On Evaluating the Performability of Degradable Computing Systems

JOHN F. MEYER, SENIOR MEMBER, IEEE

Abstract—**If the performance of a computing system is "degradable," performance and reliability issues must be dealt with simultaneously in the process of evaluating system effectiveness. For this purpose, a unified measure, called "performability," is introduced and the foundations of performability modeling and evaluation are established. A critical step in the modeling process is the introduction of a "capability function" which relates low-level system behavior to user-oriented performance levels. A hierarchical modeling scheme is used to formulate the capability function and capability is used, in turn, to evaluate performability. These techniques are then illustrated for a specific application: the performability evaluation of an aircraft computer in the environment of an air transport mission.**

Index Terms—**Degradable computing systems, fault-tolerant computing, hierarchical modeling, performability evaluation, performance evaluation, reliability evaluation.**

I. Introduction

During the past decade, performance evaluation and reliability evaluation have emerged as important technical disciplines within computer science and engineering. Evaluations of computer performance and computer reliability are each concerned, in part, with the important question of computer system "effectiveness," that is, the extent to which the user can expect to benefit from the tasks accomplished by a computer in its use environment. With regard to effectiveness issues, modeling of computer performance (see [1]–[3], for example) has stressed the need to represent the probabilistic nature of user demands (workload) and internal state behavior, under the assumption that the computer's structure is fixed (that is, there are no permanent changes in structure due to faults). On the other hand, modeling of computer reliability (beginning with the pioneering work of Bouricius *et al.* [4]) has stressed representation of the probabilistic nature of structural changes caused by transient and permanent faults of the computer.

In the face of these traditional modeling distinctions, we consider an important class of computing systems wherein system performance is "degradable," that is, depending on the history of the computer's structure, internal state, 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). In this case we find that performance evaluations (of the fault-free system) will generally not suffice since structural changes, due to faults, may be the cause of degraded performance. By the same token, traditional views of reliability (probability of success, mean time to failure, etc.) no longer suffice since "success" can take on various meanings and, in particular, it need not be identified with "absence of system failure."

Modeling needs for (gracefully) degradable systems were first investigated by Borgerson and Frietas [5] in connection with their analysis of the PRIME system [6]. Although they recognized the need to formulate the probability of each possible level of performance, that is, the probability of k "crashes" during T for $k = 0,1,2,\cdots$, their evaluation effort dealt mainly with the question of reliability (the probability of no crashes during T). Other studies employing Markov models have likewise emphasized the evaluation of reliability oriented measures (see [7]–[9], for example).

Some recent investigations, on the other hand, have dealt with measures aimed at quantifying performance as well as reliability. In particular, Beaudry [10] has introduced a number of computation related measures for degradable computing systems and has shown how to formulate these measures in terms of a transformed Markov model. In examining reconfiguration strategies for degradable systems, Troy [11] has distinguished levels of performance according to "workpower" and has formulated system effectiveness (referred to as "operational efficiency") as expected workpower. In another recent study, Losq [12] has investigated degradable systems in terms of degradable resources, where each resource is modeled by an irreducible, recurrent, finite-state Markov process.

In the discussion that follows, we describe a general modeling framework that permits the definition, formulation, and evaluation of a unified performance-reliability measure referred to as "performability." It is shown that performability relates directly to system effectiveness and is a proper generalization of both performance and reliability. A critical step in performability modeling is the introduction of the concept of a "capability function" which relates low-level system behavior to user-oriented levels of performance. A hierarchical modeling scheme is used to formulate the capability function, and capability is used, in turn, to evaluate performability.

II. System Models

A computing system, as it operates in its use environment, may be viewed at several levels. At a low level, there is a de-

Manuscript received June 19, 1978; revised February 22, 1980. This research was supported by the NASA Langley Research Center under Grant NSG 1306.

The author is with the Department of Electrical and Computer Engineering and the Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, MI 48109.

tailed view of how various components of the computer's hardware and software 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 man-made components (user input, peripheral subsystems, etc.) and natural components (radiation, weather, etc.) which can influence the computer's effectiveness. The computer, together with its environment, will be referred to as the "total system." A second view of the total system is the user's view of how the system behaves during utilization, that is, what the system accomplishes for the user during the utilization period. A third, even less detailed view, 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.

To formalize these views, we postulate the existence of a *probability space* $(\Omega,\mathcal{E},P)$ that underlies the total system, where Ω is the *sample space*, $\mathcal{E}$ is a set of *events* (measurable subsets of Ω), and P: $\mathcal{E} \rightarrow [0,1]$ is the *probability measure* (see [13], for example). This probability space represents all that needs to be known about the total system in order to describe the probabilistic nature of its behavior at the various levels described above. It thus provides a hypothetical basis for defining higher level (i.e., less detailed) models. In general, however, it will neither be possible nor desirable to completely specify Ω, $\mathcal{E}$, and P.

In the discussion that follows, let S denote the *total system*, where S is comprised of a *computing system* C and its *environment* E. At the most detailed level, the behavior of S is formally viewed as a stochastic process [14], [15]

$$X_S = \{X_t | t \in T\} \tag{1}$$

where T is a set of real numbers (observation times) called the *utilization period* and, for all $t \in T$, X_t is a random variable

$$X_t{:}\Omega \rightarrow Q$$

defined on the underlying description space and taking values in the *state space* Q of the total system. Depending on the application, the utilization period T may be discrete (countable) or continuous and, in cases where one is interested in the long-run behavior, it may be unbounded [e.g., $T = \boldsymbol{R}_+ = [0,\infty)$]. The state space Q embodies the *state sets* of both the computer and its environment, i.e.,

$$Q = Q_C \times Q_E$$

where Q_C and Q_E can, in turn, be decomposed to represent the local state sets of computer and environmental subsystems. For our purposes, it suffices to assume that Q is discrete and, hence, for all $t \in T$ and $q \in Q$, "$X_t = q$" has a probability (i.e., $\{\omega | X_t(\omega) = q\} \in \mathcal{E}$). The stochastic process X_S is referred to as the *base model* of S. An instance of the base model's behavior for a fixed $\omega \in \Omega$ is a *state trajectory* $u_\omega{:}T \rightarrow Q$ where

$$u_\omega(t) = X_t(\omega), \qquad \forall t \in T. \tag{2}$$

(The term "state trajectory" derives from modern usage in the theory of modeling [16]; synonyms in the more specific context of stochastic processes are "sample function," "sample path," and "realization" [14], [15].) Thus, corresponding to an underlying outcome $\omega \in \Omega$, u_ω describes how the state of the total system changes as a function of time throughout the utilization period T. Accordingly, the "description space" for the base model is the set

$$U = \{u_\omega | \omega \in \Omega\} \tag{3}$$

which is referred to as the (state) *trajectory space* of S.

As generally defined, the concept of a base model thus includes the type of queueing models used in computer performance evaluation [2], [3] and the kind of Markov or semi-Markov models employed in reliability evaluation [7]-[9]. The intent of the definition, however, is the inclusion of less restricted base models which can represent simultaneous variations in the computer's structure and internal state (via the state set Q_C) and environment (via the state set Q_E). In other words, the emphasis here is on the modeling of degradable computing systems where changes in structure, internal state, and environment can all have an influence on the system's ability to perform. Accordingly, these base models may be regarded as generalized performance models, where structure is allowed to vary, or equivalently as generalized reliability models where variations in internal state and/or the computational environment are taken into account.

In formal terms, the user-oriented view of system behavior is likewise defined in terms of the underlying probability space $(\Omega,\mathcal{E},P)$. Here 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 (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*. A may be finite, countably infinite, or uncountable (in the last case, A is assumed to be an interval of real numbers). Thus, for example, the accomplishment set associated with a nondegradable system is $A = \{a_0, a_1\}$ where a_0 = "system success" and a_1 = "system failure." In their modeling of the PRIME system, Borgerson and Freitas [5] viewed accomplishment as the set $A = \{a_0,a_1,a_2, \cdots\}$ where a_k = "k crashes during the utilization period T." If the user is primarily concerned with system "throughput," a continuous accomplishment set might be appropriate, i.e., $A = \boldsymbol{R}_+ = [0,\infty)$, where a number $a \in A$ is the "throughput averaged over the utilization period T."

In terms of the accomplishment set, *system performance* is formally viewed as a random variable

$$Y_S{:}\ \Omega \rightarrow A \tag{4}$$

where $Y_S(\omega)$ is the accomplishment level corresponding to outcome ω in the underlying description space. Similarly, assuming that the economic gain (or loss) derived from using the system is represented by a real number r (interpreted, say, as r dollars), *system worth* is a random variable defined as

$$W_S{:}\ \Omega \rightarrow \boldsymbol{R} \text{ (the real numbers)} \tag{5}$$

where $W_S(\omega)$ is the worth associated with outcome ω.

The terminology and notation defined previously is summarized in Table I. Note that, at this point in the development, there are no implied relationships between these three views; their only common bond so far is that they are representations

TABLE I
TERMINOLOGY AND NOTATION FOR SYSTEM MODELS

Model	Description Space
Base model X_S	Trajectory space U
System performance Y_S	Accomplishment set A
System worth W_S	The real numbers $\mathbb{R}$

of the same system or, more formally, that they are defined on the same underlying probability space. To be useful, however, the base model X_S should support the performance variable Y_S in an appropriate manner (indeed, the term "base" is suggestive of this need) and, in turn, Y_S should support the worth variable W_S. The precise nature of these connections, as they relate to the system's effectiveness, is developed in the section that follows.

III. EFFECTIVENESS, PERFORMABILITY, AND CAPABILITY

When applied to computing systems, "system effectiveness" (see [1], for example) is a measure of the extent to which the user can expect to benefit from the tasks accomplished by a computer in its use environment. More precisely, if benefit is identified with the worth W_S of the system then *system effectiveness* is expected worth, i.e., the expectation (expected value) of the random variable W_S; in short

$$\text{eff}(S) = E[W_S]. \tag{6}$$

(An implicit assumption here is that W_S is defined such that $E[W_S]$ exists; see [13], for example.) Because a direct evaluation of eff(S), using the definition of W_S, is generally not feasible (cf., our earlier remarks concerning the hypothetical nature of the underlying probability space), we wish to establish connections among the base model X_S, system performance Y_S, and system worth W_S which can be used in the process of evaluating eff(S).

To express system effectiveness in terms of system performance, the user's view of system worth must be compatible with system performance to the extent that W_S can be formulated as a function of Y_S. More precisely, we assume there exists a *worth function* $w: A \rightarrow \mathbf{R}$ such that, for all $\omega \in \Omega$,

$$W_S(\omega) = w(Y_S(\omega)). \tag{7}$$

If $a \in A$, $w(a)$ is interpreted as the "worth of performance level a." As for the performance variable Y_S, a natural measure that quantifies both system performance and reliability (i.e., the ability to perform) is the probability measure induced by Y_S. We refer to this unified performance-reliability measure as the "performability of S" which, in terms of our modeling framework, can be generally defined as follows.

Definition 1: If S is a total system with performance Y_S taking values in accomplishment set A, then the *performability of* S is the function p_S where, for each measurable set B of accomplishment levels ($B \subseteq A$),

$$p_S(B) = P(\{\omega | Y_S(\omega) \in B\}).$$

Since P is the probability measure of the underlying probability space, the interpretation of performability is straightforward, that is, for a designated set B of accomplishment levels, $p_S(B)$ is the probability that S performs at a level in B. The requirement that B be "measurable" says simply that the corresponding event $\{\omega | Y_S(\omega) \in B\}$ must lie in the underlying event space, insuring that the right-hand probability is defined.

If the performance variable Y_S is continuous then A must be continuous and, hence (by an earlier assumption), A is some interval of real numbers, including the possibility that $A = \mathbf{R} = (-\infty,\infty)$. In this case (or if Y_S happens to be discrete and yet real-valued), we know from probability theory that the induced measure p_S is uniquely determined by the *probability distribution function* of Y_S (see [13], for example), i.e., by the function F_{Y_S} (which we write simply as F_S) where, for all $b \in A$,

$$F_S(b) = P(\{\omega | Y_S(\omega) \le b\}). \tag{8}$$

Moreover, p_S can then be expressed as the Lebesgue–Stieltjes measure induced by F_S (cf., [13, sec. 4.5]), that is, for any (measurable) set B of accomplishment levels, the performability value of B is given by

$$p_S(B) = \int_B dF_S(b). \tag{9}$$

In particular, if B is a single interval $B = (b_0,b_1]$ where $b_0 < b_1$, then

$$p_S(B) = F_S(b_1) - F_S(b_0).$$

This special case has practical significance since it quantifies the ability of S to perform within the specified limits b_0 and b_1.

If, on the other hand, Y_S is a discrete random variable then each singleton set $B = \{a\}(a \in A)$ is measurable and p_S is uniquely determined by the *probability distribution* of Y_S, i.e., by the set of values

$$\{p_S(a) | a \in A\} \tag{10}$$

where $p_S(a)$ denotes $p_S(\{a\})$. Given this distribution, if B is a set of accomplishment levels then $p_S(B)$ can be written as the sum

$$p_S(B) = \sum_{b \in B} p_S(b). \tag{11}$$

Hence, the probability distribution of Y_S or, equivalently, the restriction of p_S to single accomplishment levels, suffices to determine the performability. For this reason, when Y_S is discrete the performability of S can be alternatively defined as follows.

Definition 1a: If S is a total system with performance Y_S taking values in accomplishment set A and, moreover, Y_S is a discrete random variable, then the *performability of* S is the function p_S where, for each accomplishment level $a(a \in A)$,

$$p_S(a) = P(\{\omega | Y_S(\omega) = a\}).$$

Note that Definition 1a is essentially the restriction of Definition 1 to single accomplishment levels (which have probabilities defined when Y_S is discrete). Conversely, given Definition 1a, its extension to Definition 1 can be obtained (in principle) by application of (11) to each subset B of A.

To justify the notion of performability in the context of system effectiveness, if we assume the existence of a worth function w [see (7)], then the real-valued random variable W_S is a function w of the random variable Y_S. Moreover, we know that w is a "measurable" function (e.g., see [13, sec. 3.8]) since, prior to (7), we assumed that W_S was a random variable. (Indeed, condition (7) is actually stronger than needed; its advantages, however, are its simplicity and the fact that it serves the purpose of the present discussion.) Hence, we are able to appeal to the well-developed theory of functions of a random variable and, particularly, expectations where, again, it is convenient to distinguish two cases. If the performance variable Y_S is continuous (and thus real-valued) with probability distribution function F_S (8) then

$$E[w(Y_S)] = \int_A w(a)\, dF_S(a) \qquad (12)$$

where the integral is a Lebesgue–Stieltjes integral. In case Y_S is discrete, then (12) still applies provided the levels in A are represented by real numbers. However, independent of whether Y_S is real-valued, a simpler and more familiar formulation holds in this case where, if $p_S(a)$ is as defined in (10), then

$$E[w(Y_S)] = \sum_{a \in A} w(a) p_S(a). \qquad (13)$$

By the definition of system effectiveness (6) and the fact that $W_S = w(Y_S)$ (7), we have

$$\text{eff}(S) = E[W_S] = E[w(Y_S)]$$

and, accordingly, (12) and (13) are formulations of the effectiveness of S. Moreover, we see that each formula involves the worth function w and the performability p_S [although p_S does not occur explicitly in (12), recall that F_S characterizes p_S (9)]. In other words, relative to the system-user interface delineated by the accomplishment set A, effectiveness evaluation may be decomposed into worth evaluation (on the user side of the interface) and performability evaluation (on the system side). Consequently, looking in toward the system, performability emerges as a key measure with regard to evaluations of system effectiveness.

To further justify the concept of performability, we note that traditional evaluations of computer performance and computer reliability are concerned with special types of performability. Performance evaluation is concerned with evaluating p_S under the assumption that the computer part of S is fixed (i.e., its structure does not change as the consequence of internal faults). Reliability evaluation is concerned with evaluating $p_S(B)$ where B is a designated subset of accomplishment levels associated with system "success." If A is finite, a performability evaluation can alternatively be regarded as $|A|$ reliability evaluations, one for each singleton "success set" $B = \{a\}$, and the evaluation may actually be carried out in this manner. As this process is generally more complex than a typical reliability evaluation procedure (in particular, it involves distinguishing all the performance levels as well as determining their probabilities), we reserve the term "reliability evaluation" to mean the evaluation of "probability of success" for some specified success criterion B. Moreover, even when $|A| = 2$, we find (as discussed later in this section) that performability models represent a proper extension of models typically employed in reliability evaluation.

As a final remark regarding justification, we have found that when system performance is not degradable (as in the case, for example, with fault-tolerant architectures which employ standby sparing [4], N modular redundancy [17], or combinations thereof), it is possible to treat performance and reliability as separate issues in the process of evaluating system effectiveness. On the other hand, if performance is degradable, it can be shown (see [18]) that the more general concept of performability must typically be invoked (as in (12) and (13), for example) when evaluating system effectiveness.

With performability established as the object of the evaluation process, we are now in a position to specify how the base model process X_S (1) must relate to the performance variable Y_S (4) if it is to support an evaluation of the performability p_S. To precisely state this relationship, we suppose that X_S is specified by its finite-dimensional probability distributions (or by information that determines these distributions) and we let Pr denote the probability measure (defined on a σ algebra of subsets of U) which is uniquely determined by these finite-dimensional distributions (see [14], for example). If Pr is defined for a trajectory set $V (V \subseteq U)$ then, relative to the underlying measure P,

$$\Pr(V) = P(\{\omega | u_\omega \in V\}), \qquad (14)$$

i.e., $\Pr(V)$ is the probability that an observed state trajectory u_ω [see (2)] lies in the set V. In practice, however, $\Pr(V)$ will be calculated directly from the finite-dimensional distributions that determine Pr. The measure Pr thus serves to formally describe the probabilistic nature of the base model X_S.

For X_S to support Y_S, we now impose the following restrictions. We assume first that the base model is refined enough to distinguish the levels of accomplishment perceived by the user, that is, for all $\omega, \omega' \in \Omega$,

$$Y_S(\omega) \neq Y_S(\omega') \text{ implies } u_\omega \neq u_{\omega'} \qquad (15)$$

where u_ω and $u_{\omega'}$ are the state trajectories associated with outcomes ω and ω'. This implies that each trajectory $u \in U$ is related to a unique accomplishment level $a \in A$. In addition, we assume that the probabilistic nature of Y_S is determinable from that of X_S. More precisely, if B is a measurable set of accomplishment levels, i.e., the set $\{\omega | Y_S(\omega) \in B\}$ is in the domain of the underlying measure P, then we require that the corresponding trajectory set

$$U_B = \{u_\omega | Y_S(\omega) \in B\}$$

lie in the domain of the base model measure Pr; in short

$$\text{If } B \text{ is measurable then Pr is defined for } U_B. \qquad (16)$$

Given that conditions (15) and (16) are satisfied, we can establish a link between X_S and Y_S which, in the context of effectiveness modeling [18], is generally referred to as the "capability" of S. Adopting this terminology, we have

Definition 2: If S is a system with trajectory space U and accomplishment set A then the *capability function of* S is the function $\gamma_S: U \rightarrow A$ where $\gamma_S(u)$ is the level of accomplishment resulting from state trajectory u, that is,

$$\gamma_S(u) = a \qquad \text{if for some } \omega \in \Omega,\ u_\omega = u \text{ and } Y_S(\omega) = a.$$

Condition (15) insures that the capability function γ_S is well-defined (i.e., it deserves the name "function"), for if $u_\omega = u_{\omega'}$ then $\gamma_S(u_\omega) = \gamma_S(u_{\omega'})$. Condition (16) guarantees that the inverse γ_S^{-1} of the capability function (γ_S^{-1} is a relation between A and U but generally not a function) will carry sets that are measurable with respect to Y_S into sets that are measurable with respect to X_S. To substantiate this fact, suppose that B is a measurable set of accomplishment levels. Then the inverse image of B is the trajectory set

$$\gamma_S^{-1}(B) = \{u \mid \gamma_S(u) \in B\}$$

or, equivalently, by the definition of γ_S (Definition 2),

$$\gamma_S^{-1}(B) = \{u_\omega \mid Y_S(\omega) \in B\} = U_B.$$

But condition (16) insures that Pr is defined for U_B and, hence, $\gamma_S^{-1}(B)$ is measurable where

$$\Pr(\gamma_S^{-1}(B)) = \Pr(U_B). \tag{17}$$

In effect, therefore, conditions (15) and (16) say that γ_S can be viewed as a random variable defined on the probability space (with measure Pr) induced by the base model X_S. Of more practical significance, however, is the fact that, under these conditions, X_S and γ_S suffice to determine the performability of S. To argue the latter, if B is measurable then, by (14),

$$\Pr(U_B) = P(\{\omega \mid u_\omega \in U_B\}) = P(\{\omega \mid Y_S(\omega) \in B\})$$

which, by Definition 1, is just the performability of S for accomplishment levels B, i.e.,

$$\Pr(U_B) = p_S(B). \tag{18}$$

Combining (17) and (18), we conclude that

$$p_S(B) = \Pr(\gamma^{-1}(B)), \tag{19}$$

substantiating the fact that X_S and γ_S (which determine Pr and γ_S^{-1}, respectively) suffice to support an evaluation of the performability p_S.

In view of what has just been observed, if X_S and Y_S admit to the definition of a capability function γ_S (in which case we presume that conditions (15) and (16) are satisfied), the pair (X_S, γ_S) is said to constitute a *performability model of* S. If B is a (measurable) set of accomplishment levels, the inverse image $\gamma_S^{-1}(B) = U_B$ is referred to as the *trajectory set of* B, where its determination requires an analysis of how levels in B relate back down via γ_S^{-1} to trajectories of the base model. $p_S(B)$ is then determined by a probability analysis of $\gamma_S^{-1}(B)$. In case Y_S is discrete (Definition 1a), it suffices to consider inverse images of the form $\gamma_S^{-1}(a)$ where $a \in A$. Methods of implementing this process in the discrete case are discussed in Section IV.

The role of a capability function in performability evaluation is similar to that of a "structure function" [19] in reliability evaluation. However, even when performability is restricted to reliability, the concept of a capability function is more general. The special class which corresponds to the use of structure functions in "phased mission" analysis (see [20], for example) may be characterized as follows. Let S be a system where Q is the state space of the base model and $A = \{0,1\}$ is the accomplishment set (here, 1 denotes "success" and 0 denotes "failure"). Then a capability function γ_S is *structure-based* if there exists a decomposition of T into k consecutive time periods $T_1, T_2, \cdots, T_k$ and there exist functions $\varphi_1, \varphi_2, \cdots, \varphi_k$ with $\varphi_i: Q \rightarrow \{0,1\}$ such that, for all $u \in U$,

$$\gamma_S(u) = 1 \quad \text{iff } \varphi_i(u(t)) = 1, \tag{20}$$

for all $i \in \{1, 2, \cdots, k\}$ and for all $t \in T_i$. In the context of "phased mission" analysis, T_i is referred to as the ith *phase* (of the mission) and φ_i is the *structure function* of the ith phase. For each function φ_i, the inverse image $\varphi_i^{-1}(1)$ can be interpreted as the set of "success states" of the ith phase and, accordingly, (20) says that S performs successfully ($\gamma_S(u) = 1$) if and only if $u(t)$ is a success state throughout each phase. Thus, the advantage of a structure-based formulation is that each phase may be treated independently when determining the set $\gamma_S^{-1}(1)$ of all successful state trajectories.

If system success is viewed in structural terms, as is the case in most reliability studies, a structure-based capability function will usually suffice. On the other hand, when success relates to system performance we find that capability may no longer be expressible in terms of locally defined success criteria as specified by the structure functions φ_i. The following example serves to demonstrate this fact.

Let $S = (C,E)$ where C represents a distributed computer comprised of n subsystems, and E represents the computer's workload. Suppose further that system "throughput" (i.e., the user-visible work rate of C in E) varies as a function of the number of faulty subsystems. For our purposes here, it suffices to assume that the workload E is constant and, hence, the operational states of S can be represented by the state space $Q = \{q_0, q_1, \cdots, q_n\}$ where state q_i corresponds to "i faulty subsystems." The variation in throughput is described by a function $\tau: Q \rightarrow \boldsymbol{R}_+$ where $\tau(i)$ = the throughput of S in state q_i. Assuming S is used continuously throughout a utilization period $T = [0,h]$ of duration $h > 0$, the base model of S is a stochastic process $X_S = \{X_t \mid t \in [0,h]\}$ where each X_t is a random variable taking values in Q. (The probabilistic nature of X_S is not an issue here.) As for performance, suppose that the user is interested in the average throughput of the system, where the average is taken over the utilization period T. Suppose further that system "success" is identified with a minimum average throughput $\bar{\tau}$. Then the capability function of S is the function $\gamma_S: U \rightarrow \{0,1\}$ where

$$\gamma_S(u) = \begin{cases} 1 & \text{if } \dfrac{1}{h}\displaystyle\int_0^h \tau(u(t))dt \geq \bar{\tau} \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

Due to the inherent memory of the integration operation, we find that γ_S does not admit to a structure-based formulation. To verify this fact with a simple 2-state example, suppose $Q = \{q_0, q_1\}$, $\tau(q_0) > \tau(q_1)$, and $\bar{\tau} = (\tau(q_0) - \tau(q_1))/2$. Then, according to (21), $\gamma_S(u) = 1$ if the total time for which $u(t) = q_0$ is at least $h/2$. In particular, this says that more than one trajectory results in success, i.e., $|\gamma_S^{-1}(1)| > 1$. To prove that γ_S is not structure-based let us suppose to the contrary, that is, there exist phases $T_1, T_2, \cdots, T_k$ and structure functions $\varphi_1, \varphi_2, \cdots, \varphi_k$ such that (20) is satisfied. If we let R_i denote the success states of phase i, i.e., $R_i = \varphi_i^{-1}(1)$, then $R_i \neq \phi$, for all i, or otherwise no trajectory results in success. It must also be the case that $R_i \neq \{q_0, q_1\}$ for all i, for if $R_i = \{q_0, q_1\}$ (all states are success states during phase i) then the condition $\varphi_i(u(t)) = 1$, $\forall\ t \in T_i$ [see (20)] is always satisfied, that is, phase i can be ignored when determining whether u spends at least half its time in state 0. This is clearly impossible if the duration of T_i is at least $h/2$. If the duration of T_i is less than $h/2$, trajectories u and v can be found such that $u(t) = v(t)$, $\forall\ t \in (T - T_i)$, and yet $\gamma_S(u) \neq \gamma_S(v)$ contradicting the ability to ignore phase i. The only remaining alternative is that $|R_i| = 1$, for all i, that is, each phase has exactly one success state which, in turn, implies that there is exactly one success trajectory u. This contradicts our initial observation that $|\gamma_S^{-1}(1)| > 1$ and proves that γ_S is not structure-based.

We can conclude, therefore, that even in the case of two accomplishment levels, the concept of a capability function (Definition 2) represents a proper extension of relations between state behavior and system performance that are typically assumed in the theory of reliability. Moreover, we have found that this extension permits the phases of a utilization period to be "functionally dependent" in a precisely defined sense, whereas the phases associated with a structure-based capability function must be functionally independent. The reader is referred to [21] for a more complete discussion of functional dependence and its implications.

IV. Performability Evaluation

As established in the previous section [see (19)], if (X_S, γ_S) is a performability model then the performability of S for a set of accomplishment levels B may be expressed as $p_S(B) = \Pr(\gamma_S^{-1}(B))$. Accordingly, one method of evaluating a particular $p_S(B)$ is to 1) determine $\gamma_S^{-1}(B)$ and then 2) evaluate $\Pr(\gamma_S^{-1}(B))$. Since the "distance" between the base model X_S and the accomplishment set A may be considerable, step 1) can be facilitated by introducing additional models between X_S and A.

In general, each intermediate model is defined in a manner similar to that of the base model. More precisely, if there are $m + 1$ levels in the hierarchy, the *level-i model* ($i = 0, 1, \cdots, m$, where level-0 is the least detailed model at the "top" of the hierarchy) is a stochastic process

$$X^i = \{X_t^i \mid t \in T^i\}, \qquad T^i \subseteq T$$

where, for a fixed $t \in T^i$, X_t^i is a random variable taking values in a set Q^i, the *state space* of X^i. The state space Q^i is generally composed of two components, i.e.,

$$Q^i = Q_c^i \times Q_b^i$$

where Q_c^i is the *composite state set* and Q_b^i is the *basic state set* (at level-i). States in the composite part Q_c^i represent a less detailed view of the operational status of the system than do states in Q^{i+1}, such that the state behavior at level-$(i + 1)$ uniquely determines the composite state behavior at level-i (this will be made more precise in a moment). States in Q_b^i, on the other hand, represent basic information not conveyed by states in Q^{i+1}, i.e., Q_b^i is a coordinate set of the base model state space Q. In case there is no composite (alternatively, basic) part at level-i, $Q_c^i(Q_b^i)$ is simply deleted, that is, $Q^i = Q_b^i$ ($Q^i = Q_c^i$). In particular, the above definition precludes a composite state set at level-m (the "bottom" level of the hierarchy) and, hence, $Q^m = Q_b^m$.

In specifying the model hierarchy, it is convenient to view X^i as a pair of processes which determine the projections on Q_c^i and Q_b^i, respectively. (If one of Q_c^i or Q_b^i does not exist, this pair reduces to a single process.) More precisely, given Q_c^i, the *composite process* (at level-i) is the stochastic process

$$X_c^i = \{X_{c,t}^i \mid t \in T_c^i\}, \qquad T_c^i \subseteq T^i$$

where the random variables $X_{c,t}^i$ take values in Q_c^i. For a fixed outcome ω in the underlying sample space Ω, a *composite state trajectory* is a function $u_{c,\omega}: T_c^i \rightarrow Q_c^i$ where $u_{c,\omega}(t) = X_{c,t}^i(\omega)$; the *composite trajectory space* is the set $U_c^i = \{u_{c,\omega} \mid \omega \in \Omega\}$. Similar definitions, terminology, and notation apply to the *basic process* X_b^i. To permit extension of either X_c^i or X_b^i to larger time bases, a fictitious state ¢ is adjoined to each of Q_c^i and Q_b^i so that if $t \notin T_c^i$ (similar remarks apply to the basic part) then $X_{c,t}^i$ is defined to be a degenerate random variable that always assumes the value ¢, i.e.,

$$X_{c,t}^i(\omega) = ¢, \qquad \text{for all } \omega \in \Omega. \tag{22}$$

If X_c^i and X_b^i are so extended to T^i, and we take X^i to be the process whose projections on Q_c^i and Q_b^i are X_c^i and X_b^i, respectively, then X^i is uniquely determined by X_c^i and X_b^i. (Note that the processes X_c^i and X_b^i may be statistically dependent.)

By the previous observation, we can alternatively regard the level-i model as the pair of processes

$$X^i = (X_c^i, X_b^i)$$

which is a convenient view for the purpose of specifying interlevel relationships. With this identification, a state trajectory of X^i is viewed as a pair of trajectories, i.e., the *trajectory space* U^i (at level-i) is taken to be the set $U_c^i \otimes U_b^i$ where

$$U_c^i \otimes U_b^i = \{(u_{c,\omega}, u_{b,\omega}) \mid \omega \in \Omega\}.$$

In case there is no composite (alternatively, basic) state set at level-i, the above representations of X^i and U^i are understood to be their appropriate single component versions.

The required relationship of these models to the base model, the accomplishment set, and the capability function is prescribed by the following definition.

Definition 3: If S is a total system with base model X_S and capability function γ_S, the collection $\{X^0, X^1, \cdots, X^m\}$ of level-0 to level-m models is a *model hierarchy for S* if the following conditions are satisfied.

a) $X^m = X_b^m$, that is, the bottom model is comprised only of a basic process.

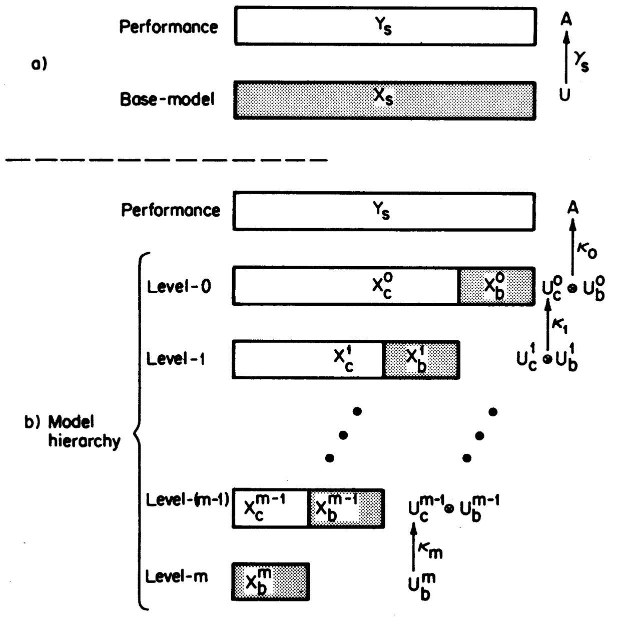


Fig. 1. Total system S with (a) base model X_S and (b) model hierarchy for S.

b) If each model X^i is extended to the utilization period T, the base model X_S is the stochastic process $X_S = \{X_t | t \in T\}$ where $X_t = (X_{b,t}^m, X_{b,t}^{m-1}, \cdots, X_{b,t}^0)$. Accordingly, the state space of X_S is $Q = Q_b^m \times Q_b^{m-1} \times \cdots \times Q_b^0$ and the trajectory space U is represented by the set $U_b^m \otimes U_b^{m-1} \otimes \cdots \otimes U_b^0$.

c) For each level i, there exists an *interlevel translation* κ_i where

$$\kappa_0: U_c^0 \otimes U_b^0 \rightarrow A$$

$$\kappa_i: U_c^i \otimes U_b^i \rightarrow U_c^{i-1} \qquad (1 < i < m)$$

$$\kappa_m: U_b^m \rightarrow U_c^{m-1}$$

such that the capability function γ_S can be decomposed as follows. If $u \in U$ where $u = (u_m, u_{m-1}, \cdots, u_0)$ with $u_i \in U_b^i$, then

$$\gamma_S(u) = \kappa_0(\kappa_1(\cdots \kappa_{m-1}(\kappa_m(u_m), u_{m-1}), \cdots), u_0). \quad (23)$$

The terminology and notation of Definition 3 is summarized in Fig. 1 where Fig. 1(a) is the original model and Fig. 1(b) is the hierarchical model.

A model hierarchy thus provides a step-by-step formulation of the capability function in terms of interlevel translations of state trajectories, beginning with a translation of the bottom model. It also permits the expression of capability relative to higher level (less detailed) views of total system behavior. More precisely, let $\overline{U}^i$ denote the Level-i trajectory space, along with all the basic trajectory spaces of higher level models, i.e.,

$$\overline{U}^i = U^i \otimes U_b^{i-1} \otimes \cdots \otimes U_b^0.$$

(Note that, at the extremes, $\overline{U}^0 = U^0$ and $\overline{U}^m = U$.) Then the *level-i based capability function* is the function

$$\gamma_i: \overline{U}^i \rightarrow A$$

defined inductively as follows. If $i = 0$ and $u \in \overline{U}^0$, then

$$\gamma_0(u) = \kappa_0(u). \quad (24)$$

If $i > 0$ and $(u,u') \in \overline{U}^i$ where $u \in U^i$ and $u' \in U_b^{i-1} \otimes \cdots \otimes U_b^0$, then

$$\gamma_i(u,u') = \gamma_{i-1}(\kappa_i(u),u'). \quad (25)$$

It is easily shown that γ_i has its intended interpretation, i.e., if u and u' correspond to a base model trajectory v then $\gamma_i(u,u') = \gamma_S(v)$. In particular, if $i = m$ then $\gamma_m = \gamma_S$.

The practical significance of the model hierarchy, however, is the ability to formulate the inverse of γ_S via the inverses of the γ_i, thereby providing a step-by-step, top-down method of elaborating a set of accomplishment levels B. Beginning with level-0-based capability, by (24) we have

$$\gamma_0^{-1}(B) = \kappa_0^{-1}(B). \quad (26)$$

Assuming that $\gamma_{i-1}^{-1}(B)$ has been determined, by (25) it follows that

$$\gamma_i^{-1}(B) = \bigcup_{(u,u') \in \gamma_{i-1}^{-1}(B)} (\kappa_i^{-1}(u),u'), \quad (27)$$

where $(\kappa_i^{-1}(u),u') = \{(v,u') | \kappa_i(v) = u\}$. This process is iterated until $i = m$, yielding $\gamma_m^{-1}(B) = \gamma_S^{-1}(B)$. Actual implementations of this procedure can exploit simplified characterizations and decompositions of trajectory sets so as to avoid the manipulation of individual trajectories (see [18] and [22]). Thus a model hierarchy for S is useful not only in the process of model construction but also in the process of model solution.

Although the purpose of this investigation has been to establish the foundations of performability modeling and evaluation, it is helpful to illustrate these ideas in the context of a specific application. The example we consider is an aircraft computing system of the type being developed for next-generation commercial aircraft [23], [24]. Such systems have provided an impetus for the work described herein and are representative of degradable computing systems that operate in a real-time control environment. Although space limitations necessitate a "scaled-down" example, it should suffice to illustrate the basic concepts and constructs. More extensive applications of this type have been investigated subsequent to the work described in this paper; in particular, see [25]-[27] which describe a performability evaluation of the SIFT computer [24].

Beginning at the highest level of description, the total system $S = (C,E)$ is a fault-tolerant computer C which operates in the environment E of a portal-to-portal flight of a commercial aircraft. The user is interested in fuel efficiency, timeliness, and safety, and accordingly, five levels of accomplishment are distinguished:

a_0:	low-fuel consumption, no diversion to an alternate landing site, and safe
a_1:	high-fuel consumption, no diversion, and safe
a_2:	low-fuel consumption, diversion, and safe
a_3:	high-fuel consumption, diversion, and safe
a_4:	unsafe (crash).

The model hierarchy consists of four levels, beginning with a high mission-oriented model and proceeding through aircraft and computational task levels to the bottom level model of the computer. More precisely, employing the notation developed above and assuming $T = [0,h]$, these models are as follows.

Level-0: The model at this level is a simple, user-oriented model which relates directly to the accomplishment levels distinguished above. At this high level, we assume that the model will be fully elaborated at the next lower level (i.e., no part of the state set is basic) and hence $Q^0 = Q_c^0$. The composite state set is taken to be the set

$$Q_c^0 = \{(q_1,q_2,q_3) \mid q_i \in \{0,1\}\}$$

where

$$q_1 = \begin{cases} 0 & \text{if the flight is fuel efficient,} \\ 1 & \text{otherwise,} \end{cases}$$
$$q_2 = \begin{cases} 0 & \text{if the flight is not diverted,} \\ 1 & \text{otherwise,} \end{cases}$$
$$q_3 = \begin{cases} 0 & \text{if the flight is safe,} \\ 1 & \text{otherwise.} \end{cases}$$

Thus, for example, the state

$$q = (0,1,0)$$

says that the flight is fuel efficient and safe but has to be diverted to an alternate landing site. By the interpretation of these states, we are modeling the status of the system on the completion of a flight and thus the time base consists of a single observation time at the end of utilization. More precisely (and to illustrate the notation developed earlier),

$$T_c^0 = \{h\}$$

and, accordingly, the composite process is a single random variable

$$X_c^0 = \{X_{c,h}^0\}$$

with trajectory space

$$U_c^0 = Q_c^0.$$

Since there is no basic process at this level, $X^0 = X_c^0$, thereby completing the description of the level-0 model. The interlevel translation κ_0: $U_c^0 \rightarrow A$ (see Definition 3c); note that $U_c^0 = Q_c^0$ represents U^0 due to the lack of U_b^0) follows immediately from the preceding definitions of Q_c^0 and A, and is given by Table II.

Level-1: At this level of the hierarchy, the model describes the extent to which various aircraft related tasks can be accomplished (the composite part) and the weather condition at the destination airport (the basic part). Although computing systems for advanced commercial aircraft will be called on to realize a variety of control functions, it suffices (for the purpose of illustration) to consider two functional tasks: fuel control (FC) and automatic landing (AL). The FC task encompasses functions such as engine control and navigation which, in a more refined model, might be treated as individual tasks. The AL task is comprised of functions required to automatically land the aircraft in zero-visibility weather; prior to landing, AL is interpreted as (computer-implemented) checkout of the AL system.

To satisfy the requirements of a model hierarchy, construction of the level-1 model must rely on knowledge of how these tasks and the condition of the weather relate to states of the level-0 model. In this regard, we presume the following knowledge, where T/C denotes the takeoff/cruise phase of the flight and L denotes the landing phase; "loss" of a functional task during a specified period of time (e.g., a phase) means failure to accomplish that task at some time during the specified period.

TABLE II
FUNCTION TABLE OF TRANSLATION κ_0

$u=(q_1,q_2,q_3)$	$\kappa_0(u)$
0 0 0	a_0
0 0 1	a_4
0 1 0	a_2
0 1 1	a_4
1 0 0	a_1
1 0 1	a_4
1 1 0	a_3
1 1 1	a_4

a) The flight is fuel efficient iff FC is accomplished throughout T/C and L.

b) The flight is diverted iff there is zero-visibility weather at the landing site (just prior to L) and either FC is lost during T/C or AL is lost during the last half of T/C.

c) The flight is unsafe iff AL is lost during an AL (attempted iff there is zero-visibility weather and the flight is not diverted) or the fuel consumption is "excessive"; excessive fuel consumption results iff FC is lost during both halves of T/C or during both T/C and L.

Translation between the level-1 and level-0 models (and thus the construction at level-1) will also rely on knowledge of the computational demands (workload) imposed by the aircraft, where we can presume the following.

d) FC is in demand throughout T/C and L (and hence is accomplished whenever it can be).

e) AL is not demanded during the first half of T/C, is in demand during the last half of T/C, and is in demand during L iff an AL is attempted [see condition c)].

Given this knowledge, we find that the set

$$Q_c^1 = \{0,1,\cdots,7\}$$

suffices as the composite state set, where the states $q \in Q_c^1$ are interpreted as follows (here, for task A and time period B, "A during B" means A can be accomplished throughout B; "no A during B" means the opposite, i.e., if A is demanded then A is lost during B):

$$q = \begin{cases} 0 & \text{if FC during } T/C \text{ and AL during the last half of } T/C, \\ 1 & \text{if FC during } T/C \text{ and no AL during the last half of } T/C, \\ 2 & \text{if FC during one half of } T/C \text{ and no FC during the other half,} \\ 3 & \text{if no FC during both halves of } T/C, \\ 4 & \text{if FC and AL during } L, \\ 5 & \text{if FC and no AL during } L, \\ 6 & \text{if no FC and AL during } L, \\ 7 & \text{if no FC and no AL during } L. \end{cases}$$

Note that our interpretation of these states is representative of "supply" as opposed to "demand," i.e., the computer's ability to supply the computations demanded by the aircraft; considerations of demand [conditions d) and e)] will be incorporated in the translation between Level-1 and Level-0. With this interpretation, it suffices to observe the system at the end of each phase, i.e.,

$$T_c^1 = \{t_2, t_3\}$$

where t_2 is the end of T/C ($t_2 = h - \frac{1}{2}$ hour) and $t_3 = h$. (Time t_1 is introduced later at level-2 of the hierarchy.) Accordingly, the composite part of the level-1 model is the process

$$X_c^1 = \{X_{c,t_2}^1, X_{c,t_3}^1\}, \tag{28}$$

where, by the definition of Q_c^1, X_{c,t_2}^1 takes values in $\{0,1,2,3\}$ and X_{c,t_3}^1 takes values in $\{4,5,6,7\}$; the composite trajectory space is therefore the Cartesian product

$$U_c^1 = \{0,1,2,3\} \times \{4,5,6,7\}. \tag{29}$$

The basic part of the level-1 model involves only the weather and is much easier to specify. Here

$$Q_b^1 = \{0,1\}$$

can serve as the state set where, if $q \in Q_b^1$, then

$$q = \begin{cases} 1 & \text{if there is zero-visibility weather at the landing site just prior to } L, \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, with this interpretation, a single observation at time t_2 (the end of T/C) suffices; hence, the basic part is the (single variable) process

$$X_b^1 = \{X_{b,t_2}^1\}.$$

To complete the construction according to the general procedure described in conjunction with Definition 3, X_c^1 and X_b^1 are then extended to the time base $T^1 = T_c^1 \cup T_b^1 = \{t_2, t_3\}$ so as to establish a common time base for both processes. Here, since $T_c^1 = T^1$, only X_b^1 requires extension, resulting in the process

$$X_b^1 = \{X_{b,t_2}, X_{b,t_3}\} \tag{30}$$

[where $X_{b,t_3} = ¢$; see (22)] with trajectory space

$$U_b^1 = \{0,1\} \times \{¢\}. \tag{31}$$

Combining the composite process X_c^1 (28) with the basic process X_b^1 (30), the level-1 model is described by the pair of processes

$$X^1 = (X_c^1, X_b^1)$$

with the (combined) trajectory space [see (29) and (31)]

$$\begin{aligned} U^1 &= U_c^1 \otimes U_b^1 \\ &= \{0,1,2,3\} \times \{4,5,6,7\} \times \{0,1\} \times \{¢\}. \end{aligned}$$

Finally, to establish that this model can indeed support the composite part (and hence all) of the level-0 model, we must be able to specify an interlevel translation $\kappa_1 : U^1 \to U_c^0$ [see Definition 3a)] which is consistent with the aircraft's in-flight behavior [conditions a)–c)] and the aircraft's computational demands [conditions d) and e)]. The required translation is obtained by applying just these conditions, for example, if $u \in U^1$ where $u = (2,5,1,¢)$ (which says FC can be accomplished during one-half but not all of T/C; FC can be accomplished during L but AL cannot be; there is zero-visibility weather at the landing site just prior to L) then, by conditions a)–e), it follows that $\kappa_1(u) = (1,1,0)$ (the flight is fuel inefficient, diverted, and safe). The remaining values of κ_1 are determined in a like manner.

Level-2: At this level, the computational capacity of the computer C is represented in terms of its ability to execute FC and AL computations during specified phases of the utilization period. We assume that this ability will be fully determined by the computer model at level-3 and, hence, the state set Q^2 coincides with the composite state set Q_c^2. The latter is taken to be the set

$$Q_c^2 = \{(q_1, q_2) \mid q_i \in \{0,1\}\}$$

where, for a given phase of T, the coordinates q_1 and q_2 are interpreted as follows:

$$q_1 = \begin{cases} 0 & \text{if FC computations can be executed throughout the phase,} \\ 1 & \text{otherwise} \end{cases}$$

$$q_2 = \begin{cases} 0 & \text{if AL computations can be executed throughout the phase,} \\ 1 & \text{otherwise.} \end{cases}$$

To support the composite part of the level-1 model (28), it suffices to distinguish three such phases, obtained from the level-1 phases by adding an observation time t_1 midway through T/C. More precisely, if t_2 and t_3 are as defined for level-1 then

$$T^2 = \{t_1, t_2, t_3\}$$

where $t_1 = t_2/2$, thereby distinguishing phase $[0,t_1]$ (first half of T/C), phase $[t_1,t_2]$ (second half of T/C), and phase $[t_2,t_3]$ (L). Accordingly, the level-2 model (comprised of a composite part only) is the process

$$X^2 = X_c^2 = \{X_{c,t_1}^2, X_{c,t_2}^2, X_{c,t_3}^2\}$$

where variable X_{c,t_i}^2 ($i = 1,2,3$) takes values in $Q_c^2 = \{0,1\}^2$, e.g., if $X_{c,t_2}^2 = (1,0)$ then the computer is unable to execute FC computations at some time during $[t_1,t_2]$ but has the resources to execute AL computations during this period. The level-2 trajectory space is therefore the set

$$U^2 = U_c^2 = \{0,1\}^2 \times \{0,1\}^2 \times \{0,1\}^2 \tag{32}$$

which is refined enough to admit an interlevel translation κ_2 from U^2 into U_c^1 (29). Specification of this translation follows immediately from the definitions of U^2 and U_c^1; for example, if $u \in U^2$ where $u = [(0,1),(1,1),(0,1)]$ (FC computations can be executed throughout the first and third phases, but not the second; AL computations cannot be executed throughout any

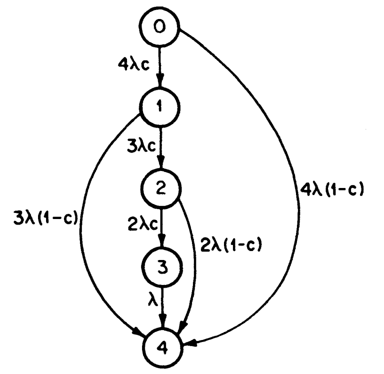


Fig. 2. Transition graph of the Level-3 Markov model.

TABLE III
COMPUTATIONAL CAPABILITY OF C

Level-3 state	Level-2 phase and computational task					
	$[0,t_1]$		$[t_1,t_2]$		$[t_2,t_3]$	
	FC	AL	FC	AL	FC	AL
0	X	X	X	X	X	X
1	X	X	X	X	X	X
2	X		X			X
3		X		X	X	
4						

phase), then $\kappa_2(u) = (2,5)$ (FC can be accomplished during one-half but not all of T/C; FC can be accomplished during L but AL cannot be).

Level-3: This model, at the bottom level of the hierarchy, describes variations in the computer's structure caused by faults which occur during utilization. Here we find that conventional stochastic models employed in reliability evaluation (e.g., continuous-time Markov models [7]-[9]) can adequately support higher level models of the type described previously. In particular, for this example, we suppose that C is a reconfigurable, fault-tolerant computer whose resources consist of four (essentially identical) processing subsystems. The integrity of these subsystems is represented by the state set

$$Q^3 = Q_b^3 = \{0,1,2,3,4\}$$

where state $q = i$ means that exactly i subsystems are faulty. Assuming that subsystems fail permanently at a constant rate λ (failures/hour) and letting c denote the "coverage" [4], the bottom model $X^3 = X_b^3$ is taken to be the continuous-time time-homogeneous Markov process described by the graph of Fig. 2. The level-3 trajectory space is the set $U^3 = U_b^3$ of all functions of the form

$$u: [0,h] \rightarrow Q^3$$

which can be realized by a Markov process of this type (see, for example, [15, ch. 8.3]).

To establish the interlevel translation κ_3: $U^3 \rightarrow U_c^2$, it is necessary to know the extent to which available computer resources (the fault-free subsystems) can support the execution of FC and AL computations during each of the level-2 phases. Generally, such knowledge will depend on a number of factors including the user's computational priorities during each phase, the computational demands of the computer's operating system, and the processing capacity of each (fault-free) subsystem. To avoid further elaboration (that would really not serve our purpose here), let us suppose these factors have already been examined, yielding the information summarized in Table III. An "X" entry in the table indicates that the resources of level-3 state i (designated by the row) are configured during the level-2 phase (designated by the column) so as to permit execution of the computational task (designated by the column); absence of an "X" signifies the contrary situation. Note that, as at level-1 and level-2 of the hierarchy, this information is representative of supply as opposed to demand, e.g., in states 0, 1, and 3 during phase $[0,t_1]$, the computer is able to execute AL computations even though it will not be called on to do so.

This information, along with certain properties of the base model X^3 (Fig. 2), suffices to determine the interlevel translation κ_3. For example, suppose that $u \in U^3$ (where u: $[0,h] \rightarrow Q^3$) where $u(0) = 0$, $u(t_1) = 2$, $u(t_2) = 3$, and $u(t_3) = 3$. Then, since resources fail permanently, it follows that $u(t) \in \{0,1,2\}$, for all $t \in [0,t_1]$; $u(t) \in \{2,3\}$, for all $t \in [t_1,t_2]$; and $u(t) = 3$, for all $t \in [t_2,t_3]$. This says, in turn (see Table III), that the computer's ability to execute AL computations is lost at some time during $[0,t_1]$, AL and FC are lost during $[t_1,t_2]$ (even though AL capability is recovered by the end of the phase), and there is no ability to execute AL computations throughout $[t_2,t_3]$ (due to a reassignment of computational priorities at the beginning phase 3). In other words, for the trajectory u in question, the only computations that are executed successfully are FC computation's throughout phases $[0,t_1]$ and $[t_2,t_3]$; hence $\kappa_3(u) = ((0,1),(1,1),(0,1))$, resulting in a trajectory that was illustrated earlier at level-2.

By similar arguments, each trajectory $u \in U_b^3$, when sampled at times $0,t_1,t_2$, and t_3, can be assigned a (unique) state trajectory $\kappa_3(u) \in U^3 = U_c^3$, thereby determining the interlevel translation κ_3. This, then, completes the specification of the four-level model hierarchy for S and, therefore, the performability model (X_S,γ_S) where $X_S = (X_b^3,X_b^1)$ and γ_S is determined by the interlevel translations κ_0-κ_3 according to (23).

To "solve" the model (i.e., evaluate the performability p_S), we apply the two step procedure stated at the outset of this section. Implementation of step 1), i.e., calculation of the trajectory sets $\gamma_S^{-1}(a)$ for each $a \in A$ (such sets suffice since A is finite, see Definition 1a, relies on the notion of a level-i based capability function [see (24) and (25)] and on successive formulations of $\gamma_i^{-1}(a)$ for $i = 0,1,2,3$ [see (26) and (27)]. Step 2) of the procedure, i.e., calculation of the probabilities $\Pr(\gamma_S^{-1}(a))$ for each $a \in A$, can be implemented via introduction of an equivalent "phased" base model and the subsequent application of specially developed formulas involving "interphase" and "intraphase" matrices. (See [28] for a detailed discussion of this technique.)

The solution procedure we have just outlined is a subject in itself and, without further elaboration, it would be difficult to illustrate its application. We can, however, illustrate the kind

TABLE IV
PERFORMABILITIES OF SYSTEMS S_1 AND S_2

a	$P_{S_1}(a)$	$P_{S_2}(a)$
a_0	0.999994	0.999821
a_1	3.4×10^{-6}	3.4×10^{-5}
a_2	0	0
a_3	2.6×10^{-6}	1.4×10^{-4}
a_4	2.2×10^{-9}	5.1×10^{-8}

of results that have been obtained for specific instances of the preceding example. In the first instance (S_1), the environment is a short flight from Washington, DC to New York City where the duration h is 1 h and the probability of zero-visibility (Category III) weather at JFK is 0.011. In the second instance (S_2), the environment is a longer flight from Washington, DC to Los Angeles where h is 5.5 h and the probability of Category III weather is 0.019. Both instances assume a computer C where the failure rate of each processing subsystem is $\lambda = 10^{-3}$, the coverage is ideal ($c = 1$), and the initial state X_0^3 is 0 with probability 1. The resulting performabilities are given in Table IV. Note that, in both instances, Level a_3 (low-fuel consumption, diversion, and safe) is impossible to accomplish, due to the probabilistic nature of the base model (Fig. 2) and the way computational tasks are allocated to available computer resources (Table III).

V. SUMMARY

Since traditional distinctions between performance and reliability become blurred when performance is degradable, we have proposed that the two be dealt with simultaneously via a unified measure called performability. After formalizing this measure in probability-theoretic terms, it was shown that performability is a natural component of system effectiveness and is a proper generalization of both performance and reliability. Performability modeling needs were then characterized via the concept of a capability function and it was demonstrated that capability is a proper extension of the kind of structure-behavior relationships that are typically assumed in reliability models. Finally, a hierarchical modeling scheme was introduced to facilitate both model construction and model solution and, in particular, to permit formulation of the capability function via interlevel translations. These concepts were then illustrated for an aircraft computing system.

It is hoped that the results of this paper can serve as a foundation for future work on unified performance-reliability models and (model-based) solution methods. Our experience to date with further developments [21], [28] and applications [25]-[27] of this methodology suggests that performability evaluation is indeed feasible, thereby providing a means of assessing the kind of performance-reliability interaction that is characteristic of degradable systems.

ACKNOWLEDGMENT

This research was conducted at the Systems Engineering Laboratory of the University of Michigan with the able cooperation of Research Assistants R. Ballance, D. Furchtgott, and L. Wu. The work has also benefited from valuable discussions with S. Bavuso at the NASA Langley Research Center, as well as with many other members of the NASA staff.

REFERENCES

[1] L. Svobodova, *Computer Performance Measurement and Evaluation Methods: Analysis and Applications.* New York: Elsevier, 1976.
[2] D. Ferrari, *Computer Systems Performance Evaluation.* Englewood Cliffs, NJ: Prentice-Hall, 1978.
[3] H. Kobayashi, *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology.* Reading, MA: Addison-Wesley, 1978.
[4] W. G. Bouricius, W. C. Carter, and P. R. Schneider, "Reliability modeling techniques for self-repairing computer systems," in *Proc. ACM 1969 Nat. Conf.,* Aug. 1969, pp. 295-305.
[5] B. R. Borgerson and R. F. Freitas, "A reliability model for gracefully degrading and standby-sparing systems," *IEEE Trans. Comput.,* vol. C-24, pp. 517-525, May 1975.
[6] H. B. Baskin, B. R. Borgerson, and R. Roberts, "PRIME-A modular architecture for terminal-oriented systems," in *1972 Spring Joint Computer Conf., AFIPS Conf. Proc.,* vol. 40. Washington, DC: Spartan, 1972, pp. 431-437.
[7] J. C. Laprie, "Reliability and availability of repairable structures," in *Dig, 1975 Int. Symp. Fault-Tolerant Computing,* Paris, France, June 1975, pp. 87-92.
[8] Y.-W. Ng and A. Avižienis, "A reliability model for gracefully degrading and repairable fault-tolerant systems," in *1977 Proc. Int. Symp. on Fault-Tolerant Computing,* Los Angeles, CA, June 1977, pp. 22-28.
[9] A. Costes, C. Landrault, and J.-C. Laprie, "Reliability and availability models for maintained systems featuring hardware failures and design faults," *IEEE Trans. Comput.,* vol. C-27, pp. 548-560, June 1978.
[10] M. D. Beaudry, "Performance-related reliability measures for computing systems," *IEEE Trans. Comput.,* vol. C-27, pp. 540-547, June 1978.
[11] R. Troy, "Dynamic reconfiguration: An algorithm and its efficiency evaluation," in *Proc. 1977 Int. Symp. Fault-Tolerant Computing,* Los Angeles, CA, June 1977, pp. 44-49.
[12] J. Losq, "Effects of failures on gracefully degradable systems," in *Proc. 1977 Int. Symp. Fault-Tolerant Computing,* Los Angeles, CA, June 1977, pp. 29-34.
[13] P. E. Pfeiffer, *Concepts of Probability Theory.* New York: McGraw-Hill, 1965.
[14] E. Wong, *Stochastic Processes in Information and Dynamical Systems.* New York: McGraw-Hill, 1971.
[15] E. Çinlar, *Introduction to Stochastic Processes.* Englewood Cliffs, NJ: Prentice-Hall, 1975.
[16] B. P. Zeigler, *Theory of Modelling and Simulation.* New York: Wiley, 1976.
[17] F. P. Mathur and A. Avižienis, "Reliability analysis and architecture of a hybrid-redundant digital system: Generalized triple modular redundancy with self-repair," in *Proc. 1970 Spring Joint Computer Conf., AFIPS Conf.,* vol. 36. Washington, DC: Spartan, 1970, pp. 375-383.
[18] J. F. Meyer, "Models and techniques for evaluating the effectiveness of aircraft comput. systems," Semiannu. Status Rep. 3, NASA Rep. CR158992 (NTIS Rep. N79-17564/2GA), Jan. 1978.
[19] Z. W. Birnbaum, J. D. Esary, and S. C. Saunders, "Multicomponent systems and structures and their reliability," *Technometrics,* vol. 3, pp. 55-77, Feb. 1961.
[20] J. D. Esary and H. Ziehms, "Reliability of phased missions," in *Reliability and Fault Tree Analysis,* SIAM, Philadelphia, PA, 1975, pp. 213-236.
[21] R. A. Ballance and J. F. Meyer, "Functional dependence and its application to system evaluation," in *1978 Proc. Johns Hopkins Conf. Inform. Sciences and Systems,* Johns Hopkins, Univ., Baltimore, MD, Mar. 1978, pp. 280-285.
[22] J. F. Meyer, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Semiannu. Status Rep. 4, NASA Rep. CR158993 (NTIS, Rep. N79-17563/4GA), July 1978.
[23] A. L. Hopkins, Jr., T. B. Smith, III, and J. H. LaLa, "FTMP—A highly reliable fault-tolerant multiprocessor for aircraft," *Proc. IEEE,* vol. 66, pp. 1221-1239, Oct. 1978.

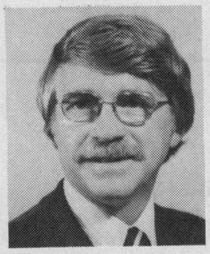
[24] J. H. Wensley, L. Lamport, J. Goldberg, M. W. Green, K. N. Levitt, P. M. Melliar-Smith, R. E. Shostak, and C. B. Weinstock, "SIFT: Design and analysis of a fault-tolerant computer for aircraft control," *Proc. IEEE,* vol. 66, pp. 1240–1255, Oct. 1978.

[25] D. G. Furchtgott and J. F. Meyer, "Performability evaluation of fault-tolerant multiprocessors," in *1978 Dig. Government Microcircuit Applications Conf.,* Monterey, CA, Nov. 1978, pp. 362–369.

[26] J. F. Meyer, D. G. Furchtgott, and L. T. Wu, "Performability evaluation of the SIFT computer," in *Proc. 1979 Int. Symp. Fault-Tolerant Computing,* Madison, WI, June 1979, pp. 43-50.

[27] J. F. Meyer, D. G. Furchtgott, and L. T. Wu, "Performability evaluation of the SIFT computer," *IEEE Trans. Comput.,* vol. C-29, pp. 501–509, June 1980.

[28] L. T. Wu and J. F. Meyer, "Phased models for evaluating the performability of computing systems," in *1979 Proc. Johns Hopkins Conf. Information Sciences and Systems,* Baltimore, MD, Mar. 1979, pp. 426–431.



John F. Meyer (M'60–SM'71) received the B.S. degree from the University of Michigan, Ann Arbor, the M.S. degree from Stanford University, Stanford, CA, and the Ph.D. degree in communication sciences, also from the University of Michigan, in 1957, 1958, and 1967, respectively.

He is currently a Professor in the Department of Electrical and Computer Engineering and the Department of Computer and Communication Sciences, University of Michigan. He is also associated with their graduate program in computer, information, and control engineering and is a member of the Systems Engineering Laboratory. In addition to his university affiliations, he is a consultant to several firms. During the past 20 years, he has been active in computer research and has published widely in the areas of system modeling and fault-tolerant computing. In the summer of 1977 he was a Visiting Researcher at the Laboratoire d'Automatique et de d'Analyse des Systèmes, Toulouse, France, and in 1975, during an academic leave, he was affiliated with the Direction de l'Informatique de la Société Thomson-CSF, Paris. Prior to joining the Michigan faculty in 1967, he was a Research Engineer at the California Institute of Technology Jet Propulsion Laboratory where his contributions included the first patent issued to the National Aeronautics and Space Administration.

Dr. Meyer is a member of Sigma Xi, Tau Beta Pi, Eta Kappa Nu, the Association for Computing Machinery, and the American Association for the Advancement of Science. In the IEEE Computer Society, he served as Chairman of the Technical Committee on Fault-Tolerant Computing from 1976–1979. He is also a member of the Publications Committee and has served as a Guest Editor of the IEEE TRANSACTIONS ON COMPUTERS.