et al. [10].

The choice of a fault-tolerant aircraft computer for this evaluation exercise reflects the principal interest of the NASA Langley Research Center in their support of this research. Between the two fault-tolerant architectures being developed for Langley (SIFT and the Fault-Tolerant Multiprocessor [11], [12] developed by the C. S. Draper Laboratory), the choice of SIFT was due mainly to the availability of information regarding the allocation of tasks to processor-memory units [9]. These task allocations, particularly in degraded modes of operation, indicate how the structure of SIFT relates to the accomplishment of aircraft functional tasks. Moreover, the attributes used to distinguish the "criticalities" of functional tasks [10] support a natural definition of "accomplishment levels" for the total system.

In Section II of the paper we give a brief review of the basic concepts and terminology of performability modeling [1]. Section III describes the construction of a hierarchical model for the SIFT computer in the computational environment specified above. The concluding section (Section IV) summarizes the solution methods used to determine the performability values and presents numerical results of the evaluation for various choices of computer and environment parameter values.
II. PERFORMABILITY MODELING

A computing system, as it operates in its use environment, may be viewed at several levels. At a low level, there is a detailed view of how various components of the computer's hardware and software structure behave throughout the utilization period. At this level, there is also a detailed view of the behavior of the computer's "environment" where by this term we mean both demands (workload) imposed by users and peripheral systems, and natural conditions (e.g., weather, radiation, etc.) which can influence the system's performability. The computer, together with its environment, is referred to as the total system. A second (and usually much less detailed) view of the total system is the user's view of system behavior, that is, what the system accomplishes for the user during the utilization period. A third view (which may coincide with the second) is the economic benefit derived from using the system, that is, the computing system's "worth" (as measured, say, in dollars) when operated in its use environment. Performability evaluation is concerned with quantifying a system's ability to perform when it is viewed at the user interface (the second view) and hence questions of economic benefit may be avoided if desired. On the other hand, if economic benefit is the primary concern, performability can be evaluated in these terms by placing the user interface at this level (i.e., by identifying the second and third views).

To formally represent these views, let $S = (C, E)$ denote the total system where $C$ is the computer and $E$ is its environment.
(Although C is referred to as the "computer," it should be
generally interpreted as that part of the total system which
is the object of the evaluation, that is, the part which lies
within the "system boundary"; see [2], [3], for example.)
As is typically done in probabilistic modeling, the low level
view of S is modeled as a stochastic process $X_S$ defined over a
time period $T$ called the utilization period. The process $X_S$
is referred to as the base model of S. Each random variable $X_t$
($t \in T$) of the base model $X_S$ takes on values in a state space $Q$,
where a given state in $Q$ represents a particular status of both
the computer and its environment. More precisely, $Q = Q_C \times Q_E$
where $Q_C$ is the state space of the computer and $Q_E$ is the state
space of its environment. Moreover, a state $q \in Q_C$ may describe
both the structural configuration of the computer and the internal
state of that structure. An instance of the base model's behavior
is a state trajectory $u: T \rightarrow Q$ where $u(t) = X_t$, the state of $S$
at time $t$. Finally, the collection of all possible state trajectories is denoted $U$ and is referred to as the trajectory space
of $S$. (For a more detailed and more precise development of these
concepts, the reader should refer to [1].)

Regarding the second, user-oriented view of the total system,
we assume that the user is interested in distinguishing a number
of different levels of accomplishment when judging how well
the system has performed throughout the utilization period $T$
(one such level may be total system failure). The user's
"description space" is thus identified with an accomplishment
set $A$ whose elements are referred to alternatively as accomplishment
levels or (user-visible) performance levels. Accordingly, the user's view of total system behavior is modeled by a random variable $Y_S$ taking values in the accomplishment set $A$. $Y_S$ is referred to as the performance of $S$. In the terminology of computer performance evaluation, $Y_S$ can be regarded as any user-oriented performance "measure" or "index" that summarizes the behavior of $S$ throughout the utilization period $T$. Thus, "average response time" (averaged over $T$) could be regarded as a performance variable $Y_S$ whereas "response time" would not.

Given this representation of user-visible performance, a natural measure which quantifies the "ability to perform" is the probability distribution function of the performance variable $Y_S$. In case the accomplishment set $A$ is discrete (which is the only case we have considered in the context of aircraft computer evaluation) the probability (mass) function of $Y_S$ suffices, that is, the performability of $S$ is the function $p_S:A \rightarrow [0,1]$ where

$$p_S(a) = \text{the probability that } S \text{ performs at level } a \quad (1)$$

In constructing a model that can support an evaluation of performability, we assume that enough is known about the probabilistic nature of the computer's structure and environment to permit the specification of the base model, i.e., the stochastic process $X_S$. At the outset, on the other hand, little if anything is known about the performance variable $Y_S$, except that it takes on values in a designated accomplishment set $A$. Thus, to determine
the probabilistic nature of $Y_S$ (and hence the performability $p_S$), we must establish how state trajectories of the base model $X_S$ relate to the accomplishment levels of the performance variable $Y_S$. We refer to this relationship as the capability function of $S$ which, in terms of the notation introduced above, is defined as a function $\gamma_S$ from the trajectory space $U$ into the accomplishment set $A$. If $u \in U$ then $\gamma_S(u)$ is interpreted as the level of accomplishment (performance) that results when the state trajectory is $u$. (For a more detailed discussion of capability functions and their properties, see references [1] and [13].)

Once the base model $X_S$ is specified, the essential problem in performability modeling is to formulate the capability function $\gamma_S$ or, more precisely, its inverse $\gamma_S^{-1}$. The technique we have used to solve this problem is to elaborate the base model into a model hierarchy, permitting a decomposition of $\gamma_S$ into interlevel translations [1]. Once the capability function is formulated, model solution is basically a two-step procedure:

1. For each accomplishment level $a$ in $A$, determine the set of all state trajectories that result in $a$, that is, determine the inverse image $U_a = \gamma_S^{-1}(a)$.

2. Using the base model $X_S$, for each $a$ in $A$, compute the probability of the trajectory set $U_a$ (which is equal to the performability value $p_S(a)$).

Although the above review of performability modeling is somewhat brief, it will hopefully serve as an adequate guide
for the discussion that follows. Moreover, these various concepts should acquire more meaning once they are illustrated in the context of a specific application.

III. MODELING OF SIFT AND ITS ENVIRONMENT

The concepts, models and methods described above appear to be applicable to a variety of systems (both man-made and natural) wherein performance may degrade with changes in the system's structure and environment. The primary motivation for this development, however, has been to evaluate the performability of aircraft computing systems of the type envisioned for next-generation commercial aircraft. Within this context, the sections that follow describe a relatively comprehensive performability modeling and evaluation exercise.

In the terminology of the previous section and for reasons discussed in the introduction, the system considered is the total system \( S = (C,E) \) where \( C \) is the SIFT computer [8],[9] and \( E \) is a transoceanic flight of an advanced commercial aircraft. Assuming that the user is the airline that owns the aircraft, the user's view of desired total system performance can be stated quite simply: "Transport passengers from airport A to a transoceanic airport B, safely, directly, and with minimum operational and economic penalties." Examining this statement in more detail, total system performance can be described in terms of four attributes: safety, no change in mission profile, no operational penalties, and no economic penalties. (See [10] for a more precise interpretation of these
attributes, used there to distinguish the "criticalities" of various aircraft functional tasks.)

To determine the accomplishment set $A$ for the performance variable $Y_S$ we assume that safety is the most important attribute, i.e., safe flights have the greatest worth, the remaining attributes being worth successively less in the order they are listed. (These assumptions regarding relative worths conform with the "reliability requirements" specified in [10] for the corresponding criticality levels.) Assuming further that safety is worth considerably more than no change in mission profile, which in turn is worth considerably more than no operational penalties, etc., the following accomplishment set suffices to describe the performance levels of interest to the user:

$$A = \{a_0, a_1, a_2, a_3, a_4\}$$

where

- $a_0 =$ no economic penalties, no operational penalties, no change in mission profile, and no fatalities
- $a_1 =$ economic penalties, no operational penalties, no change in mission profile, and no fatalities
- $a_2 =$ operational penalties, no change in mission profile and no fatalities
- $a_3 =$ change in mission profile, and no fatalities
- $a_4 =$ fatalities.

Accordingly, the performance of $S$ (see Section II) is a random variable $Y_S$ taking values in the accomplishment set $A$ specified above. The performability $p_S$, which we seek to evaluate, is the probability (mass) function of $Y_S$.

To construct a base model $X_S$ that can support an evaluation of $p_S$, the state spaces $Q_C$ of SIFT and $Q_E$ of its environment must be refined enough so that each state trajectory
u:T + Q_c \times Q_e

of the process X_S (see Section II) results in a uniquely determined accomplishment level a_i \in A. In other words, the trajectory space U must admit to the formulation of a capability function \gamma_S:U + A. On examining the architecture of the SIFT computer, whose general organization is depicted in Figure 1, we find that it suffices (with one exception to be discussed later) to know the number of processor-memory units, and the number of busses which are fault-free. In other words, a state of q \in Q_c can be expressed as an ordered pair

q = (i,j)

where i is the number of fault-free processor-memory units and j is the number of fault-free busses. Regarding the environment, we find that the weather condition at the destination airport is an influential variable and, under reasonable assumptions, the only environmental variable that need be considered. (Other environmental factors, such as the duration of the utilization period T, are fixed for a specific total system and hence are regarded as parameters rather than variables.) Accordingly, the state space of the environment is taken to be the two-element set Q_e = \{0,1\} interpreted as follows:

1: Zero visibility (Category III) weather at the destination airport

0: Not 1.
Regarding the utilization period $T$, we assume that the utilization of SIFT is continuous from ramp departure of the aircraft to ramp arrival at the destination airport. More precisely, taking the departure time to be $t=0$, if $h$ is the duration of utilization (in hours) then $T$ is the closed real interval

$$T = [0,h] = \{t|0 \leq t \leq h\}.$$  

Accordingly, the base model is a stochastic process

$$X_S = \{X_t|t \in [0,h]\}$$  

where each random variable $X_t$ takes values in the state space $Q_C \times Q_E$. If further, we let $X_{C,t}$ and $X_{E,t}$ denote the projections of $X_t$ on $Q_C$ and $Q_E$, respectively, it is reasonable to assume that the processes

$$X_C = \{X_{C,t}|t \in [0,h]\} \quad (4)$$

and

$$X_E = \{X_{E,t}|t \in [0,h]\} \quad (5)$$

are (statistically) independent. (What we are saying here is that the number of fault-free resources in SIFT is independent of the weather condition at the destination airport. This should not be confused with how the use of SIFT's resources depends on the weather; the latter type of dependence is "functional" [13] and is determined by the nature of the capability function $\gamma_S$.) Thus the base model $X_S$ is determined once we specify the probabilistic nature of the stochastic processes $X_C$ and $X_E$. It is convenient, however to defer these details to the subsequent discussion of a hierarchical model of $S$.

In general, to facilitate the description of the capability function, we have proposed the use of a model hierarchy (see
which, proceeding from the top down (the "top" model is closely related to the performance variable $Y_S$), consists of a sequence of models describing the total system in successively more detail. The "bottom" model of the hierarchy is comprised of those components of the base model which cannot be introduced directly at higher levels.

For the system in question, we find it convenient to introduce three levels of detail (abstraction) and refer to them, respectively as the "mission level" (level 0), the "aircraft level" (level 1), and the "computer level" (level 2). Following the terminology and notation of [1], the model at level $i$ ($0 \leq i < \infty$) is a stochastic process $X^i$ defined in terms of a composite process $X^i_c$ and basic process $X^i_b$. (The composite process inherits its behavior from the level $i+1$ model; the basic process does not, i.e., it is a component of the base model process $X_S$.) The trajectory spaces of $X^i_c$ and $X^i_b$ are denoted $U^i_c$ and $U^i_b$, respectively, and trajectories in $U^i_c \otimes U^i_b$ determine trajectories in $U^{i-1}_c$ ($i \geq 1$) via an interlevel translation $\kappa_i$. (When $i=0$, $\kappa_0$ is a function from level 0 trajectories into the accomplishment set $A$.)

This notation is summarized in Figure 2 which depicts the model hierarchy for $S$ and its relation to the performance variable $Y_S$. (It is helpful to compare Figure 2 with Figure 1b in [1], where the latter depicts the general form of a model hierarchy.) The specific nature of the hierarchy in question is described in the subsections that follow.
Mission Level (Level 0)

The model at this level, which is the "top" of the model hierarchy, is close to the accomplishment set \( A \) and simply formalizes the attributes used earlier to distinguish accomplishment levels. More precisely, we take the level 0 state space to be the set

\[
Q^0 = \{0,1\}^4
\]

where the four coordinates are referred to as ECONOMICS, OPERATIONS, PROFILE and SAFETY, respectively. A coordinate value of 0 denotes the presence of the corresponding attribute; 1 denotes its absence. Thus, for example, the state

\[
q = (1,0,1,0)
\]

says that the flight incurred economic penalties, no operational penalties, a change in mission profile, and was safe. Moreover, we assume that all the coordinates are "composite," that is their values will be uniquely determined by a state trajectory of the level 1 model. Hence \( Q^0 = Q_c^0 \) and accordingly (by definition) all the state trajectories of the level 0 model will be composite. (This explains the lack of a basic trajectory space \( U_b^0 \) in Figure 2.) Moreover, since we are modeling the outcome of the mission after utilization is completed (i.e., at time \( t = h \)) the level 0 model is a single-variable random process

\[
X_c^0 = \{x_c^0\}
\]

with a trajectory space that coincides with the state space, i.e.,

\[
U^0 = U_c^0 = Q_c^0 = \{0,1\}^4
\]

(Note that the term "trajectory" is somewhat misleading in this
instance; however, we prefer to maintain a common vocabulary that applies to any level of the hierarchy.) Given the interpretation of \( Q^0 \) (and hence \( U^0 \)) specified above, the translation \( \kappa_0: U^0 \rightarrow A \) is determined immediately from the definitions of the accomplishment levels (see (2)). The function table of \( \kappa_0 \) is shown in Table 1 where * denotes that the variable value can be either 0 or 1.

Aircraft Level (Level 1)

The model at this level of the hierarchy describes the extent to which various aircraft functional tasks can be accomplished during various phases of the flight. The environment part of the base model (i.e., the weather condition at the destination airport) is also introduced at this level. The latter is possible since task allocation priorities in the SIFT computer (see [12]) are weather independent. Note, however, this does rule out "use" dependencies of the type referred to earlier, e.g., computations required for an automatic landing are not used during clear weather. Such dependencies are captured by the translation of level 1 trajectories into level 0 trajectories, which will be discussed momentarily.

The aircraft functional tasks considered are a representative subset of those identified in [10] and subsequently added to and modified in [9]. More specifically, we make the following assumptions regarding the aircraft, where the functional tasks considered (a total of 8) are signified by capital letter names or acronyms.
a) The aircraft has an Aircraft Integrated Data System (AIDS) which continuously executes in-flight analyses of various on-board data. This information is economically useful to the airline for assessing aircraft performance and for scheduling maintenance. Hence, "loss" of AIDS is assumed to result in economic penalties. (By the "loss" of a functional task we mean the inability to accomplish that task.)

b) The aircraft has two means of navigation. The first involves an inertial guidance system (INERTIAL), while the second means involves an air data system (AIR DATA) along with two radio beacon systems: Very-High Frequency Omirange (VOR) and Distance Measuring Equipment (DME). (Support of VOR and DME is regarded as a single functional task denoted VOR/DME.) We assume that the signals generated by the VOR/DME system will not be receivable by aircraft more than 250 nautical miles from a transmitting station, and in particular, more than 250 nautical miles from land. The AIR DATA task is required to support the VOR/DME task.

c) If INERTIAL is lost before the aircraft enters a region where it cannot receive VOR/DME signals (especially an oceanic region on a transoceanic mission), it will return to its origin. We make the simplifying assumption that if the aircraft must make such a diversion, it returns safely to its origin with no further incidents. Such a diversion is considered a change in mission profile.

d) If INERTIAL is lost while the aircraft is out of range of the VOR/DME system, the aircraft loses its navigational capability. Likewise, loss of INERTIAL along with VOR/DME or AIR DATA results in a loss of navigational capability. These losses are assumed to result in a change in mission profile.

e) Loss of VOR/DME or AIR DATA tasks results in economic penalties.

f) The aircraft has an autoland system (AUTOLAND) which, if operational, will land the plane in any weather. The AUTOLAND system requires the results of INERTIAL computations as well as AUTOLAND computations. If, just prior to initiation of landing, the destination airport has Category III weather and the aircraft does not have AUTOLAND capability then a diversion is made to another airport. Such a diversion is considered a change in mission profile.
g) If, just prior to the initiation of landing, the destination airport has Category III weather and the aircraft has the AUTOLAND capability, AUTOLAND is used. Loss of AUTOLAND during such a landing will cause the plane to crash, resulting in an unsafe mission.

h) The aircraft has active flutter control (ACTIVE FLUTTER CONTROL), attitude control (ATTITUDE CONTROL), and engine control (ENGINE CONTROL) functions, all of which are critical to the airworthiness of the plane. Loss of any of these functions results in an unsafe mission.

Given the above assumptions regarding the aircraft, we find that computer behavior, when viewed at the aircraft level, can be represented as SIFT's ability to accomplish (via execution of required computational tasks) each of eight aircraft functional tasks:

<table>
<thead>
<tr>
<th>Task</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AIDS</td>
<td>AIR DATA</td>
<td>VOR/DME</td>
<td>INERTIAL</td>
<td>AUTOLAND</td>
<td>ACTIVE FLUTTER CONTROL</td>
<td>ENGINE CONTROL</td>
<td>ATTITUDE CONTROL</td>
</tr>
</tbody>
</table>

during each of four phases of the utilization period:

<table>
<thead>
<tr>
<th>Phase</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Takeoff/cruise until VOR/DME out of range</td>
<td>Cruise until VOR/DME in range again</td>
<td>Cruise until landing is to be initiated</td>
<td>Landing</td>
</tr>
</tbody>
</table>

Accordingly, the trajectory space $U^1_c$ of the composite process $X^1_c$ can be conveniently represented by the set of all $8 \times 4$ matrices

$$u = [q_{i,j}]$$

where, except for coordinates $(5,1)$ and $(5,2)$,

$$q_{i,j} = \begin{cases} 
0 & \text{if task } i \text{ can be accomplished throughout phase } j \\
1 & \text{otherwise}.
\end{cases}$$

(6)
In the case of coordinates (5,1) and (5,2), since we know that task 5 (AUTOLAND) need not be accomplished during phases 1 and 2, "q5,1" and "q5,2" are assigned a constant value "∅". During phase 3, the AUTOLAND task is interpreted as the checkout of the autoland system (prior to its possible use during landing).

To permit level 1 trajectories to be translated into level 0 trajectories (i.e., to determine the interlevel translation \( \kappa_1 \)), the level 1 model must also convey the weather condition at the destination airport just prior to the initiation of the landing phase (see assumptions f) and g)). Accordingly, the basic part of the level 1 model is taken to be the environment model \( X_E \) (see (5)) when sampled at the end of phase 3. More precisely, if phase 3 ends at time \( t_3 \), then the basic part is the (degenerate) process.

\[
X^1_b = \{X_E, t_3\}
\]

Extending the time base of \( X^1_b \) to that of \( X^1_c \), the combined trajectory space \( U^1 = U^1_c \otimes U^1_b \) can be represented by the set of all \( 9 \times 4 \) matrices

\[
u = [q_{i,j}] \quad (1 \leq i \leq 9, 1 \leq j \leq 4)
\]

where the first 8 rows represent trajectories in \( U^1_c \) (as specified above), \( q_{9,3} \in O_E \) (see (3)), and \( q_{9,1} = q_{9,2} = q_{9,4} = ∅ \).

Given this representation of level 1 state trajectories and under assumptions a)–h) stated above, the interlevel translation \( \kappa_1 : U^1 \rightarrow U^0 \) can then be specified. In general, however, when specifying an interlevel translation \( \kappa_1 \), we
seek to avoid a complete tabulation of the values $k_i(u)$ for each trajectory $u \in U_i$ since, as we move down a model hierarchy, the size of the trajectory sets $U_i$ can become unmanageable large. (Note that, even in the case of the small space $U^0 = \{0,1\}^4$, we avoided complete tabulation through use of the symbol "*".) In response to this need, we have developed a general method (see [14]-[15]) for specifying the $k_i$ in a form that is feasible for large trajectory spaces and is suited to solution methods for computing performability.

Although space does not permit a detailed description of this specification method, its application to the interlevel translation $k_1$ can be summarized as follows. $k_1$ is first decomposed into its projections onto the individual coordinates of $U^0 = \{0,1\}^4$, i.e., into the functions $\xi_i k_1$ ($1 \leq i \leq 4$) where, if $u \in U^0$, $\xi_i(u)$ is the value of the $i^{th}$ coordinate of $u$. ($\xi_i k_1$ denotes the composition of functions $k_1$ and $\xi_i$, first applying $k_1$.) Each function $\xi_i k_1$ is then specified by specifying the inverse image $(\xi_i k_1)^{-1}(v)$ for each value $v \in \xi_i(U^0) = \{0,1\}$. However, instead of tabulating all the trajectories in this preimage (which is a subset of $U^1$), it is expressed as a disjoint union of "Cartesian" subsets of $U^1$.

(Since trajectories in $U^1$ are represented by matrices (6), a Cartesian subset $V$ of $U^1$ can be regarded as a $9 \times 4$ matrix whose entries are the component sets of the Cartesian product $V$.) This representation thus parallels the use of "subcubes" to represent switching functions (see [16], for example) although, in general, we allow the coordinate values to be
elements of an arbitrary finite set (the state set of the level
i model).

To illustrate part of the specification of \( \kappa_1 \), suppose
i=3 (the PROFILE coordinate of \( U^0 \)) and \( v=0 \) (no change in mission
profile). Then the corresponding set of level 1 trajectories
is the union of three Cartesian subsets of \( U^1 \), that is,

\[
(\xi_3\kappa_1)^{-1}(0) =
\begin{bmatrix}
\ast \ast \ast \ast \\
\ast \ast \ast \\
0 0 0 \ast \\
\ast \ast \ast \\
\ast \ast \ast \\
\not\ast \ast \ast \\
0 0 1 \ast \\
\ast \ast \ast \\
\ast \ast \ast \\
\not\ast \ast \ast \\
\end{bmatrix}
\begin{bmatrix}
\ast \ast \ast \\
\ast \ast \ast \\
0 0 0 \ast \\
\ast \ast \ast \\
\ast \ast \ast \\
\not\ast \ast \ast \\
0 0 1 \ast \\
\ast \ast \ast \\
\ast \ast \ast \\
\not\ast \ast \ast \\
\end{bmatrix}
\begin{bmatrix}
\ast \ast \ast \\
\ast \ast \ast \\
0 0 0 \ast \\
\ast \ast \ast \\
\ast \ast \ast \\
\not\ast \ast \ast \\
0 0 1 \ast \\
\ast \ast \ast \\
\ast \ast \ast \\
\not\ast \ast \ast \\
\end{bmatrix}
\]

where 0, 1, \( \not\ast \) and \( \ast \) denote the sets \{0\}, \{1\}, \{\not\ast\} and \{0,1\},
respectively. The complete specification of \( \kappa_1 \), along with a
more detailed discussion of its derivation, can be found in [15].

Computer Level (Level 2)

The model \( X_b^2 \) at the bottom of the hierarchy is the
computer component of the base model process \( X_s \), i.e., the
stochastic process \( X_c \) identified earlier in the discussion
(see (4)). In determining the specific nature of \( X_c \), many
of the issues to be resolved are similar to those encountered
in reliability modeling (see [5] - [7], for example) and,
in particular, those addressed by SRI in their investigation of
reliability models for SIFT (see [9], Section VII). Since
the emphasis here is on needs that are peculiar to performability
modeling, our construction of \( X_c \) is based on a relatively
idealized Markov model of SIFT (referred to in [9] as "Model I")
where faults are assumed to be permanent and reconfiguration
times are assumed to be instantaneous. On the other hand, to
deal with performance issues such as the effect of different computational demands (workloads) during different phases of the flight, the utilization period $T$ is decomposed into eight phases at level 2 (see Table 2). Within a given phase, we take the process $X_C$ to be a time-homogeneous Markov process similar to SRI's Model I. However, we generally permit these intraphase processes to differ from phase to phase, where the probabilities of interphase state transitions (which take place at the time of a phase change) are specified by interphase transition matrices (see [14], [15]). Assuming a maximum of $n$ processors (i.e., processor-memory units) and $m$ busses, the intraphase Markov process assumed for all phases except the takeoff phases is given by the transition graph of Figure 3. For the takeoff phase, state pairs $(2, j)$ and $(2', j)$ are identified for all $j$, in which case the model reduces to SRI's Model I ([9], p. 151, Figure VII-2). The need for the states $(2', j)$ during phases 2-8 is to distinguish whether a particular processor is reduced from 3 to 2. (The fact that processors do not look alike when only three remain fault-free is a consequence of task allocation constraints which will be discussed momentarily.

Given the computer level model $X^2_b = X_C$, it remains to specify how trajectories in $U^2_b$ (variations in the structure of SIFT) translate via $\kappa_2$ into trajectories in $U^1_c$ (variations in SIFT's ability to accomplish aircraft functional tasks). Such a specification is based primarily on how functional tasks or, more precisely, the computational tasks
that support them, are allocated to a given number of fault-free processors. Assuming that each processor has a capacity of 0.16 MIPS (millions of instructions per second) and each memory has 5 kilowords of storage (these assumptions are scaled down from those of [9] since we are considering a reduced number of functional tasks), this allocation is determined by an algorithm (see [15]) similar to the one employed in [9]. As a consequence, for each phase of the level 2 model, we are able to specify which functional tasks are lost (cannot be accomplished by SIFT) as a function of the number of fault-free processors. This information is summarized in Table 3.

The information in Table 3, along with the assumption that communication among any number of processors is insured as long as at least two busses remain fault-free (see [9]), suffices to determine the interlevel translation \( K_2 \). Because busses do not play an essential role when at least 2 remain fault-free, it suffices to specify \( K_2 \) for trajectories over the state space

\[
\tilde{Q} = \{1, 2, 2', 3, 4, 5, 6\}
\]

where, in terms of the states of the bottom model,

\[
\begin{align*}
1 &= \{F\} \\
i &= \{(i, j) \mid 2 \leq j \leq m\} \text{ if } i = 2, 2', 3, 4, 5 \\
6 &= \{(i, j) \mid 6 \leq i \leq n, 2 \leq j \leq m\}
\end{align*}
\]

Accordingly, trajectories over \( \tilde{Q} \) are represented by the set of all \( 1 \times 8 \) matrices

\[
u = [q_k] \quad (1 \leq k \leq 8)
\]
where

\[ q_k = 1 \quad \text{if the structural state of SIFT is in set } i \text{ (i.e., Q) at the end of phase } k. \]

(During phase 1, states 2 and 2' are identified since, according to Table 3, there is no need to distinguish them.) The method used to specify \( \kappa_2 \) is identical to that used at level 1, except that 32 coordinates must now be accounted for instead of 4. A full specification of \( \kappa_2 \), obtained by this method, is described in [15]. Having established all the ingredients of the hierarchy (Figure 2), our modeling of \( S \) is complete.

IV. Solution Methods and Results

As outlined in the concluding paragraph of Section II, once a performability model has been constructed for a system \( S \), the computation of its performability \( p_S \) (see (1), Section II) is basically a two-step procedure. The first step relies on a knowledge of the capability function \( \gamma_S \) and, for each accomplishment level \( a \in A \), yields an appropriate representation of all the base model state trajectories that result in \( a \), i.e., all trajectories in the set \( U_a = \gamma^{-1}_S(a) \). The second step relies on a knowledge of the probabilistic nature of the base model \( X_S \) and, for each trajectory set \( U_a \), yields the performability value \( p_S(a) \).

The problems encountered in carrying out these steps are both interesting and challenging since, in effect, they are generalized versions of problems currently being dealt with in the more specific contexts of performance evaluation and reliability evaluation. Our work to date concerning each of these
steps has been carried to the point where models of moderate complexity, such as the one just described in the previous section, can be solved without an undue amount of effort. Certain of the algorithms used, particularly in the second step, have been implemented by programs that reside in a prototype software package called METAPHOR (Michigan Evaluation Aid for Performability). Other algorithms, which have not yet been programmed, can fortunately be carried out manually, although the effort required is somewhat tedious and laborious.

Since space does not permit discussion of these methods, we can only outline the underlying ideas and point, as we did in Section III, to some recent technical reports for further information. Regarding the first step, i.e., the determination of the trajectory sets $U_a$, the algorithm used here is based on the fact that $\gamma^{-1}_S$ can formulated in terms of the inverses of the interlevel translations $\kappa_i$. Thus, for the hierarchy in question (Figure 2), $\gamma^{-1}_S(a)$ is computed by first determining $\kappa_0^{-1}(a)$, and then applying $\kappa_1^{-1}$ followed by $\kappa_2^{-1}$. An important feature of this algorithm is that it manipulates Cartesian representations of the type illustrated in Section III (see (7)). Moreover, the trajectory sets determined at each level of the hierarchy are always expressed as disjoint unions of Cartesian subsets. Details concerning the derivation and application of this algorithm can be found in [15].

The second step of the solution procedure computes the probability of each base model trajectory set $U_a$. The algorithm requires that $U_a$ be expressed as a disjoint union of Cartesian
components, but this is automatically provided by the output of step 1. The probability of each Cartesian component is then computed using a specially developed algorithm that involves the "intraphase" and "interphase" transition matrices of the base model [14]. Summing the probabilities of these components yields the performability value \( p_S(a) \) and, when this is done for each level \( a \), the computation of \( p_S \) terminates.

Applying these algorithms, the performability of SIFT was evaluated for a number of specific instances of the total system model described in Section III. An instance of the model is obtained by fixing the values of the following computer and environment parameters:

**COMPUTER (SIFT)**

C1) Hardware resources, that is, the number of processors \( n \) and the number of busses \( m \) (see Figures 1 and 3).

C2) Hardware failure rates, that is, the processor failure rate \( p \), and the bus failure rate \( q \) (see Figure 3).

C3) Initial state distribution, that is, the probability distribution of the random variable \( X_{C,0} \) (see (2)).

**ENVIRONMENT**

E1) Flight duration \( h \) and phase durations.

E2) Probability of Category III weather at destination airport.

Evaluations were based on the following selection of parameter values:

C1) \( n = 6 \) and \( m = 6 \).

C2) As in [9], we assume that \( p = 10^{-4} \) and \( q = 10^{-5} \) (failures per hour).
C3) Two types of initial state distributions are considered. The first type is "deterministic" in the sense that one computer state has probability 1 of being the initial state (the remaining states having probability 0). If \( (i,j) \) is the state having probability 1, this distribution is denoted 
\[ \text{Det}(i,j). \]
The second type of initial state distribution considered is truly probabilistic where one of two specific distributions are assumed. These are denoted \( I_1 \) and \( I_2 \) and are given in Table 4.

E1) Two flight missions are considered, a 6 hour and 25 minute flight from London to New York (JFK Airport) and a 10 hour flight from Tel Aviv to New York. The assumed phase durations associated with each flight are given in Table 5.

E2) The probability of Category III weather at JFK is taken to be 0.011 (see [17], p. 173). For the fixed values of \( C_1, C_2, \) and \( E_1 \) indicated above and for choices \( C_3 \) and \( E_1 \) as indicated in Tables 4 and 5, 14 specific systems were evaluated (denoted \( S_1, S_2, \ldots, S_{14} \)). For each system \( S_i \), the results of the performability evaluation are tabulated in Table 6, where the entry corresponding to system \( S_i \) and accomplishment level \( a_j \) is the probability \( p_{S_i}(a_j) \).

On examining Table 6, we see that a performability evaluation provides the user with a "spectrum" of numbers which quantifies degradable performance when viewed at the user interface. Although the user's primary concern, in this case, is safety, if the probability \( p_{S_i}(a_4) \) of an unsafe flight is acceptably low, the performability at safe levels (levels \( a_0-a_3 \)) is also a legitimate concern of the user. Moreover, we believe that the design of an aircraft computer should reflect this concern, that is, performability should be accounted for by design algorithms (e.g., the allocation and scheduling of computational tasks) and should be evaluated
in the process of assessing design alternatives.

Although the results given by Table 6 are interesting in themselves, we will resist the temptation to interpret this data since our intent here is not to critique the design of the SIFT computer. Instead, the purpose of this study has been to demonstrate the feasibility of performability modeling and evaluation and to illustrate the type of results that can be obtained. We believe that this has been accomplished and, moreover, we hope that the preceding discussion has helped to clarify the kind of modeling concepts and solution techniques needed to evaluate the performability of degradable computing systems.
Figure 1

Hardware organization of SIFT
Figure 2

The model hierarchy for S and its relation to $y_s$. 

Performance of S

Mission Level (Level 0)

Aircraft Level (Level 1)

Computer Level (Level 2)
Key:  

\[ p \] = processor failure rate
\[ q \] = bus failure rate

Figure 3
Markov transition graph for phases other than takeoff
Table 1
Function table of $\kappa_0$

<table>
<thead>
<tr>
<th>ECONOMICS</th>
<th>OPERATIONS</th>
<th>PROFILE</th>
<th>SAFETY</th>
<th>$\kappa_0(\mu)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$a_0$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$a_1$</td>
</tr>
<tr>
<td>*</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$a_2$</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>1</td>
<td>0</td>
<td>$a_3$</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>1</td>
<td>$a_4$</td>
</tr>
<tr>
<td>Phase</td>
<td>Description</td>
<td>Level 1 Phase</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>--------------</td>
<td>---------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Take-off</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Climb</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Cruise I</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Cruise II</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Cruise III</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Descent</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Approach</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Landing</td>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 2**

Level 2 phases and their relation to level 1.
<table>
<thead>
<tr>
<th>No. of Processors</th>
<th>Phase</th>
<th>2 - 6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>AIDS</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>INERTIAL AID</td>
<td>AIDS</td>
<td>AIDS</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>INERTIAL AID</td>
<td>AIDS</td>
<td>AIR DATA AIDS</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>INERTIAL AID</td>
<td>INERTIAL AID</td>
<td>AIDS</td>
<td></td>
</tr>
<tr>
<td>2'</td>
<td>ENGINE CONTROL INERTIAL AID</td>
<td>ENGINE CONTROL INERTIAL</td>
<td>ENGINE CONTROL INERTIAL AIDS</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>All Tasks</td>
<td>All Tasks</td>
<td>All Tasks</td>
<td>All Tasks</td>
</tr>
</tbody>
</table>

Table 3
Loss of functional tasks
### Table 4
Initial state probabilities

<table>
<thead>
<tr>
<th>State</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i, j)</td>
<td>I₁</td>
</tr>
<tr>
<td>(6,6)</td>
<td>.64</td>
</tr>
<tr>
<td>(6,5)</td>
<td>.128</td>
</tr>
<tr>
<td>(6,4)</td>
<td>.032</td>
</tr>
<tr>
<td>(5,6)</td>
<td>.16</td>
</tr>
<tr>
<td>(5,5)</td>
<td>.032</td>
</tr>
<tr>
<td>(5,4)</td>
<td>.008</td>
</tr>
<tr>
<td>Others</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 5
Phase durations

<table>
<thead>
<tr>
<th>Phase</th>
<th>London-New York</th>
<th>Tel Aviv-New York</th>
</tr>
</thead>
<tbody>
<tr>
<td>Takeoff</td>
<td>1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>Climb</td>
<td>15 minutes</td>
<td>15 minutes</td>
</tr>
<tr>
<td>Cruise I</td>
<td>25 minutes</td>
<td>25 minutes</td>
</tr>
<tr>
<td>Cruise II</td>
<td>5 hours</td>
<td>8 hours 35 minutes</td>
</tr>
<tr>
<td>Cruise III</td>
<td>25 minutes</td>
<td>25 minutes</td>
</tr>
<tr>
<td>Descent</td>
<td>15 minutes</td>
<td>15 minutes</td>
</tr>
<tr>
<td>Approach</td>
<td>3 minutes</td>
<td>3 minutes</td>
</tr>
<tr>
<td>Landing</td>
<td>1 minute</td>
<td>1 minute</td>
</tr>
<tr>
<td>Total Duration</td>
<td>6 hours 25 minutes</td>
<td>10 hours</td>
</tr>
</tbody>
</table>

Table 5
Phase durations
<table>
<thead>
<tr>
<th>System</th>
<th>C3 Init. State</th>
<th>EL Flight</th>
<th>$\delta_0$</th>
<th>$\delta_1$</th>
<th>$\delta_2$</th>
<th>$\delta_3$</th>
<th>$\delta_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>Det(6,6)</td>
<td>Lon-NY</td>
<td>9.96x10^{-1}</td>
<td>1.80x10^{-3}</td>
<td>3.78x10^{-12}</td>
<td>6.02x10^{-6}</td>
<td>1.95x10^{-12}</td>
</tr>
<tr>
<td>$S_2$</td>
<td>Det(6,5)</td>
<td>Lon-NY</td>
<td>9.96x10^{-1}</td>
<td>1.80x10^{-3}</td>
<td>3.79x10^{-12}</td>
<td>6.02x10^{-6}</td>
<td>1.95x10^{-12}</td>
</tr>
<tr>
<td>$S_3$</td>
<td>Det(6,4)</td>
<td>Lon-NY</td>
<td>9.96x10^{-1}</td>
<td>1.80x10^{-3}</td>
<td>3.32x10^{-10}</td>
<td>6.05x10^{-6}</td>
<td>2.97x10^{-12}</td>
</tr>
<tr>
<td>$S_4$</td>
<td>Det(5,6)</td>
<td>Lon-NY</td>
<td>0</td>
<td>9.97x10^{-1}</td>
<td>1.03x10^{-9}</td>
<td>3.17x10^{-3}</td>
<td>1.55x10^{-9}</td>
</tr>
<tr>
<td>$S_5$</td>
<td>Det(5,5)</td>
<td>Lon-NY</td>
<td>0</td>
<td>9.97x10^{-1}</td>
<td>1.03x10^{-9}</td>
<td>3.17x10^{-3}</td>
<td>1.55x10^{-9}</td>
</tr>
<tr>
<td>$S_6$</td>
<td>Det(5,4)</td>
<td>Lon-NY</td>
<td>0</td>
<td>9.97x10^{-1}</td>
<td>1.16x10^{-9}</td>
<td>3.17x10^{-3}</td>
<td>1.55x10^{-9}</td>
</tr>
<tr>
<td>$S_7$</td>
<td>Det(6,6)</td>
<td>TA-NY</td>
<td>9.94x10^{-1}</td>
<td>6.03x10^{-3}</td>
<td>6.07x10^{-12}</td>
<td>1.52x10^{-5}</td>
<td>1.30x10^{-11}</td>
</tr>
<tr>
<td>$S_8$</td>
<td>Det(6,5)</td>
<td>TA-NY</td>
<td>9.94x10^{-1}</td>
<td>6.03x10^{-3}</td>
<td>6.12x10^{-12}</td>
<td>1.52x10^{-5}</td>
<td>1.30x10^{-11}</td>
</tr>
<tr>
<td>$S_9$</td>
<td>Det(6,4)</td>
<td>TA-NY</td>
<td>9.94x10^{-1}</td>
<td>6.03x10^{-3}</td>
<td>2.09x10^{-10}</td>
<td>1.53x10^{-5}</td>
<td>1.71x10^{-11}</td>
</tr>
<tr>
<td>$S_{10}$</td>
<td>Det(5,6)</td>
<td>TA-NY</td>
<td>0</td>
<td>9.95x10^{-1}</td>
<td>1.03x10^{-9}</td>
<td>5.03x10^{-3}</td>
<td>7.15x10^{-9}</td>
</tr>
<tr>
<td>$S_{11}$</td>
<td>Det(5,5)</td>
<td>TA-NY</td>
<td>0</td>
<td>9.95x10^{-1}</td>
<td>1.03x10^{-9}</td>
<td>5.03x10^{-3}</td>
<td>7.15x10^{-9}</td>
</tr>
<tr>
<td>$S_{12}$</td>
<td>Det(5,4)</td>
<td>TA-NY</td>
<td>0</td>
<td>9.95x10^{-1}</td>
<td>1.23x10^{-9}</td>
<td>5.03x10^{-3}</td>
<td>7.15x10^{-9}</td>
</tr>
<tr>
<td>$S_{13}$</td>
<td>Det(6,6)</td>
<td>TA-NY</td>
<td>7.95x10^{-1}</td>
<td>2.04x10^{-1}</td>
<td>2.18x10^{-10}</td>
<td>1.02x10^{-3}</td>
<td>1.44x10^{-9}</td>
</tr>
<tr>
<td>$S_{14}$</td>
<td>Det(6,5)</td>
<td>TA-NY</td>
<td>8.95x10^{-1}</td>
<td>1.05x10^{-1}</td>
<td>1.10x10^{-10}</td>
<td>5.17x10^{-4}</td>
<td>7.26x10^{-10}</td>
</tr>
</tbody>
</table>

Table 6
Performability results
REFERENCES


