THE UNIVERSITY OF MICHIGAN

COMPUTING RESEARCH LABORATORY[1]

MODELS FOR EVALUATING THE PERFORMABILITY OF DEGRADABLE COMPUTING SYSTEMS

Liang Tai Wu

CRL-TR-7-82

JUNE 1982

Room 1079, East Engineering Building
Ann Arbor, Michigan 48109
USA
Tel: (313) 763-8000

---

[1]This work was supported by NASA Grant NSG 1306. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies.

ABSTRACT

MODELS FOR EVALUATING THE PERFORMABILITY OF DEGRADABLE COMPUTING SYSTEMS

by

Liang Tai Wu

Chairman: John F. Meyer

Recent advances in multiprocessor technology have established the need for unified methods to evaluate computing systems performance and reliability. In response to this modeling need, this dissertation considers a general modeling framework that permits the modeling, analysis and evaluation of degradable computing systems. Within this framework, several user-oriented performance variables are identified and shown to be proper generalizations of the traditional notions of system performance and reliability. Furthermore, a time-varying version of the model is developed to generalize the traditional fault-tree reliability evaluation methods of phased missions.

The modeling and evaluation methods considered in this dissertation provide a relatively straightforward approach to integrate reliability and availability measures with performance measures. The hierarchical decomposition approach permits the modeling and evaluation of a computing system's subsystems (e.g., hardware, software, peripherals, interfaces, user demand systems) as a whole rather than the traditional methods of evaluating these subsystems independently. Accordingly, it becomes possible to evaluate

the performance of the system software and the reliability of the system hardware simultaneously in order to measure the effectiveness of the system design. Moreover, since the performance variables considered in this study permit the characterization of system performance according to the application needs of a system, the results obtained represent more accurate assessments of the system's ability to perform than the existing performance or reliability measures.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Table

CHAPTER 1

INTRODUCTION

1.1 Background

The recent developments in multiprocessor systems have stimulated a growing interest in degradable computing systems that are designed to provide a high degree of performance and reliability by reallocating the computer's resources when faults are detected. To assess the effectiveness of these computing systems, it has been found that the traditional way of evaluating the performance and the reliability as distinct attributes of a computer is no longer adequate [1], [2]. Traditional performance evaluation methods generally assume that the computer to be evaluated is fault free and are concerned with the quantification of the effectiveness in which the computer's resources handle a specific application (see [3] and [4], for example). Traditional reliability evaluation methods, on the other hand, deal with the measurement of a computer's ability to remain operational in the event of physical failures (see [5] through [8]). Since the level of performance of a degradable computing system may decrease with successive failures, the performance and the reliability of the system must be dealt with simultaneously to measure the extent to which the user can benefit from the tasks accomplished by the computer.

In response to the above modeling need of degradable computing systems, some recent investigations have attempted to formulate new modeling and evaluation methods that combine both the performance and the reliability characteristics of computing systems. Particularly, Beaudry [9] has considered performance-reliability measures that reflect the computational capacity of a system, defined as the amount of useful computations available per unit of time, and has shown that these measures can be evaluated in terms of a transformed Markov process. By examining the set of jobs executed by a computing system, Mine and Hatayama [10] have considered the reliability of the system with respect to a specific job, called job-related reliability. Although the above models have shown the feasibility of combining the performance and reliability measures into a single measure, their efforts have focused mainly on the maximum capacity at which a computer can handle its computation. The effect of interactions between the demand for computation (by the user) and its supply (by the computer) has not been considered explicitly.

Another approach to quantifying the unified performance and reliability of computing systems is based on Markov reward processes [11]. By assigning a throughput rate to each state of a Markov process that describes the resource availability of a computing system, Gay [12] has considered the expected system throughput and the throughput availability of a system. A similar model has also been used in [13] by De Souza to estimate the reduction in operating cost when fault-tolerance features are incorporated in commercial systems. More recently, based on renewal process models, Castillo and

Siewiorek [14] have considered the apparent capacity and expected elapsed time required to execute a program correctly.

In contrast to the above efforts to formulate specific performance measures for degradable computing systems, Meyer [1] has developed a general modeling framework that permits the definition, formulation and evaluation of user-oriented performance measures. A hierarchical model is defined [1] which assumes that the probabilistic nature of the total system S (the computer and its environment) is modeled by a stochastic process X_S. It is further assumed that the process X_S can be used to determine the probability distribution function of a random variable Y_S which describes the user's view of how well the system performs. The probability distribution function of Y_S is shown to induce a useful performance measure, referred to as the performability of S, in the context of degradable computing system performance.

1.2 Research Objectives

In this investigation, among other things, we wish to extend the modeling framework in [1] to provide a more concrete basis for studying the evaluation of degradable computing systems. By introducing extra ingredients to the modeling framework, we wish to develop a general stochastic process model of degradable computing systems that satisfies the following objectives:

(1) The model should be general enough to permit uniform formulation of different performance measures.

(2) The model should be specific enough to permit derivations of computational algorithms and formulas.

(3) The model should be flexible enough to be related to traditional performance and reliability models so that it may serve as a basis for unifying traditional computing system evaluation methods.

(4) The model should be able to reflect the information processing needs of the user as well as internal structural changes of the system caused by component failures.

In addition to the above efforts of model development, we also wish to apply the results obtained to evaluate a large class of fault-tolerant computing systems known as "degradable multiprocessor systems." By comparing the effectiveness of various design strategies, we wish to illustrate the tradeoffs between different techniques of incorporating fault-tolerance in the design of a multiprocessor system.

Chapter 2 puts this work in context with respect to the general modeling framework considered in [1]. It describes the components of a performability model and formalizes the relationships among these components. The major results of this chapter include the precise formulation of the notion of system performance in a broad context and the clarification of the notion of supporting the evaluation of system performance using a stochastic process model.

Chapter 3 introduces a general notion of recoverability and establishes necessary and sufficient conditions for an operational model to be recoverable. For both the recoverable and the nonrecoverable models, it examines the solution methods of a generally defined performance variable where the performance is identified with the minimum value of a functional. The modeling approach and the evaluation methods are then illustrated through the evaluations of a multiprocessor system. The results obtained indicate that the performance variable is, indeed, a proper generalization of the traditional notions of the system performance and reliability. The modeling and the evaluation methods proposed thus represent a unifying approach for integrating the performance and the reliability measures of computing systems.

Chapter 4 presents a specific operational model for evaluating the performability of degradable multiprocessor systems. The model is constructed according to a hierarchical decomposition of a system's behavior. A Markovian base model is developed to represent the resource availability of the system, and priority queueing models are used to determine the operational rates of the resource states. The model not only demonstrates the generality of an operational model but also illustrates the feasibility of modeling and evaluating the system performance via a step-by-step hierarchical approach. The methods developed in this chapter thus represent a straightforward approach to produce a composite picture of a computer's ability to meet overall throughput goals.

Chapter 5 extends the concept of an operational model to phased

missions where the environment of a system can vary in time. Both the combinatorial and the probabilistic properties of the extended model are examined in detail. In addition, an example is constructed to illustrate the performability evaluation of a phased mission with multiple accomplishment levels. The results obtained in this chapter represent an important step toward the understanding and the development of a more general time-varying operational model.

Chapter 6 summarizes the results of this study and suggests topics for further research.

CHAPTER 2

PERFORMABILITY EVALUATION OF COMPUTING SYSTEMS

2.1 Introduction

The concept of hierarchical organization has become an important tool in the design of computing systems. Hardware components are typically formed by putting together some basic modules or building blocks, and software components are often structured into subroutines in a top-down manner. By carefully organizing the structure of a computer into a hierarchy of components, it becomes possible to increase greatly the capability and the functional features of the computer. Although this concept of hierarchical organization has been used extensively in the design of computing systems since the invention of the first electronic computer, its implication in computing systems performance evaluation has only been exploited recently.

As suggested in [2], a computing system can be described by a hierarchy of system models that vary in "scope" and "level of abstraction" (see Figure 2.1 for an example of what we call a model hierarchy). In this representation, a higher level model has a larger scope and a higher level of abstraction i.e., it describes a larger portion of the computer and its environment, but possibly in less amount of detail. In particular, the top model has a scope that includes all the subsystems that can influence the computational process of the system (e.g., hardware and software, peripherals,

interfaces, maintenance systems, user demand system, etc., collectively referred to as the *total system*). The level of abstraction at the top level is expressed in a form easily usable by the system user. On the other hand, the bottom model may involve low level representations of the computer's hardware and operating system structure.

Based on the above model hierarchy of the total system, various performance and reliability measures can then be associated with models at each level of the hierarchy. The part of the total system that one is interested in evaluating must be identified first with a specific level in the hierarchy (referred to as the *object system*). The part of the total system outside the object system is then regarded as the *environment* of the object system. The choice of an object system is, to a large extent, determined by the particular problem one is interested in solving. For example, if the performance or reliability of a data-base system is to be evaluated, the object system will not only include the hardware and operating system but also the data bases and their supporting programs.

Once a specific object system is selected, the *performance* of the system can then be defined as how well the object system satisfies the computational demands (also referred to as the *workload*) imposed by its environment [4]. The performance is typically considered as a random variable

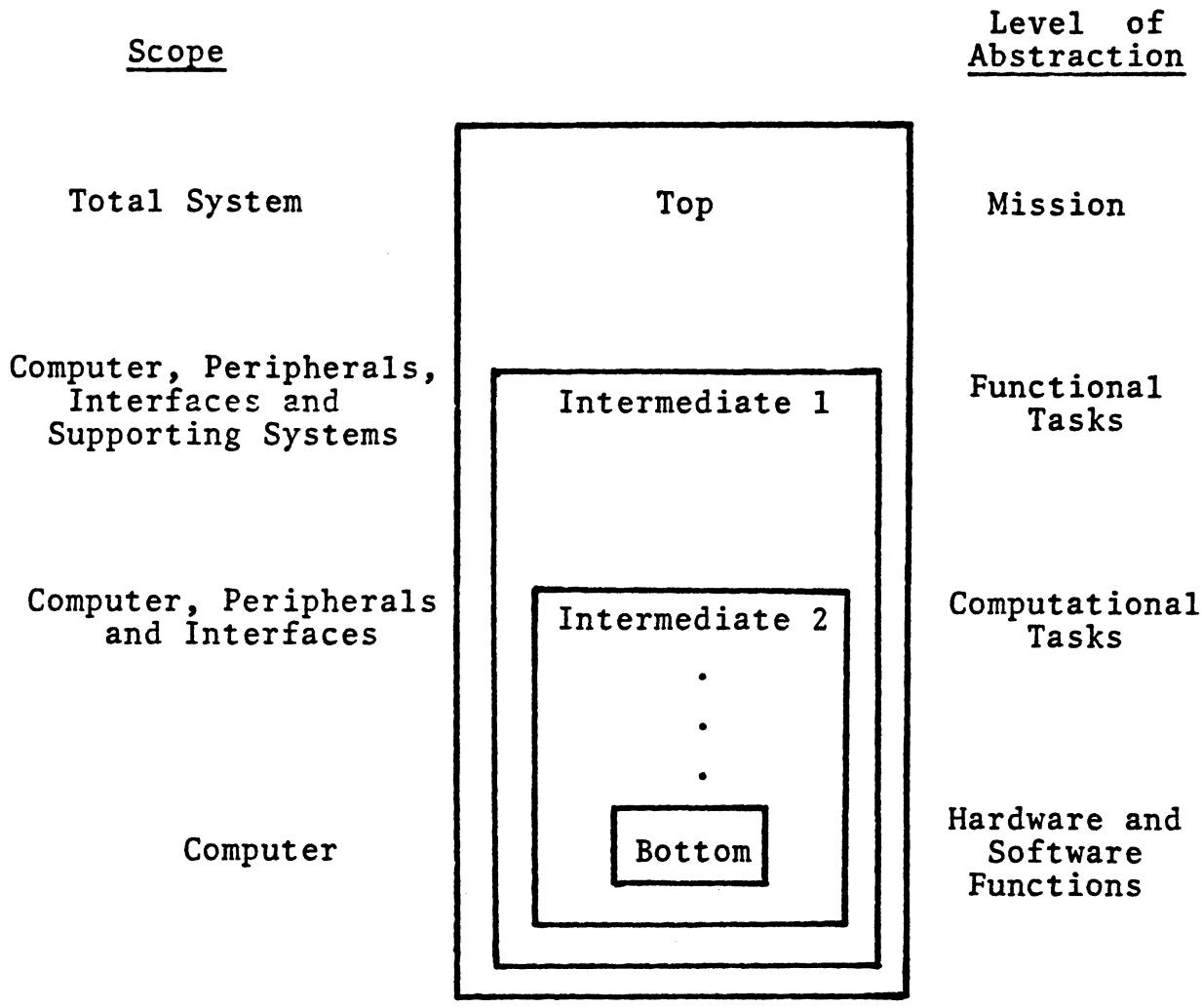


Figure 2.1

A Model Hierarchy

referred to as a *performance variable* or *performance index*. Thus we can talk about the mean, variance, distribution function, and the like of a performance variable.

If we regard computer performance measures to be the measurements of the quality of a computer according to the above broad definition, some observations about the relationships among different performance measures can now be made. First, we note that there are no essential differences between what are traditionally called performance measures (e.g., throughput rate, response time or utilization rate) and what are traditionally called reliability measures (e.g., reliability, availability or maintainability). They differ only in the way performance criteria are formed. Any performance or reliability measures, viewed in the broadest context, must account for both the workload and the probabilistic nature of the object system. Accordingly, in the discussion that follows, the term performance measure will be used to include both the performance and the reliability aspects of a system. Second, we also note that different performance measures can be associated with different levels of the model hierarchy. Thus, for each level of the hierarchy, it is possible to formulate various performance measures according to the application needs as well as the modeling requirements of the object system.

In the following section, we first define more precisely the basic elements of a performance study by considering the basic components of a

performability model. The relationships among those components are then formalized via the concept of capability functions. Finally, the notion of supporting the evaluation of system performance using a stochastic process is made precise by relating the probabilistic nature of the performance variable to the known properties of the underlying stochastic process.

2.2 System Models

A major objective of this section is to formulate precisely what we mean by a "stochastic model" for system performability. It is assumed that the total system S=(C,E) contains a computer C operating in an environment E. The computer C is composed of several processors, memory modules, input/output devices, buses, etc., and the environment E includes man-made components (e.g., interface circuits and peripheral subsystems), operational rules (e.g., job submitting policies and maintenance procedures) and other conditions (e.g., weather) that can influence the computer's effectiveness. At this level of abstraction, it is appropriate to view S as a network of interconnected subsystems with simultaneous information flow among subsystems. Accordingly, S can be described as an autonomous state transition system that changes state due to events occurring in time.

Given the above characterization of the total system, the behavior of S can be viewed as a stochastic process

$$X_S = \{X_t | t \epsilon T\} \tag{2.1}$$

where T is the time range involved (called the *utilization period*) and, for each $t \epsilon T$,

$$X_t: \Omega \rightarrow Q$$

is a random variable defined on a common probability space (Ω, E, P) and taking values in the *state space* Q of the total system. In the following discussions, it will be assumed that T is a set of real numbers and Q is a discrete set. Thus, without los of generality, the states of X_S will often be named by positive integers, viz.

$$Q = \{1,2,3,...\}$$

or, when Q is finite,

$$Q = \{1,2,...,n\} .$$

The stochastic process X_S will be referred to as the *base model* of S and is denoted simply as X when the system context is clear.

Although the base model X provides a detailed description of the system's state behavior, the description is generally invisible to the users. It is assumed that the users are concerned only with distinguishing different "levels

of accomplishment" when judging how well the system has performed. Accordingly, the user's view of the system's behavior can be formulated as a random variable with respect to the underlying probability space (Ω,$\mathcal{E}$,P), i.e.,

$$Y: \Omega \rightarrow A \tag{2.2}$$

where, for each $\omega \in \Omega$, $Y(\omega)$ takes a value in the *accomplishment set* A. Depending on the application, the accomplishment set A can be any set of real numbers where the elements of A are taken to be the various degrees of user satisfaction such that a>b if a is preferred over b (i.e., the ordering relation > as implied by the user preference coincides with the natural ordering of real numbers). For example, to evaluate the reliability of a nondegradable system, the accomplishment set can be taken to be A={0,1} where 1 = "system success" and 0 = "system failure." On the other hand, if the user is interested in evaluating the system throughput, A can be taken to be an interval of real numbers. The random variable Y will be referred to as a *performance variable* of S.

As generally defined above, the performance variable Y clearly can be used to characterize either the performance or the reliability aspects of a system. Thus a natural measure that can be used in the evaluation of computing systems is the probability measure induced by Y (e.g., see [15] p.97). This unified performance-reliability measure is referred to as the *performability of S* which, in terms of our modeling framework, can be defined as the function $perf_S$ where, for each measurable set $B \subseteq A$ (i.e.,

$\{\omega | Y(\omega) \epsilon B\} \epsilon E$) of accomplishment levels,

$$perf_S(B) = P(\{\omega | Y(\omega) \epsilon B\}), \tag{2.3}$$

i.e., $perf_S(B)$ is the probability that S performs at a level in B. The requirement that B be measurable insures the existence of the probability on the right side of (2.3).

In theory, it is possible to determine the performability of S from the underlying probability space (Ω, E, P) of the performance variable Y. However, in practice, the underlying probability space is generally unknown and, consequently, the performability must be determined from known properties of the base model X. Hence an important step to determine the performability of S is to establish relations between the base model X and the performance variable Y based on the given properties of X.

Following a common practice in probability theory, we assume that the base model X is specified by its finite-dimensional distributions or by information that determines these distributions (e.g., Markov assumptions together with a transition function and an initial distribution). Based on these finite-dimensional distributions, we then construct a "coordinate probability space" (see [16] or [17] for the details of this construction) and express the performance variable Y in terms of an equivalent random variable defined on the new probability space. The advantage of this approach is that the resulting

probability space is considerably structured and, hence, questions about the probabilistic nature of X and Y can be addressed more easily by dealing with the coordinate probability space. The construction of this probability space described in [16] is summarized as follows.

Suppose that the base model X is described by a family of finite dimensional distributions

$$\Phi = \{ F_{t_1,\ldots,t_n} \mid t_1, \ldots, t_n \epsilon T \text{ and } n \epsilon N \} \tag{2.4}$$

where T is the utilization period and N is the set of all natural numbers. Then the *coordinate probability space* is a probability space (U,*F*,Pr) where

1. The coordinate sample space U is the set of all functions u:T $\rightarrow$ Q where Q is the state space of X. In other words, U is the $|T|$-dimensional direct product of the state space Q.

2. To construct the event space *F*, let $\mathbf{B}^n$ be the smallest σ-algebra generated by the relative topology of Q^n (also referred to as the topology of Q^n induced by the n-dimensional Euclidean space). For each set B in $\mathbf{B}^n$ and given $t_1,\ldots,t_n$ in T, let

$$C = \{ u \epsilon U \mid [u(t_1),\ldots,u(t_n)] \epsilon B \}, \tag{2.5}$$

i.e., C is the set of all functions u in U such that the values of u at t_i ($1 \leq i \leq n$), when regarded as an n-tuple, is an element in B. If we let F_0 be the set of all C such that C is obtained from (2.5) for all $n \epsilon N$, all $B \epsilon B^n$, and all $t_1, \ldots, t_n \epsilon T$, then F_0 is a field. Finally, the event space F is taken to be the completion of the smallest σ-algebra containing F_0.

3. To construct the probability measure Pr, we first define a measure μ on F_0 such that for each $C \epsilon F_0$,

$$\mu(C) = \int_B \cdots \int dF_{t_1, \ldots, t_n}(\eta_1, \ldots, \eta_n) \tag{2.6}$$

where C is generated by the set B with indices $t_1, \ldots, t_n$ (see (2.5)) and $F_{t_1, \ldots, t_n}$ is a multivariate distribution in Φ (see (2.4)). Then the probability measure Pr is the completed version of the above measure μ.

Given the coordinate probability space (U,F,Pr) of X, we can construct an equivalent process of X defined on (U,F,Pr) such that both processes have the same multivariate distributions Φ. More precisely, let us define

$$\hat{X} = \{\hat{X}_t | t \epsilon T\}$$

where, for all $t \epsilon T$ and all $u \epsilon U$,

$$\hat{X}_t(u) = u(t). \tag{2.7}$$

Furthermore, for all B in $\mathbf{B}^n$ and all $t_1,\ldots,t_n$ in T, let us assign (see (2.6))

$$\int_{B} \cdots \int dF_{t_1,\ldots,t_n}(\eta_1,\ldots,\eta_n)$$

to be the probability of the event

$$\{ u\epsilon U \mid [\hat{X}_{t_1}(u),\ldots,\hat{X}_{t_n}(u)]\epsilon B \}.$$

Then, for each $t\epsilon T$, $\hat{X}_t$ is a function from the set U to the set Q and the family of functions $\hat{X}$ is a stochastic process defined on (U,*F*,Pr) having the multivariate distributions Φ (see [17], pp. 10-11). Since, for each $u\epsilon U$, u(t) specifies the state of $\hat{X}$ at t (see (2.7)), each element in U will be referred to as a *state trajectory* and the set U will be referred to as the *trajectory space.*

The notion of a coordinate probability space permit us to answer questions about the probabilistic nature of X by relating the questions to the state behavior of $\hat{X}$. In particular, for each fixed $\omega\epsilon\Omega$ of the underlying probability space (Ω,*E*,P) let us define a function u_ω: T → Q such that, for all $t\epsilon T$,

$$u_\omega(t) = X_t(\omega). \tag{2.8}$$

Moreover, given an event V in F, let

$$W = \{\omega | u_\omega \epsilon V\} \tag{2.9}$$

be a subset of Ω. Then, it is known (see [17], pp. 621-622) that W is an event in E and that

$$P(W) = Pr[V] . \tag{2.10}$$

On the other hand, given a subset W of Ω measurable with respect to the induced probability space of X, there exists an event V in F such that (2.10) is satisfied.

In the following discussions, we consider the question of what we need to know about X in order to determine the distribution function of the performance variable Y. Formally, we says that X *supports* Y if there exists a random variable

$$\gamma: U \rightarrow A \tag{2.11}$$

defined with respect to the coordinate probability space (U,F,Pr) such that for each $\omega \epsilon \Omega$

$$Y(\omega) = \gamma(u_\omega) \tag{2.12}$$

where u_ω is the state trajectory associated with the outcome ω. Since γ can be regarded as the user's performance criteria for judging the "capability" of the total system, it is referred to as the *capability function* of S.

When X supports Y, the capability function permits us to determine the performability of S using the finite-dimensional distributions of X. To substantiate this claim, let us define a function

$$h: \Omega \rightarrow U$$

such that, for all $\omega \epsilon \Omega$, $h(\omega)=u_\omega$ where u_ω is the state trajectory associated with the outcome ω. Then, by (2.12), X supports Y implies

$$Y = \gamma \cdot h ,$$

i.e., Y is the functional composition of γ and h, applying h first. Accordingly, taking the preimage on both side, we have

$$Y^{-1} = h^{-1} \cdot \gamma^{-1}.$$

Hence, for any measurable set $B \subseteq A$, if we let

$$V = \{u|\gamma(u)\epsilon B\}\subseteq U$$

and

$$W = \{\omega|Y(\omega)\epsilon B\}\subseteq \Omega\ ,$$

then

$$W = \{\omega|u_\omega \epsilon V\} = h^{-1}(V)\ .$$

Accordingly, by (2.10), $P(W) = Pr[V]$, which, in turn, implies

$$\begin{aligned} perf_S(B) &\underline{\Delta} P(\{\omega|\dot{Y}(\omega)\epsilon B\}) \\ &= Pr[\gamma^{-1}(B)]\ . \end{aligned} \tag{2.13}$$

Since the probability $Pr[\gamma^{-1}(B)]$ can be determined directly from the finite-dimensional distributions of X, we have shown that $\hat{X}$ together with γ suffice to support an evaluation of the performability $perf_S$.

In view of what has been observed, if X supports Y, then the pair $(\hat{X},\gamma)$ is said to constitute a *performability model* of S. If B is a measurable set of accomplishment levels, the inverse image $\gamma^{-1}(B)$ 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^{-1} to trajectories of the base model. In the following

discussions, since we will be dealing with $\hat{X}$ instead of X, the induced process $\hat{X}$ will be called the base model and denoted simply as X.

Given a performability model (X,γ), equation (2.13) permits us to evaluate the performability of S for a set B of accomplishment levels by i) determining the trajectory set $\gamma^{-1}(B)$ and ii) calculating $Pr[\gamma^{-1}(B)]$. In general, the trajectory set $\gamma^{-1}(B)$ is difficult to obtain because the "distance" between the base model X and the performance variable Y may be considerable. The difficulty can be alleviated by introducing intermediate models between X and Y based on the concept of a model hierarchy discussed in Section 2.1. The use of a model hierarchy allows the capability function or, more accurately, the trajectory set $\gamma^{-1}(B)$ to be derived step-by-step in a top-down manner from more elementary components in a clearly conceived way. In particular, by introducing an intermediate model called an "operational model," we show in the following chapter that the performability of S can be determined by evaluating the intermediate model.

Finally, we note that the role of a capability function in performability evaluation is similar to that of a structure function [18] in reliability evaluation. However, even when performability is restricted to reliability, the concept of a capability function is still more general because a capability function must take into account the behavior of S throughout the utilization period while a structure function is restricted to modeling the instantaneous behavior of S at a given moment in time [2].

CHAPTER 3

OPERATIONAL MODELS

3.1 Introduction

When modeling degradable computing systems by stochastic processes for system performance or reliability evaluation, the models used are typically Markov processes (see [8], for example) or models which can be analyzed in terms of embedded Markov processes (for example, certain queueing models such as M/G/1 or GI/M/m queues; see [21]). However, to ensure the validity of the Markov assumption, it is usually necessary to model the structure and behavior of the system at a low level, e.g., a level describing the system's physical resources (processing units, memory units, input buffers, etc.). Performance and reliability measures, on the other hand, often quantify the system's behavior in terms of high-level, user-oriented variables (throughput, response time, operational status, etc.) which, if viewed as stochastic processes, are seldom Markovian. In such cases, an essential part of the modeling effort is to establish a "connection" between the low and high levels to resolve the probabilistic nature of the measure in question.

Historically, in the context of reliability modeling, this connection has taken a form that lies at one of two extremes. At one extreme, system "success" is defined in terms of the underlying structural resources (at least so

many fault-free processors, at least so many fault-free memory units, etc.), in which case the connection between structure (available fault-free resources) and performance (success or failure) is immediate. At the other extreme, the object of the modeling effort is the connection, per se, and the resulting model is typically some form of event-tree or fault-tree (see [18], for example).

In general, as discussed in the previous chapter, the general nature of this connection can be formalized as a capability function of the system. In this setting, a total system S, comprising a computing system and its computational environment, is modeled at a low level by a stochastic process X (the base model of S). Then, relative to a high level variable Y (the performance of S), the capability function of S is a function γ which translates state trajectories (sample paths) of the process X into corresponding values of the performance variable Y. Knowing X and γ, it is possible to solve for the probability distribution function of Y and, hence, determine the performability of S.

When the performance variable Y is far removed from the base model X, solution procedures can be simplified by introducing intermediate model at levels between X and Y. One use of such a model hierarchy is a step-by-step formulation of the preimage of γ beginning at Y and terminating at the base model X. If Y is discrete, the performability of S can then be evaluated by determining the probabilities of certain trajectory sets that correspond (under γ^{-1}) to performance values of Y. Another role that can be

played by an intermediate model, and the one we explore in this chapter, is to represent the probabilistic nature of S at a level that is higher than the base model and thus "closer" to the performance variable.

To characterize the behavior of an intermediate model, Section 3.2 introduces a general notion of recoverability and shows that a performance process is nonrecoverable if and only if the state behavior of the process can be determined by taking a "snapshot" at the end of the utilization period. For both the recoverable and nonrecoverable models, Section 3.3 examines the solution methods of a generally defined performance variable where the performance is identified with the minimum value of a functional. The modeling and the solution methods are then illustrated in Section 3.4 through the evaluation of a degradable computing system. The results of the evaluation indicate that the performance variable considered in this chapter is a proper generalization of the traditional notions of the system performance and reliability. The modeling and the evaluation methods considered thus provide a unifying approach for evaluating the integrated performance and the reliability of degradable computing systems.

3.2 Recoverability

Generally, in reliability modeling, a system is said to be repairable or nonrepairable according to whether maintenance actions are permitted during its utilization to reduce the incidence of system failure or to return a failed system to an operating state. The classification is useful because, when a

system is nonrepairable, the computation of system reliability at a time t amounts to calculating the probability that the system functions at that moment in time (see [18], for example). On the other hand, when a system is repairable, the computation requires a more complete knowledge of the system's behavior during the entire utilization period T.

The above classification and properties of reliability models can be extended to models of degradable computing systems by considering the way in which system performance may change in time. The generalization not only permits us to obtain a better understanding of the performance degradation of a degradable computing system, but also provides us with a common basis for unifying traditional performance and reliability methods.

To begin, let us define an *operational model* to be a stochastic process

$$Z = \{Z_t | t \epsilon T\}$$

with $Z_t: \Omega \rightarrow Q$ such that the state space Q of Z is partially ordered by some partial ordering $\leq$. The partial ordering $\leq$ can be interpreted as the ranking of system states according to the degree of user satisfaction with the system operating in a given state (hence the term "operational"). Although operational models are introduced here to characterize intermediate-level models, it should be noted that operational models can often be defined at the base model level with some natural ordering of states. For instance, consider a system S

containing m subsystems where each of them can be in one of two operational conditions; functioning or failed. Then, one natural way to define a state space for S is by taking

$$Q = \{0,1\}^m \tag{3.1}$$

and, assuming no compensating effects of successive failures, the state space can be ordered by taking the Cartesian product of the component ordering relations, i.e., for all $(a_1,a_2, \ldots ,a_m)$ and $(b_1,b_2,...,b_m)$ in Q, let

$$\begin{aligned} &(a_1,a_2, \cdots ,a_m) \leq (b_1,b_2,...,b_m) \\ &\text{if and only if } a_i \leq b_i \text{ for all } 1 \leq i \leq m \ . \end{aligned} \tag{3.2}$$

The above ordering of component states is a standard practice in reliability theory and plays an important role in fault-tree analysis (see [18], for example). The applicability of operational models in modeling degradable computing systems will be discussed in more detail in the next section.

Given an operational model Z, the concept of repairability can be extended as follows. We say Z is *nonrecoverable* if, for all $s,t \in T(s \leq t)$ and all $i,j \in Q$,

$$Pr[Z_s=i, Z_t=j] > 0 \text{ implies } i \geq j \ .$$

In less formal terms, an operational model is nonrecoverable iff its operational status can only degrade monotonically in time. Moreover, by considering the contrapositive form of the above condition, it follows that Z is nonrecoverable iff for all $s,t \epsilon T$ ($s \leq t$) and all $i,j \epsilon Q$,

$$i \not\geq j \text{ implies } \Pr[Z_s=i, Z_t=j] = 0. \tag{3.3}$$

Similarly, we say Z is *recoverable* if it is not nonrecoverable, i.e., if there exist $s,t \epsilon T$ ($s \leq t$) and $i,j \epsilon Q$, such that

$$i \not\geq j \text{ and } \Pr[Z_s=i, Z_t=j] > 0 . \tag{3.4}$$

In other words, Z is recoverable if there is a nonzero probability that the state of the system may "recover" from a degraded state i to a higher level state j ($j > i$) or to a noncomparable state j ($j \not\geq i$ and $j \not\leq i$).

The notion of nonrecoverability can be characterized in a number of useful ways, as indicated by the following theorem.

Theorem 3.1:

Let Z be an operational model with state space Q. Then the following statements are equivalent:

(1) Z is nonrecoverable.

(2) For all $s,t \epsilon T$ $(s \leq t)$ and all $k \epsilon Q$

$$\Pr[Z_s \not\geq k, Z_t = k] = 0 .$$

(3) For all $s,t \epsilon T$ $(s \leq t)$ and all $k \epsilon Q$

$$\Pr[Z_s \not\geq k, Z_t \geq k] = 0 .$$

(4) For all $s,t \epsilon T$ $(s \leq t)$ and all $k \epsilon Q$

$$\Pr[Z_s \geq k, Z_t \geq k] = \Pr[Z_t \geq k] .$$

Proof:

(1) implies (2):

Suppose that Z is nonrecoverable. By (3.3), for all $s,t \epsilon T$ $(s \leq t)$ and all $i,j \epsilon Q$,

$$i \not\geq j \text{ implies } \Pr[Z_s = i, Z_t = j] = 0 .$$

Since Q is denumerable, we then have, for all $k \epsilon Q$,

$$\Pr[Z_s \not\geq k, Z_t = k]$$

$$= \sum_{i \not\geq k} \Pr[Z_s = i, Z_t = k] = 0 .$$

(2) implies (3):

Suppose that, for all $s,t \in T$ ($s \leq t$) and all $j \in Q$,

$$\Pr[Z_s \not\geq j,\ Z_t = j] = 0 .$$

Then, since Q is denumerable,

$$\Pr[Z_s \not\geq k,\ Z_t \geq k]$$

$$= \sum_{j \geq k} \Pr[Z_s \not\geq k,\ Z_t = j]$$

$$\leq \sum_{j \geq k} \Pr[Z_s \not\geq j,\ Z_t = j] = 0 .$$

(3) implies (4):

Note first that, for all $s,t \in T$ ($s \leq t$) and all $k \in Q$,

$$\Pr[Z_t \geq k] = \Pr[Z_s \geq k,\ Z_t \geq k] + \Pr[Z_s \not\geq k,\ Z_t \geq k]. \qquad (3.5)$$

Hence, for all $s,t \in T$ ($s \leq t$) and all $k \in Q$,

$$\Pr[Z_s \not\geq k,\ Z_t \geq k] = 0$$

if and only if

$$\Pr[Z_s \geq k,\ Z_t \geq k] = \Pr[Z_t \geq k] .$$

Thus (3) and (4) are equivalent and, in particular, (3) implies (4).

(4) implies (1):

From (3.5), when condition (4) is satisfied, we have, for all $s,t \in T$ ($s \leq t$) and all $j \in Q$,

$$\Pr[Z_s \not\geq j,\ Z_t \geq j] = 0.$$

Thus, $i \not\geq j$ implies

$$\Pr[Z_s = i,\ Z_t = j] \leq \Pr[Z_s \not\geq j,\ Z_t \geq j] = 0 ,$$

in particular, it implies $\Pr[Z_s = i,\ Z_t = j] = 0$, i.e., as characterized in (3.3), Z is nonrecoverable.

This circle of implications thus completes the proof of Theorem 3.1.

An alternative way to characterize the recoverability of an operational model is by examining the state behavior of the model over the entire utilization period. In this regard, let us restrict our attention to operational models that are *separable* in the sense as defined in [17], i.e., there exists a denumerable subset R of T and an event Λ of probability 0 such that,

for any closed interval B and any open interval I in $(-\infty,+\infty)$, we have

$$\bigcap_{s\epsilon IR}\{Z_s\epsilon B\} - \bigcap_{s\epsilon IT}\{Z_s\epsilon B\} \subseteq \Lambda \tag{3.6}$$

where $IR = I\bigcap R$ and $IT = I\bigcap T$. The set R is referred to as a *separability set*.

When Z is separable, we are able to show that Z is nonrecoverable if and only if its state behavior over any time interval can be summarized by observing the state of Z at the end of the interval.

Theorem 3.2:

Suppose Z is a separable operational model. Then Z is nonrecoverable if and only if,

$$\begin{aligned} &\text{for all } r,t\epsilon T\ (r<t) \text{ and all } k\epsilon Q \\ &Pr[Z_s\geq k,\ r\leq s\leq t] = Pr[Z_t\geq k]\ . \end{aligned} \tag{3.7}$$

Proof:

Suppose Z is nonrecoverable and, given a state $k\epsilon Q$, let

$$E_s = \{\omega | Z_s(\omega)\geq k\} \quad (s\epsilon T).$$

Then, in terms of this notation, (3.7) says

$$\Pr[\bigcap_{s\in[r,t]} E_s] = \Pr[E_t] \,. \qquad (3.7)'$$

Furthermore, let us denote the intersection of two sets A and B by AB. Since Z is separable, there exists a denumerable subset R of T and a null event Λ such that, for all $r,t\in T$,

$$\bigcap_{s\in(r,t)R} E_s - \bigcap_{s\in(r,t)} E_s \subseteq \Lambda \,.$$

Accordingly,

$$E_r E_t \left[\bigcap_{s\in(r,t)R} E_s - \bigcap_{s\in(r,t)} E_s \right] \subseteq \Lambda$$

and, hence,

$$\bigcap_{s\in[r,t]R'} E_s - \bigcap_{s\in[r,t]} E_s \subseteq \Lambda \qquad (3.8)$$

where $R'=R\bigcup\{r,t\}$.

Clearly, the first set in (3.8) is measurable because $[r,t]R'$ is denumerable. Thus, under the separability hypothesis, the second set (which is contained in the first) is likewise measurable and has the same probability (assuming the probability measure Pr is complete; see Chapter 2, p. 16). More

precisely, for all $r,t \in T$ $(r<t)$,

$$\Pr[\bigcap_{s\in[r,t]} E_s] = \Pr[\bigcap_{s\in[r,t]R'} E_s] \,. \tag{3.9}$$

Hence, if we can show that the probability on the right side of (3.9) equals $\Pr[E_t]$, we establish the desired result, i.e., (3.7)'.

If we denote

$$D = \bigcap_{s\in[r,t]R'} E_s \,,$$

then, since $t \in [r,t]R'$, we have $D = DE_t$. Thus it suffices to show that

$$\Pr[DE_t] = \Pr[E_t]$$

or equivalently,

$$\Pr[\bar{D}E_t] = 0. \tag{3.10}$$

Next, note that

$$\bar{D}E_t = \left(\bigcup_{s\epsilon[r,t]R'} \bar{E}_s \right) E_t$$

$$= \bigcup_{s\epsilon[r,t]R'} \bar{E}_s E_t \, .$$

Accordingly, we have

$$\Pr[\bar{D}E_t] \leq \sum_{s\epsilon[r,t]R'} \Pr[\bar{E}_s E_t]$$

or equivalently (in our original notation),

$$\Pr[\bar{D}E_t] \leq \sum_{s\epsilon[r,t]R'} \Pr[Z_s \geq k, Z_t \geq k] \, .$$

Since Z is nonrecoverable, by Theorem 3.1 (condition 4), each term on the right side of the above equation is zero whence

$$\Pr[\bar{D}E_t] = 0 \, .$$

This proves (3.10) and thus establishes the necessity of (3.7).

To prove that (3.7) is sufficient, suppose (3.7) holds. Then, if we let $s,t\epsilon T$ and let $k\epsilon Q$, by Theorem 3.1, it suffices to show that

$$\Pr[Z_s \geq k, Z_t \geq k] = \Pr[Z_t \geq k]$$

or, equivalently, using the notations introduced above

$$\Pr[E_s E_t] = \Pr[E_t] \ . \tag{3.11}$$

The above equality is trivially true when $s=t$, so let us suppose $s<t$. By (3.7), it follows that

$$\Pr[\bigcap_{u\in[s,t]} E_u] = \Pr[E_t] \ .$$

Then, since

$$\Pr[E_s E_t] \leq \Pr[E_t]$$

and

$$\Pr[E_s E_t] \geq \Pr[\bigcap_{u\in[s,t]} E_u] = \Pr[E_t] \ ,$$

it follows that

$$\Pr[E_s E_t] = \Pr[E_t]$$

which establishes the sufficiency of (3.7).

Using Theorem 3.2 and the fact that Q is denumerable, recoverability can also be characterized by each of the following alternative conditions (the proofs are immediate and are omitted):

(1) For all $r,t \epsilon T$ $(r<t)$ and all $k \epsilon Q$

$$\Pr[Z_s>k,\ r \leq s \leq t] = \Pr[Z_t>k] \ . \tag{3.12}$$

(2) For all $r,t \epsilon T$ $(r<t)$ and $k \epsilon Q$

$$\Pr[Z_s \geq k,\ r \leq s < t,\ Z_t=k] = \Pr[Z_t=k] \ . \tag{3.13}$$

Theorem 3.2 provides us with a convenient way for relating the concept of recoverability to the traditional notion of repairability. To see this, let us define the *level-k reliability* of Z at time t to be

$$R_k(t) = \Pr[Z_s \geq k,\ 0 \leq s \leq t] \tag{3.14}$$

and define the *level-k availability* of Z at time t to be

$$A_k(t) = \Pr[Z_t \geq k] \ . \tag{3.15}$$

Clearly, when $Q=\{0,1\}$ where 0=failure and 1=success, $R_1(t)$ and $A_1(t)$ reduce to the usual notions of system reliability and system availability, respectively. Moreover, when Z is nonrecoverable, Theorem 3.2 implies that,

$$\text{for all } t\epsilon T \text{ and all } k\epsilon Q$$
$$A_k(t) = R_k(t). \tag{3.16}$$

In other words, when Z is nonrecoverable, its level-k reliability is reduced to the level-k availability for all $k\epsilon Q$. The significance of this observation is that, when Z is nonrecoverable, the calculation of the level-k reliability at a time t amounts to calculating the probability that the system operates at a level greater than or equal to k at that particular moment in time. On the other hand, let $T=[0,h]$ and suppose (3.16) holds. Then, since, for all $r,t\epsilon T$ $(r<t)$ and all $k\epsilon Q$,

$$\Pr[Z_s \geq k,\ 0 \leq s \leq t] \leq \Pr[Z_s \geq k,\ r \leq s \leq t] \leq \Pr[Z_t \geq k]\ ,$$

we have $A_k(t)=R_k(t)$ implies

$$\Pr[Z_s \geq k,\ r \leq s \leq t] = \Pr[Z_t \geq k]\ ,$$

i.e., condition (3.16) is necessary and sufficient for nonrecoverability. Thus, by taking the negation of the above condition, we also have the following alternative characterization of recoverability:

Theorem 3.3:

Let Z be a separable operational model with a state space Q. Then Z

is recoverable if and only if, there exist $t \epsilon T$ and $k \epsilon Q$, such that

$$A_k(t) > R_k(t). \tag{3.17}$$

Roughly speaking, the above theorem says that Z is recoverable if and only if, for some $k \epsilon Q$, the level-k availability is subject to improvement by maintenance actions.

Theorems 3.2 and 3.3 not only provide us with a useful tool for characterizing the behavior of operational models, they also provide us with a basis for evaluating the performability of degradable computing systems. Each of equations (3.14) and (3.15) defines an important class of performance measures that are proper generalizations of the traditional notions of system reliability and system availability. When the operational model is nonrecoverable, both classes convey the same information and the behavior of the system can be determined by taking a "snapshot" at the end of the utilization period. Motivated by the above properties of an operational model, we consider in the following section a single user-oriented performance variable that integrates these notions of system reliability and availability.

3.3 Evaluation of Computing Systems Using Functionals of a Markov Process

When describing system behavior in user-oriented terms, it is often possible to identify various operational "modes" for the system (including a failure mode) which result in different degrees of user satisfaction. Moreover,

for a given mode of operation, the extent of user satisfaction can often be quantified as a real number "rate" at which that operation benefits or penalizes the user. Depending on the application, these rates can have a variety of interpretations relating to the system's productivity, responsiveness, etc., or at a higher level, to such things as economic benefit (e.g., the worth rate measured, say, in dollars/unit time) associated with a given mode of operation.

Under the above conditions, a user-oriented model can be constructed in a natural way. As in the previous discussions, let S denote the total system in question and suppose that we have already determined a base model X and a capability function γ relative to some specified performance variable Y. Suppose further that the base model process X is defined relative to a continuous time interval T (the utilization period), that is,

$$X = \{X_t | t \epsilon T\} \tag{3.18}$$

where the random variables X_t take values in a denumerable state space Q (see (2.1) for the definition of a base model). Finally, we presume that at the base level, the system model is Markovian with a time-invariant structure, that is, X is a continuous-time time-homogeneous Markov process. Unless otherwise specified, it will be assumed that Q is countably infinite throughout the following discussions.

Within this framework, let us now consider the situation discussed

above where, at a higher level, one is able to identify various operational modes for S, each having an associated operational rate. If, further, each state of the base model can be classified according to some mode of operation, then there is a naturally defined real-valued function

$$f:Q \rightarrow R \tag{3.19}$$

where, for each $i \epsilon Q$, f(i) is the operational rate associated with the mode containing i. Moreover, if we let $\bar{Q}$ denote the range of f (i.e., $\bar{Q}=\{f(i)|i \epsilon Q\}$) and, for each variable X_t of X (see (3.18)), we let

$$Z_t = f(X_t) \, . \tag{3.20}$$

It follows that

$$Z = \{Z_t | t \epsilon T\} \tag{3.21}$$

is a stochastic process with state space $\bar{Q}$ referred to generally as a *functional* of the underlying Markov process X (see [22], for example).

When f is not 1-1 (i.e., some different states have the same mode of operation), the derived process Z will typically represent a simpler, higher level view of the system and is generally non-Markovian unless certain stringent conditions are satisfied. (Conditions under which the derived processes become

Markovian are discussed in the Appendix.) To qualify Z as an intermediate model, we must also require that Z be compatible with the performance variable Y to the extent that the probability distribution function of Y can be determined from Z. More precisely, letting κ denote the translation of trajectories of X to trajectories of Z (i.e., $\kappa(u)=\bar{u}$ where $\bar{u}(t)=f(u(t))$, for all $t\epsilon T$), there must exist a capability function $\bar{\gamma}$ for Z such that

$$\bar{\gamma}\cdot\kappa = \gamma \tag{3.22}$$

where $\cdot$ denotes functional composition, first applying κ. Although the above condition appears somewhat formidable, it says simply that the higher level model Z must remain detailed enough to permit solution of the system's performability. This condition can be typically satisfied in practice if the definition of performance (i.e., Y) is taken into account when identifying the various modes of operation and assigning rates to these modes.

If f, as defined in (3.19), satisfies condition (3.22) then we refer to f as an *operational structure* of S and, since states inherit the rates assigned to modes, the value f(i) is referred to as the *operational rate of i* or, when context permits, simply the "rate of i." Likewise, the corresponding functional Z is referred to as an *operational model of S* or, alternatively, a *model of S at the operational level*.

In reliability modeling where, at the operational level, a system is

typically viewed as either operating or not operating, the concept of an operational structure reduces to the familiar notion of a structure function [18]. Technically, a function $f:Q \rightarrow R$ is a structure function if Q has binary coordinates, i.e., $Q=\{0,1\}^m$, and f(i) is 1 or 0 according as S is operating or not operating in state i. More recently, operational structures have been employed at least implicitly in the context of performance-reliability modeling where the operational rates are referred to as computational capacities [9], [12]. Although capacity (which typically refers to the maximum rate at which a computer can "supply" computations) is a legitimate interpretation of operational rate, it should be emphasized that, in general, such rates can represent an interaction of supply (by the computer) and demand (from the environment); this is because that, as generally conceived, a state i of the base model represents a particular status of both the computer and its environment; hence, both supply and demand can be accounted for when translating i, via f, to its corresponding operational rate f(i).

In various special forms, then, the concept of an operational structure is no stranger to performance and reliability modeling. On the other hand, the general nature of associated functional Z, how it relates to the base model, how it can be exploited in solution procedures, etc., appear to be subjects that deserve further investigation.

In the following discussions, we focus our investigation on the evaluation of performability with respect to a generally defined performance

variable. This variable is defined in terms of an arbitrary operational model which is generally non-Markovian. However, by relating this variable to the underlying Markov process, it is shown that system performability can still be evaluated using traditional Markov process methods. The performance variable is motivated by the level-q reliability

$$R_q(t)=Pr[f(X_s)\geq q,\ 0\leq s\leq t]$$

(where $q\epsilon\bar{Q}$) discussed in Section 3.2.

Recall that, by (3.16), the operational model $Z=\{Z_t|t\epsilon T\}$ is nonrecoverable if and only if, for all $t\epsilon T$, Z_t is the "worst case" rate experienced by Z during [0,t]. On the other hand, if Z is recoverable, it was shown in Theorem 3.3 that the operational rate at the end of the utilization (i.e., the value Z_t) will generally not convey the worst case rate. Motivated by the above considerations, a performance variable Y_t, indicating the worst case operational rate during [0,t], can be defined on Z as follows:

$$Y_t = \min\{Z_s|0\leq s\leq t\}\ . \qquad (3.23)$$

As defined above, we note first that Y_t is a discrete performance variable since the base model X has a denumerable number of states and, hence, there are a denumerable number of operational rates. Therefore the

performability $perf_S$ of S (see (2.3)) is simply the probability distribution of Y_t, i.e.,

$$perf_S(q) = Pr[Y_t=q] \; . \tag{3.24}$$

Before attempting to solve the performability of S, let us consider the recoverability of Z in more detail. Since the underlying base model is a time-homogeneous Markov process, significant insights can be obtained regarding the relationship between Z and X by expressing the recoverability of Z in terms of the probabilistic nature of X.

In this regard, let us restrict our attention to Markov processes X which are regular in the sense that their transition probabilities are uniquely determined by a generator matrix or, equivalently, a state-transition-diagram (see [23], for example). Moreover, borrowing the terminology from [22], we say that i *leads* to j (where $i,j \epsilon Q$) and write $i \rightarrow j$ if and only if there exists a $t>0$ in T such that

$$Pr[X_t=j|X_0=i] > 0 \; .$$

Then, it can be shown that

Theorem 3.4:

Let Z be an operational model associated with the base model X and the operational structure f. Furthermore, let X be a time-homogeneous Markov process. Then, Z is recoverable if and only if, for some $s \epsilon T$ and some $i,j \epsilon Q$, both of the following conditions are satisfied

$$\begin{aligned} &(1) \quad \Pr[X_s=i] > 0 \\ &(2) \quad i \rightarrow j \text{ and } f(i) < f(j) . \end{aligned} \tag{3.25}$$

Proof:

By (3.4), Z is recoverable if and only if there exist $s,t \epsilon T$ ($s \leq t$) and $q,r \epsilon \bar{Q}$, such that

$$i < j \text{ and } \Pr[Z_s=q, Z_t=r] > 0 . \tag{3.26}$$

(Here, since $\bar{Q}$ is a totally ordered set, we are able to replace $\not\geq$ with $<$ in (3.4).) Now, since $\bar{Q}$ is denumerable,

$$\Pr[Z_s=q, Z_t=r]$$

$$= \sum_{\substack{f(i)=q \\ f(j)=r}} \Pr[X_s=i, X_t=j] ,$$

and, hence, equation (3.26) holds if and only if, for some $i,j \epsilon Q$,

$$f(i)=q,\ f(j)=r \text{ and } \Pr[X_s=i,\ X_t=j] > 0\ . \tag{3.27}$$

Moreover, since

$$\begin{gathered}\Pr[X_s=i,\ X_t=j] > 0 \text{ if and only if} \\ \Pr[X_s=i] > 0 \text{ and } i \rightarrow j\ ,\end{gathered} \tag{3.28}$$

it follows that the conditions stated in (3.25) are necessary and sufficient for Z to be recoverable.

By taking the negation of (3.25), similar result can also be derived to characterize the nonrecoverability of Z as follows:

Corollary:

Let Z be an operational model associated with the base model X and the operational structure f. Moreover, let X be a time-homogeneous Markov process. Then, Z is nonrecoverable if and only if, for all $i,j \epsilon Q$ and all $s \epsilon T$, at least one of the following conditions is satisfied:

$$\begin{aligned}&(1)\quad \Pr[X_s=i] = 0 \\ &(2)\quad i \rightarrow j \text{ implies } f(i) > f(j)\ .\end{aligned} \tag{3.29}$$

Note that if we assume Q is the *minimal state space* of X in the sense as defined in [22], i.e., for all $i \epsilon Q$, there exists an s in T such that $\Pr[X_s=i]>0$,

then the first conditions in (3.25) and (3.29) can both be eliminated. To show this, we first observe that

$$\Pr[X_s=i] > 0 \text{ if and only if}$$
$$\Pr[X_0=k] > 0 \text{ and } \Pr[X_s=i|X_0=k] > 0$$

for some $k\epsilon Q$. Now since X is regular, the transition probability $\Pr[X_t=i|X_0=k]>0$ as a function of t vanishes either everywhere or nowhere in T (see [23], p. 240). Thus, it must be the case that $\Pr[X_t=i|X_0=k]>0$ for all $t\epsilon T$. Clearly, it then follows that

$$\Pr[X_t=i] \geq \Pr[X_0=k]\cdot\Pr[X_t=i|X_0=k] > 0$$

for all $t\epsilon T$.

The recoverability of Z can also be characterized in terms of partitions induced by f on the state space Q. Note first that the binary relation $\rightarrow$ induces an equivalence relation on Q as follows (see [22], for example): we say i *communicates* with j (denoted $i \leftrightarrow j$) if and only if $i \rightarrow j$ and $j \rightarrow i$. Let $[i]_c$ be the communicating class containing i. Then the partition of Q induced by the communication relation can be denoted as a set

$$\pi_c = \{[i]_c | i\epsilon Q\} . \qquad (3.30)$$

Note also that the operational structure f:Q → R induces another partition on Q

$$\pi_f = \{[i]_f | i \epsilon Q\} \tag{3.31}$$

such that i,j belong to the same equivalence class if f(i)=f(j). In terms of the above partitions and assuming that X is a time-homogeneous Markov process with minimal state space Q, we can then show that

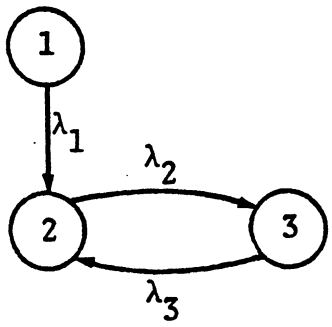
Lemma:

If Z is nonrecoverable, then π_c is finer than π_f (denoted $\pi_c \leq \pi_f$).

Proof:

Suppose that i and j belong to the same block in π_c. It must be the case that $i \leftrightarrow j$, which in turn, implies that $f(i) \geq f(j)$ and $f(j) \geq f(i)$ because Z is nonrecoverable. Thus, it follows that f(i)=f(j), i.e., i and j belong to the same block in π_f.

The converse of the above lemma is generally not true. For example, let X be a Markov process with transition graph



and initial distribution $Pr[X_0=1]=1$. Suppose the operational structure $f:Q \rightarrow R$ is given by $f(1)=0$ and $f(2)=f(3)=1$. Then

$$\pi_c = \{\{1\},\{2,3\}\} = \pi_f .$$

However, Z is recoverable, because $1 \rightarrow 2$ but $f(1)<f(2)$.

The above example also suggests a necessary and sufficient condition for Z to be a nonrecoverable model. Under the same assumptions as those for the above lemma, we first define a partial ordering of the set π_c: For all $[i]_c$ and $[j]_c$ in π_c, let

$$[i]_c \rightarrow [j]_c \quad \text{if} \quad i \rightarrow j . \tag{3.32}$$

Clearly, the partial ordering as a relation is reflexive, transitive and antisymmetric. Furthermore, let us define a mapping

$$h:\pi_c \rightarrow \bar{Q} \tag{3.33}$$

such that $h([i]_c)=f(i)$. Then,

Theorem 3.5:

Z is nonrecoverable if and only if h is well-defined and order-preserving.

Proof:

Suppose Z is nonrecoverable. Then, by the above lemma, $\pi_c \leq \pi_f$. In other words, for all i and j in Q, $i \leftrightarrow j$ implies $f(i)=f(j)$ and, hence, $h([i]_c)=f(i)$ is well-defined. Moreover, suppose $[i]_c \rightarrow [j]_c$. Then, by (3.30), we have $i \rightarrow j$ and, hence,

$$h([i]_c) = f(i) \geq f(j) = h([j]_c) ,$$

i.e., h is order-preserving.

Conversely, let us suppose that h is well-defined and order-preserving. Then, for all $i,j \in Q$,

$$i \rightarrow j \quad \text{implies} \quad [i]_c \rightarrow [j]_c .$$

Now since h is well-defined, we have

$$h([i]_c) = f(i) \text{ and } h([j]_c) = f(j) .$$

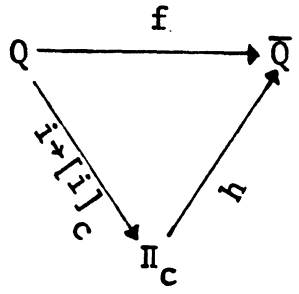
Applying the order-preserving assumption of h, it then follows that

$$f(i) = h([i]_c) \geq h([j]_c) = f(j)$$

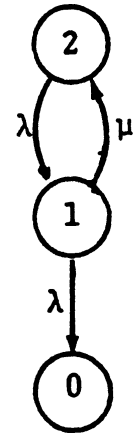
i.e., Z is nonrecoverable.

Corollary:

If Z is nonrecoverable, then the following diagram commutes



Theorem 3.5 permits us to determine whether an operational model is nonrecoverable by comparing the state diagram of the underlying Markov process with the operational structure. To illustrate, let us consider a base model X with the following state diagram



and initial distribution $Pr[X_0=2]=1$. Suppose the operational structure is given by $f(2)=f(1)=1$ and $f(0)=0$. Then, $\pi_c=\{\{0\},\{1,2\}\}=\pi_f$, $\{ 1,2 \} \rightarrow \{0\}$ and $h(\{1,2\})=1 > h(\{0\})=0$. Thus, by Theorem 3.5, Z is nonrecoverable.

It should also be noted that a Markov process X may induce a nonrecoverable model with respect to an operational structure but a recoverable model with respect to other operational structures. For example, using the same base model as above and the operational structure $g(2)=2$, $g(1)=1$ and $g(0)=0$, then it is clear that $\pi_c \not\leq \pi_g$ (π_c is not finer than π_g). Hence, by the lemma of Theorem 3.5, Z induced by g is a recoverable model.

Returning now to the problem of evaluating the performability of S (with respect to the performance variable Y_t as defined by (3.23)), we consider the problem in two cases based on the recoverability of the operational model Z. If the operational model Z is nonrecoverable, Theorem 3.2 shows that the behavior of Z during [0,t] can be determined by the state of Z at the time instant t. In particular, we have

$$Pr[Z_s \geq q,\ 0 \leq s \leq t] = Pr[Z_t \geq q] \tag{3.34}$$

and

$$Pr[Z_s > q,\ 0 \leq s \leq t] = Pr[Z_t > q]\ . \tag{3.35}$$

Hence the performability of S (see (3.24)) can be obtained by evaluating the

finite dimensional distribution $Pr[Z_t=q]$. More precisely, we have

$$\begin{aligned} perf_S(q) &= Pr[Y_t=q] \\ &= Pr[\min\{Z_s | 0\le s\le t\}=q] \\ &= Pr[Z_s\ge q,\ 0\le s\le t] - Pr[Z_s>q,\ 0\le s\le t] \\ &= Pr[Z_t\ge q] - Pr[Z_t>q] \\ &= Pr[Z_t=q] \ . \end{aligned} \tag{3.36}$$

Thus, in this case, evaluating performability (i.e., to determine the probability distribution function of Y_t) is tantamount to evaluating the transition function of X, i.e.,

$$\begin{aligned} perf_S(q) &= \sum_{f(j)=q} Pr[X_t=j] \\ &= \sum_{\substack{f(j)=q \\ i,j<Q}} Pr[X_t=j|X_0=i]\cdot Pr[X_0=i] \ . \end{aligned} \tag{3.37}$$

On the other hand, if Z is recoverable, more elaborate solution methods are required since (3.36) is no longer satisfied.

When the operational model is recoverable, the performability $perf_S$ of S can be obtained by calculating the conditional probabilities

$$m_{ij}^{q}(t) = Pr[Y_t=q, X_t=j|X_0=i] \tag{3.38}$$

where $i,j\epsilon Q$, $q\epsilon\bar{Q}$ and $t\geq 0$. Note that, when where $t=0$, the minimum operational rate is just the operational rate associated with the initial state; in short

$$m_{ij}^{q}(0) = \begin{cases} 1 & \text{if } i=j \text{ and } f(i)=q, \\ 0 & \text{otherwise.} \end{cases} \tag{3.39}$$

Then, by summing over some of the indices of $m_{ij}^{q}(t)$, we have

$$perf_S(q) = \sum_{i,j\epsilon Q} m_{ij}^{q}(t)\cdot p_i \tag{3.40}$$

where $p_i=Pr[X_0=i]$.

There are several ways of expressing the conditional probabilities $m_{ij}^{q}(t)$ in terms of the state transition probabilities of the underlying Markovian base model X [24]. First, let us introduce another stochastic process based on X and the performance variable Y_t as

$$\bar{X} = \{(X_t,Y_t)|t\epsilon T\} . \tag{3.41}$$

Then, since

$$Y_t = \min[\inf_{s<r\leq t}\{Z_r, Y_s\}]$$

and since X is a Markov process, it can be shown that $\bar{X}$ is a Markov process.

Clearly, the generator matrix of $\bar{X}$ can be expressed in terms of the state transition rates of the underlying Markov process X. For all $i,j\epsilon Q$ $(i\neq j)$, let λ_{ij} denote the transition rate of X from state i to state j. Then the generator matrix of X is the $|Q|\times|Q|$ matrix

$$A = [a_{ij}]$$

where, for all $i,j\epsilon Q$,

$$a_{ij} = \begin{cases} \lambda_{ij} & \text{if } i\neq j, \\ -\sum_{k\neq i} \lambda_{ik} & \text{if } i=j. \end{cases} \tag{3.42}$$

If, further, we let the generator matrix of $\bar{X}$ be denoted by

$$\bar{A} = [\bar{a}_{yz}]$$

where $y=(k,r)$ and $z=(j,q)$ belong to $Q\times\bar{Q}$, then

$$\bar{a}_{yz} = \begin{cases} a_{kj} & \text{if 1) } f(k)\geq r,\ f(j)\geq q \text{ and } r=q, \text{ or} \\ & \quad 2)\ f(k)\geq r,\ f(j)=q \text{ and } r>q, \\ 0 & \text{otherwise.} \end{cases} \tag{3.43}$$

Accordingly, if we denote the transition function of $\bar{X}$ by

$$\bar{p}_{xz}(t) = \Pr[\bar{X}_t=z|\bar{X}_0=x]$$

where $x,z\epsilon Q\times\bar{Q}$, then the transition functions of $\bar{X}$ can be expressed as the system of differential equations (see [23]; pp. 254-255, Theorem 4.5)

$$\frac{d}{dt}\bar{p}_{xz}(t) = \sum_{y\epsilon Q\times\bar{Q}} \bar{p}_{xy}(t)\cdot\bar{a}_{yz}\ . \tag{3.44}$$

Furthermore, notice that when $x=(i,f(i))$ and $z=(j,q)$,

$$\begin{aligned} \bar{p}_{xz} &= \Pr[\bar{X}_t=(j,q)|\bar{X}_0=(i,f(i))] \\ &= \Pr[X_t=j,\ Y_t=q|X_0=i,\ Y_0=f(i)] \\ &= m_{ij}^{q}(t)\ . \end{aligned}$$

Hence, by varying the value of y over $Q\times\bar{Q}$ and replaying $\bar{a}_{yz}$ by (3.43), the above system of differential equations (3.44) can be expressed more

conveniently as

$$\frac{d}{dt} m_{ij}^{q}(t) = \begin{cases} \sum_{f(k)\geq q} m_{ik}^{q}(t)\cdot a_{kj} & \text{if } f(i)\geq q,\ f(j)>q \\ \sum_{\substack{r\geq q \\ f(k)\geq r}} m_{ik}^{q}(t)\cdot a_{kj} & \text{if } f(i)\geq q,\ f(j)=q, \\ 0 & \text{otherwise.} \end{cases} \tag{3.45}$$

It was suggested [24] that the above equations can also be obtained by relating (3.38) to the notion of taboo probabilities [22].

Another approach for determining the probability distribution of the random variable Y_t is to modify the underlying Markov process X by making some of the states in Q absorbing [24]. More precisely, let us rename and rearrange the elements in $\bar{Q}$ into an increasing sequence $\bar{Q}=\{1,2,...\}$. For each $q\epsilon\bar{Q}$, let B_q be a subset of Q such that

$$B_q = \{i | f(i)<q\}$$

where $f:Q \rightarrow R$ is the operational structure of Z. We then replace the state space of X by a reduced one in which a single state b_q replaces the states B_q and denote the transformed process by

$$X^q = \{X_t^q | t \epsilon T\} .$$

Moreover, let us denote the corresponding generator matrix of X^q by

$$A^q = [\hat{a}_{ij}] .$$

then, for all $i,j \epsilon \{k | f(k) \geq q\} \bigcup \{b_q\}$,

$$\hat{a}_{ij} = \begin{cases} a_{ij} & \text{if } f(i) \geq q,\ f(j) \geq q, \\ \sum_{\ell \epsilon B_q} a_{i\ell} & \text{if } i \neq b_q,\ j = b_q, \\ 0 & \text{if } i = b_q. \end{cases} \tag{3.46}$$

If, further, we let the transition functions of X^q be denoted by

$$p_{ij}^q(t) = \Pr[X_t^q = j | X_0^q = i] \tag{3.47}$$

where $i,j \epsilon \{k | f(k) \geq q\} \bigcup \{b_q\}$. Then the conditional probabilities $m_{ij}^q(t)$ can be expressed as follows: For all $i,j \epsilon Q$ and all $q \epsilon \bar{Q}$, where $f(i) \geq q$ and $f(j) \geq q$,

$$m_{ij}^{q}(t) = Pr[Y_t \geq q, X_t = j | X_0 = i] - Pr[Y_t \geq q+1, X_t = j | X_0 = i]$$

$$= Pr[Z_s \geq q, 0 \leq s \leq t, X_t = j | X_0 = i]$$

$$- Pr[Z_s \geq q+1, 0 \leq s \leq t, X_t = j | X_0 = i]$$

$$= Pr[X_t^q = j | X_0 = i] - Pr[X_t^{q+1} = j | X_0 = i]$$

$$= p_{ij}^{q}(t) - p_{ij}^{q+1}(t) \; . \tag{3.48}$$

Hence, if we solve the transition probabilities $p_{ij}^{q}(t)$ for all $q \epsilon \bar{Q}$, the performability of S can be computed by

$$perf_S(q) = \sum_{\substack{f(i) \geq q \\ f(j) \geq q}} p_{ij}^{q}(t) \cdot p_i - \sum_{\substack{f(i) \geq q+1 \\ f(j) \geq q+1}} p_{ij}^{q+1}(t) \cdot p_i$$

$$= \sum_{f(i) \geq q} [1 - p_{ib_q}^{q}(t)] \cdot p_i - \sum_{f(i) \geq q+1} [1 - p_{ib_{q+1}}^{q+1}(t)] \cdot p_i \tag{3.49}$$

where $p_i = Pr[X_0 = i]$ are the initial probabilities of X.

When the total system is modeled by a recoverable operational model, either (3.45) or (3.49) can be used to evaluate the performability of the system. The solution method described in (3.45) requires evaluating a large system of differential equations. On the other hand, the solution method described in (3.49) decomposes the system of differential equations of (3.45) into smaller subsystems. Thus, evaluating (3.49) amounts to a step-by-step

iterative solution to (3.45). Finally, we also note that, in addition to the applications illustrated in the following example and the next chapter, (3.45) and (3.49) can also be used to compute the "intraphase transition probabilities" considered in the context of phased models (see Section 5.4).

3.4 Performability of a Triplicated Fault-Tolerant Computing System

To illustrate the solution methods for the evaluation of system performabilities, let us consider a degradable fault-tolerant computing system wherein resources are triplicated and voted (triple modular redundance). This type of resource redundancy has been employed in a variety of hardware architectures (see [25] for an example of current usage) and, in a more general form (N modular redundancy), has been investigated as a means for implementing fault-tolerant software (see [26], for example). Although such a system will generally possess a number of triplicated resources (e.g., the various "triads" of the FTMP architecture [25]), let us restrict our attention to a single resource, say, a triplicated processor consisting of three identical processor modules and a voter. With respect to hardware faults, we assume that the processor modules fail independently and that each fails permanently with a constant failure rate λ (failures/hr.). The system's ability to recover from a hardware fault in a processing module is accounted for by a coverage parameter c (see [27]).

When the system is free of hardware faults, we further assume that it has some capability of recovering from errors due to design faults in the software. Such errors are presumed to occur at a constant rate σ (errors/hr.). Transitions from the error state, with or without recovery, occur at a constant rate μ (transitions/hr.) which we assume to be much larger than the processor module failure rate λ. The probability of software error recovery, given a

software error, is a constant d (the software error coverage parameter); error recovery thus occurs at a rate $d\mu$. Lack of recovery from a software error is presumed to cause a crash (system failure). If a processor module becomes faulty and the fault is tolerated via successful voting, the input-output behavior of the system remains the same. With this loss of a processing resource, however, we assume that the system is no longer capable of software error recovery. Hence any further errors, due to a software fault or a second faulty processor module, result in failure of the system.

Under the above assumptions, the system can be conveniently represented by a 4-state Markovian base model, where the states are interpreted as follows:

State	Processor Fault	Software Error
1	No	No
2	Yes	No
3	No	Yes
4	System failure	

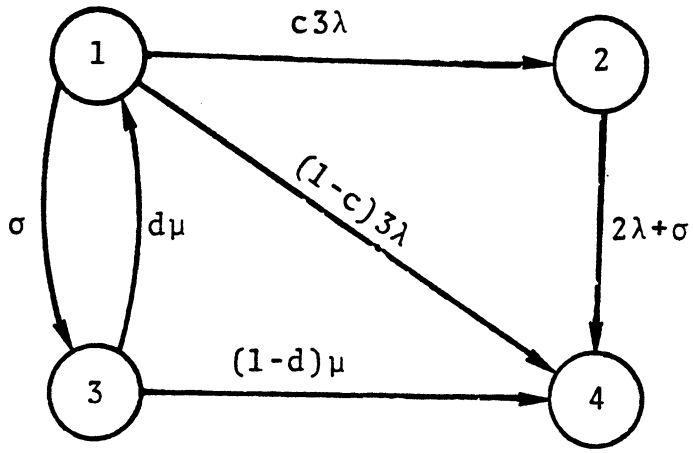


Figure 3.1

Markov Model of a TMR System

with Software Error Recovery

The state-transition-rate diagram of the model is depicted in Figure 3.1. Note that when the system is attempting recovery from a software error (state 2), there are no transitions representing the occurrence of a hardware fault. This is a consequence of our assumption that $\mu \gg \lambda$, in which case such occurrences are negligible.

As for performance, let us suppose the user is interested in three levels of accomplishment: full performance (as would be exhibited by a fault-free version of the system), degraded performance (at least one software error during utilization but successful recovery in each case), and system failure. To obtain an appropriate operational model that can support this view of performance, we see that states 1 and 2 can be identified with one mode of operation while states 3 and 4 must be distinguished. Moreover, because the mode {1,2} is preferred over {3} and {3} is preferred over {4}, we find that the following function will suffice as an operational structure:

i	f(i)
1	1
2	1
3	1/2
4	0

To establish that the performance variable in question can indeed be supported by the functional Z of X (Fig. 3.1) induced by this choice of f, suppose that Y_t is formulated as in (3.23), i.e., Y_t is the minimum operational rate experienced during the utilization period [0,t]. Then it is easily verified that Y_t conveys the desired information, i.e.,

Value of Y_t	Interpretation
1	Full performance
1/2	Degraded performance
0	Failure

Note also that the operational model Z is recoverable in the sense defined in (3.4) due to the error/recovery cycle from rate 1 to 1/2 and then back to rate 1. The performability of the system can thus be evaluated using either of the methods discussed in Section 3.3.

To illustrate the solution method described in (3.49), note that the generator matrix of X is

$$A = \begin{bmatrix} -(3\lambda+\sigma) & c3\lambda & \sigma & (1-c)3\lambda \\ 0 & -(2\lambda+\sigma) & 0 & 2\lambda+\sigma \\ d\mu & 0 & -\mu & (1-d)\mu \\ 0 & 0 & 0 & 0 \end{bmatrix} .$$

Thus, with respect to each operational rate $q=1$, $1/2$ or 0, the generator matrices of the corresponding transformed Markov processes are given by (3.46), viz.,

$$A^1 = \begin{bmatrix} -(3\lambda+\sigma) & c3\lambda & \sigma+(1-c)3\lambda \\ 0 & -(2\lambda+\sigma) & 2\lambda+\sigma \\ 0 & 0 & 0 \end{bmatrix}$$

and

$$A^{1/2} = A^0 = A\ .$$

Hence, if we denote the transition functions of each transformed process by a matrix

$$P^q(t) = [p_{ij}^q(t)] \qquad (i,j\epsilon\{k|f(k)\geq q\}\bigcup\{b_q\})$$

where $p_{ij}^q(t)$ is defined by (3.47), then $P^q(t)$ is determined by A^q uniquely by the well known formula

$$P^q(t) = e^{A^q t}\ .$$

In particular, we have

$$P^1(t) = \begin{bmatrix} e^{-(3\lambda+\sigma)t} & 3c[e^{-(2\lambda+\sigma)t} - e^{-(3\lambda+\sigma)t}] & c_1 \\ 0 & e^{-(2\lambda+\sigma)t} & c_2 \\ 0 & 0 & 1 \end{bmatrix}$$

where

$$c_1 = 1 - e^{-(3\lambda+\sigma)t} - 3c[e^{-(2\lambda+\sigma)t} - e^{-(3\lambda+\sigma)t}]$$

$$c_2 = 1 - e^{-(2\lambda+\sigma)t},$$

and

$$P^{1/2}(t) = \begin{bmatrix} d_1 & d_2 & d_3 & 1-(d_1+d_2+d_3) \\ 0 & e^{-(2\lambda+\sigma)t} & 0 & 1-e^{-(2\lambda+\sigma)t} \\ d_4 & d_5 & d_6 & 1-(d_4+d_5+d_6) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where if we let

$$z = \sqrt{9\lambda^2 + \mu^2 + \sigma^2 + 6\lambda\sigma - 6\lambda\mu - 2\sigma\mu + 4d\mu\sigma}$$

$$x = \frac{3\lambda + \mu + \sigma + z}{2}$$

$$y = \frac{3\lambda + \mu + \sigma - z}{2}$$

then

$$d_1 = \frac{3\lambda - \mu + \sigma - z}{2z} e^{-yt} + \frac{3\lambda - \mu + \sigma + z}{2z} e^{-xt}$$

$$d_2 = \frac{3\lambda c(y-\mu)}{z(y-2\lambda-\sigma)} e^{-yt} - \frac{3\lambda c(x-\mu)}{z(x-2\lambda-\sigma)} e^{-xt}$$

$$+ \frac{3\lambda c(\mu-2\lambda-\sigma)}{(x-2\lambda-\sigma)(y-2\lambda-\sigma)} e^{-(2\lambda+\sigma)t}$$

$$d_3 = \frac{\sigma}{z} e^{-yt} - \frac{\sigma}{z} e^{-xt}$$

$$d_4 = \frac{d\mu}{z} e^{-yt} - \frac{d\mu}{z} e^{-xt}$$

$$d_5 = \frac{6\lambda cd\mu}{(\lambda+\sigma-\mu+z)z} e^{-yt} - \frac{6\lambda cd\mu}{(\lambda+\sigma-\mu-z)z} e^{-xt}$$

$$+ \frac{12\lambda cd\mu}{(\lambda+\sigma-\mu+z)(\lambda+\sigma-\mu-z)} e^{-(2\lambda+\sigma)t}$$

$$d_6 = \frac{3\lambda - \mu + \sigma + z}{2z} e^{-yt} - \frac{3\lambda - \mu + \sigma - z}{2z} e^{-xt} \quad .$$

Accordingly, applying (3.40), the performability of S can be expressed as

$$perf_S(1) = p_1 \cdot (1-c_1) + p_2 \cdot (1-c_2)$$

$$perf_S(1/2) = p_1 \cdot (d_1+d_2+d_3) + p_2 \cdot e^{-(2\lambda+\sigma)t}$$

$$+ p_3 \cdot (d_4+d_5+d_6) - perf_S(1)$$

$$perf_S(0) = 1 - perf_S(1/2) - perf_S(1) \qquad (3.50)$$

where $p_i = Pr[X_0 = i]$.

By expressing the performability of the system in terms of the above closed form solution (3.49), various design tradeoffs can then be investigated by varying the parameter values. To illustrate, let us fix the following base model parameters to be

$$\lambda = 5 \times 10^{-4}$$

$$c = .99999$$

$$\mu = 10^3$$

$$d = .9$$

and assume that the system initially has all three modules operational, i.e.,

$$p = [1\ 0\ 0\ 0].$$

Then, depending on the choice of the software failure rate σ, the

performability of the system exhibits various kinds of relationships between different levels of accomplishment. In particular, Figure 3.2 and 3.3 display the performability of S as a function of t (the duration of the utilization) for $\sigma=10^{-2}$ and $\sigma=10^{-3}$, respectively.

In both figures, the performability of S is represented by three curves: I, II and III. Curve I is the probability of fault-free operation throughout the utilization. The probability decreases from 1 to 0 as t goes to infinity. Curve II is the probability that the system suffered from performance degradation due to software faults while remaining operational throughout the utilization period. Finally, we note that curve III is the familiar S-shape function for system unreliability.

When we compare Figure 3.2 with Figure 3.3, we find that substantial performance improvement is obtained by reducing the number of design faults in the software. For example, consider the case when the duration of the utilization is 400 hours. With the reduction in software failure rate by a factor of 10, the probability of degraded performance and the probability of system failure are reduced, respectively, to 1/2 and 1/3 of their original values; more significantly, the probability of full performance is increased by a factor of 37.

Although the above example has established the feasibility of the hierarchical modeling approach for the evaluation of system performabilities, the operational model constructed in the example is based on a specific state

transition diagram (Figure 3.1). Thus, to extend the generality of the model, we consider in the next chapter a systematic approach for describing the underlying Markov process together with a general method for determining the operational structures of degradable computing systems. Moreover, we also note that the results developed in this chapter are further extended in Chapter 5 to permit the modeling and evaluation of systems with a time-varying environment.

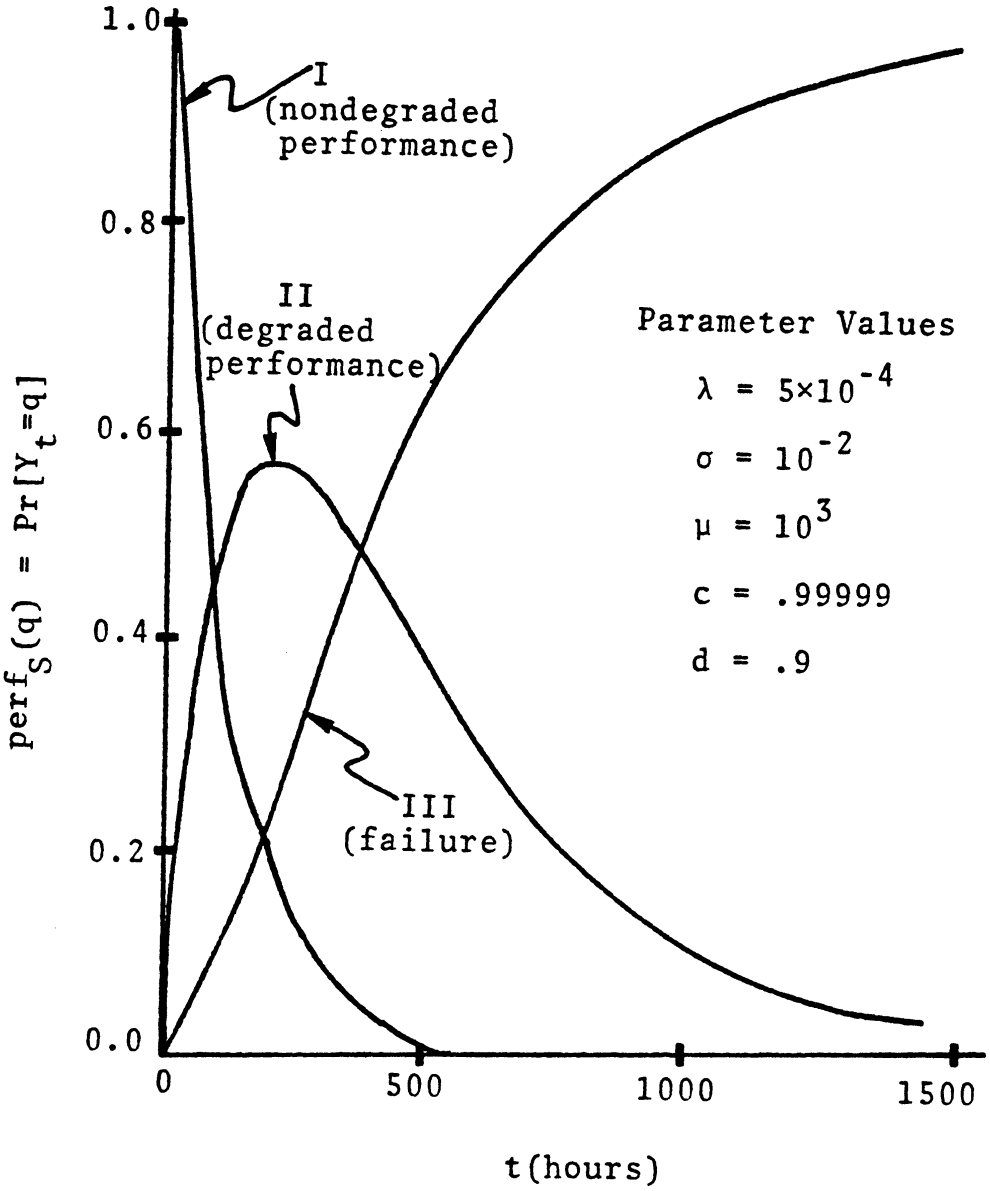


Figure 3.2

Performability of S

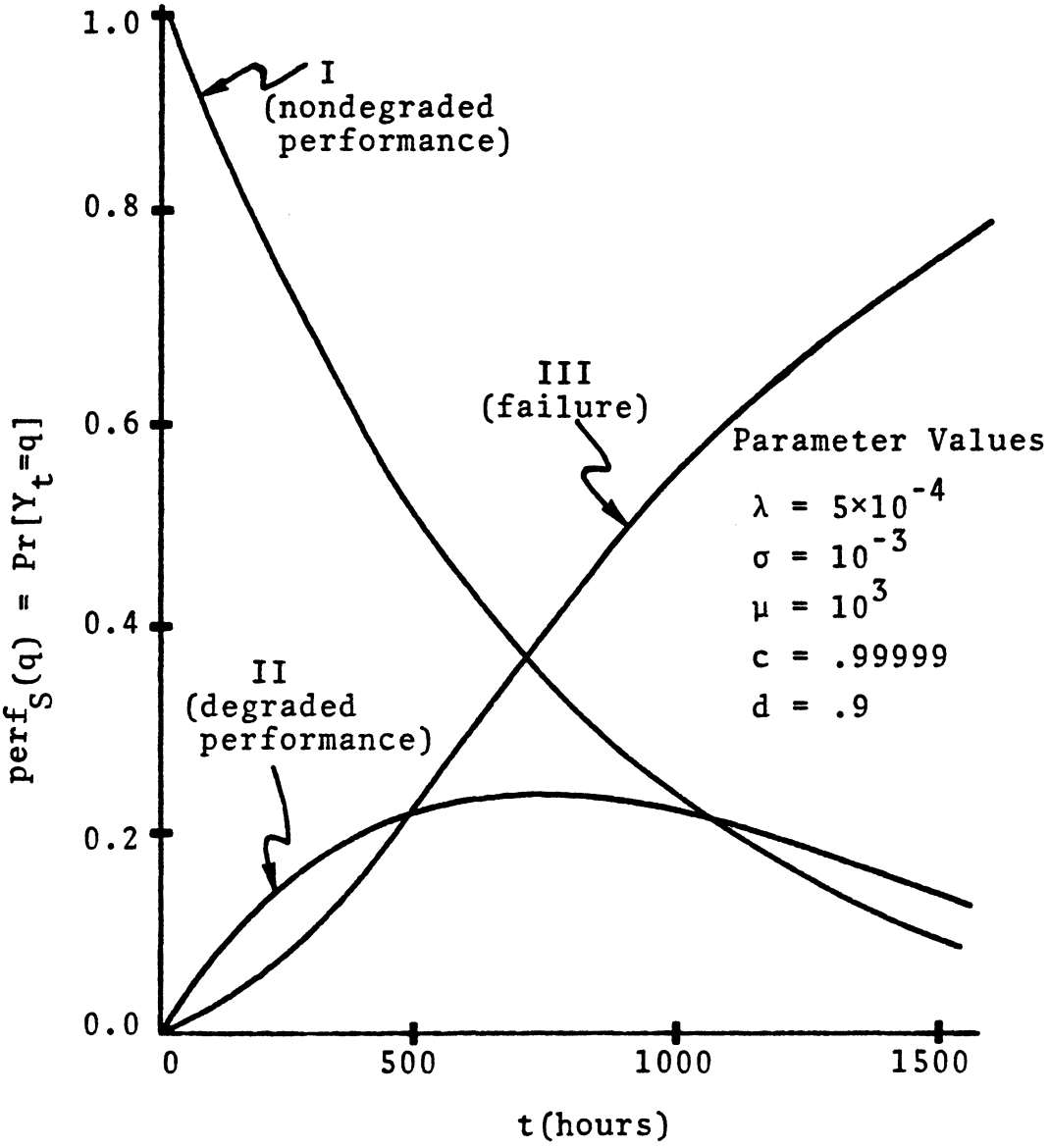


Figure 3.3

Performability of S

CHAPTER 4

MODELING AND EVALUATION OF DEGRADABLE MULTIPROCESSOR SYSTEMS

4.1 Introduction

The design of a distributed multiprocessor system (see [28]-[30], for example) is generally approached in a sequential manner beginning with the identification of the computing system's application. The problem identification phase is followed by a functional breakdown of the application into major subtasks to be performed by the system. Following these phases, the designer then specifies the performance and reliability requirements in terms of the resource requirements for each task, the time relationship between tasks, the executive software overhead for system control, etc. Finally, based on the performance and reliability requirements of the system, alternative hardware and software architectures are then considered to optimize cost, performance and other trade-off criteria.

Moreover, in the design of multiprocessor systems for real-time control applications, tasks to be performed are often partitioned into several priority groups and priority interrupt mechanisms are used to meet the stringent constraints of fast response time (see [29] and [30], for example). Normally,

all tasks are executed iteratively to generate sample-time updates of control variables. However, when the computer's resources decrease due to faults to a point that only some of the tasks can be completed in time, tasks from the higher priority groups are given preferential treatment over tasks from the lower priority groups. Computing systems capable of performance degradation as above are often referred to as *gracefully degradable computing systems* or simply *degradable computing systems*.

Performance degradation of degradable multiprocessor systems are typically realized through the following steps [31]:

1) Error detection: Basic techniques for error detection include error detecting/correcting codes, time-out counter, memory protection, majority voting, periodic testing, etc.

2) Fault location and hardware reconfiguration: Once an error is detected, diagnostic programs and testing strategies are used to localize the faulty components. The hardware reconfiguration program is then called upon to establish an operational configuration.

3) Computation recovery: Computation recovery concerns the restoration of a valid system state from which the system can resume its operation. The restoration of a valid system state can be achieved by rollback and retry, uses of traces, program roll-aheads, etc.

4) Software reconfiguration: Fault-tolerant software permits the replacement of suspected software modules with their alternative versions at

run time. Current approaches include N-version programming [26] and the use of recovery blocks [32].

There are many different designs of degradable multiprocessor systems that are potential candidates for real-time control applications. In general, these designs can be characterized in terms of the degree of redundancies built into their basic components (e.g., simplex, duplex and triple-modular-redundancy). The characterization is useful because each class of systems can be attributed with certain specific performance and reliability trade-offs.

Replicated components are often used in a degradable multiprocessor systems to enhance the system reliability. For example, when triple-modular-redundancy (TMR) is used, not only all single faults can be tolerated, but procedures for error detection, fault-location and system reconfiguration are also simplified considerably. However, since there is no parallel-processing within TMR, the configuration represents a substantial loss of computing power. Systems using triplicated components in their design include C.vmp [33], SIFT [30] and FTMP [25].

When the application of a degradable multiprocessor system requires not so much reliability (i.e. uninterrupted operation throughout the utilization period) as the ability to recover from failures, simplex components are often used to improve the performance/cost ratio of the system (e.g., PRIME [34] and PLURIBUS [29]). Since such systems rely mostly on complicated self-

testing logic for error detection and location, the reliability of these systems is generally lower than that of systems using replicated components. This is because single faults may cause such systems to crash and the detection latency [35] (the time period between the first error and the first detected error) of these systems are generally longer.

In this chapter, a comprehensive model for degradable multiprocessor systems is presented for studying the trade-offs between systems with different degrees of component redundancy. The model is based on the approach considered in the previous chapter; a Markovian base model is described to represent the physical resources of the system and priority queuing models are used to determine the operational structure associated with the base model. Since our model supports the evaluation of system performability, it differs substantially from those considered by Borgerson [36] and Losq [37] which stress hardware-oriented measures such as reliability or availability.

4.2 System Model

As compared with existing Markov models for degradable multiprocessor systems (see [36] and [37], for example), the model presented here has the advantages that (i) the partitioning of the system is based on the system's available resources as well as computational requirements of the user's application, (ii) the hierarchical representation (i.e., the base model together with the operational structure) permits the formulation and evaluation of user-oriented performance variables.

4.2.1 Base Model for System Resources

Degradable multiprocessor systems typically can be divided into several physical resources each of which is made up of one or more identical components. These resources generally form a pool shared by tasks to be performed by the system. For example, as described in [25], the FTMP computer includes 15 processors, 9 memory units, 5 busses and 48 bus guardian units to be shared by aircraft functional tasks. Generally, the amount of resources that a system can provide varies from time to time depending on the intrinsic hardware failure rates, the effectiveness of fault tolerance mechanisms and the repair procedures. Hence, if we assume that (i) the occurrences of failures and repairs are independent among different components, (ii) each component has constant failure and repair rates, and (iii) fault tolerance mechanisms are "memoryless" in the sense that they are determined by the current state of the system, then the resource availability of the system can be represented as a Markov process.

Before attempting to describe a general Markov model for system resources, let us consider first the effects of failures on the system resources. As suggested in [37], two classes of hardware faults can be distinguished according to the characteristics of fault tolerance mechanisms. The first class corresponds to hardware faults that are detected and recovered from as soon as they occur. Hardware faults of the first class are referred to as *safe faults* because failed components are removed instantaneously from the resource pool

to avoid data contamination or faulty control. The second class of hardware faults, referred to as *unsafe faults*, contains those that are either latent faults or those that are in the process of fault recovery. Depending on the degree of component redundancy, failure to tolerate unsafe faults may cause the system to crash because of the loss or the corruption of important information.

Since the performance of a system is determined by the amount of fault-free resources, the state of the resource model can be defined to be the number of safe faults and unsafe faults. More precisely, suppose that the multiprocessor system contains N resources where the i^{th} resource contains n_i identical components. Suppose further each component of the i^{th} resource has failure rate λ_i and repair rate μ_i. If we assume that the occurrence of more than one event such as failure or repair completion has negligible probability, then the system can be represented as a Markov process

$$X = \{X_t | t \epsilon T\} \tag{4.1}$$

where, for each $t \epsilon T$, X_t is random variable taking values in the state set

$$Q = \{(a_1,b_1,a_2,b_2,...,a_N,b_N) | 0 \leq a_i + b_i \leq n_i \text{ for all } 1 \leq i \leq N\} .$$

For each state $(a_1,b_1,a_2,b_2,...,a_N,b_N)$ in Q, a_i denotes the number of safe faults of the i^{th} resource and b_i denotes the number of unsafe faults of the i^{th}

resource.

Four types of state transitions can be distinguished: repair, safe fault, unsafe fault and fault recovery (see Table 4.1). On the completion of a repair, the number of safe faults is decreased by one. Once a hardware fault has occurred, either the number of the safe faults or the number of unsafe faults is increased by one depending on whether the fault is safe or unsafe. Finally, the successful recovery of an unsafe fault will decrease the number of unsafe faults by one and increase the number of safe faults by one.

To derive the transition rates of the above state transitions, the system's ability to recover from a hardware fault is modeled by a single parameter c defined to be the conditional probability that a system will be able to recover once a hardware fault has occurred (referred to as the coverage of the system, see Section 3.6). It is further assumed that the probability of a failure being transient is represented by another parameter α. Based on the above assumptions about the failure and repair characteristics of the system, the transition rates of the above Markov process can be expressed as in Table 4.1 (where the parameters are summarized in Table 4.2).

Type of Transition	Next state	Transition Rate
repair	$(a_1,b_1,...a_i-1, b_i,...,a_N,b_N)$	$a_i \cdot \mu_i$
safe fault	$(a_1,b_1,...,a_i+1, b_i, \ldots, a_N,b_N)$	$(n_i-a_i-b_i)\cdot(1-\alpha)\cdot c\cdot\lambda_i$
unsafe fault	$(a_1,b_1, \ldots, a_i, b_i+1,...,a_N,b_N)$	$(n_i-a_i-b_i)\cdot(1-c)\cdot\lambda_i$
recovery	$(a_1,b_1, \ldots, a_i+1, b_i-1,...,a_N,b_N)$	$b_i \cdot \nu_i$

Present State = $(a_1,b_1,...,a_i,b_i,...,a_N,b_N)$

Table 4.1

State Transition Rates

of a General Resource Model

When the system contains only one type of resource (i.e., N=1), the model can be represented more conveniently using transition graphs as in Figure 4.1. This single resource model, as a special case to the above model, is the same as the one considered in [37] by Losq except that states are named differently here to simplify the formulation of operational rates.

Another special case of the above general model can be obtained when all unsafe faults result in system failure. In this case, since unsafe faults cause system resources to vanish, the state of the system can be taken to be the number of fault-free components at each moment in time. In other words, the state set Q defined in (4.1) can be represented as

$$Q = \{(a_1,a_2,...,a_N) | 0 \leq a_i \leq n_i \text{ for all } 1 \leq i \leq N\} \tag{4.2}$$

where, for each $(a_1,a_2,\ldots,a_N)$ in Q, a_i is the number of fault-free components of the i^{th} resource. The simplification on the corresponding transition rates is described in Table 4.3. Again, as a special case, the single resource model can be represented by a transition graph as in Figure 4.2.

Parameter	Interpretation
λ_i	Component failure rate of the i^{th} resource
μ_i	Component repair rate of the i^{th} resource
ν_i	Component recovery rate of the i^{th} resource following unsafe faults
α	Probability of a fault being transient
c	Probability of a fault being safe

Table 4.2

Base Model Parameters

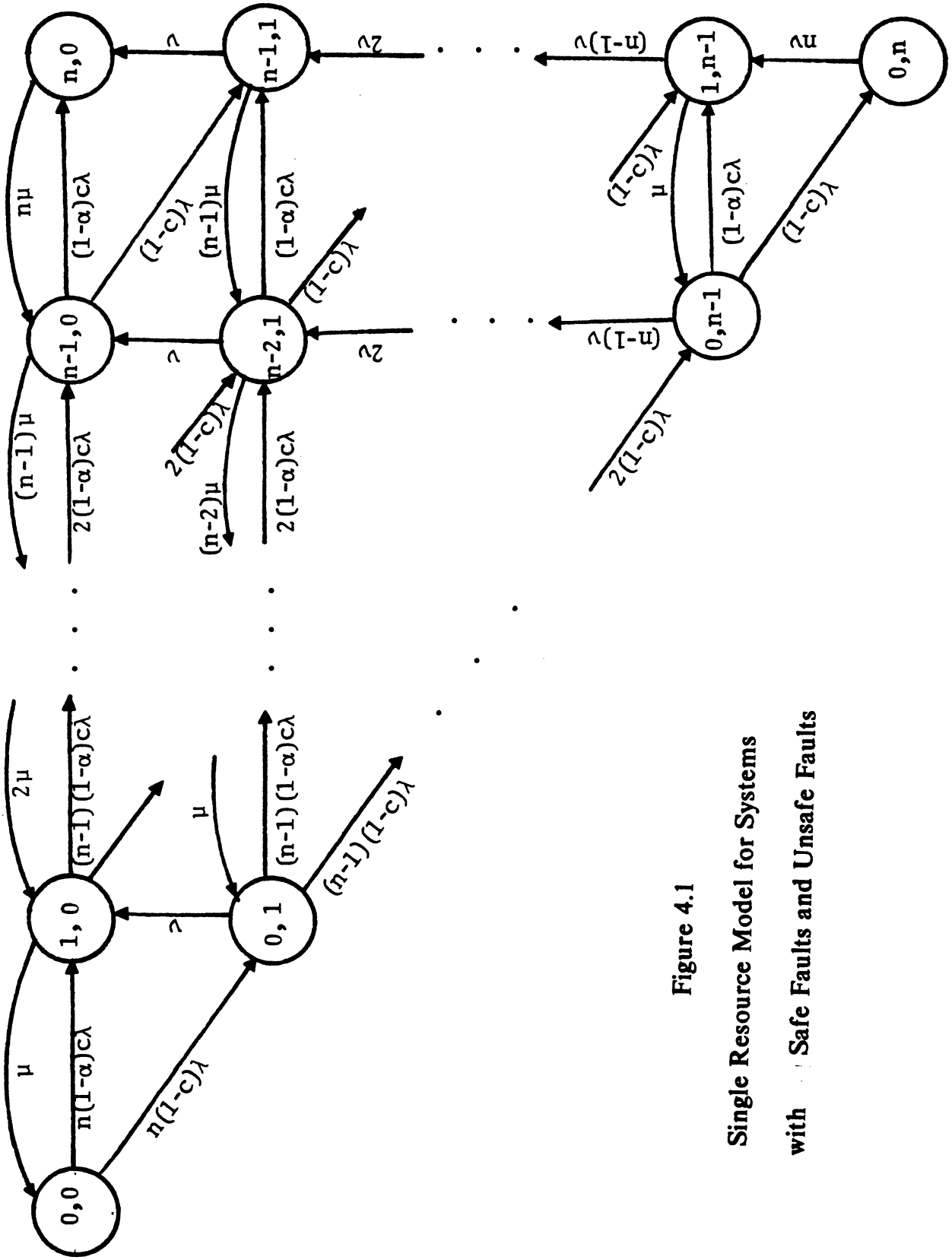


Figure 4.1

Single Resource Model for Systems with Safe Faults and Unsafe Faults

Type of Transition	Next state	Transition Rate
repair	$(a_1,...,a_{i-1},a_i+1,a_{i+1},...,a_N)$	$(n_i-a_i)\cdot\mu_i$
hardware fault	$(a_1,...,a_{i-1},a_i-1,a_{i+1},...,a_N)$	$a_i\cdot(1-\alpha)\cdot c\cdot\lambda_i$
recovery failure	$(a_1,\ldots,a_{i-1},0,a_{i+1},...,a_N)$	$a_i\cdot(1-c)\cdot\lambda_i$

Present State = $(a_1,...,a_i,\ldots,a_N)$

Table 4.3

Transition Rates for Systems with

Instantaneous Detections and Recoveries

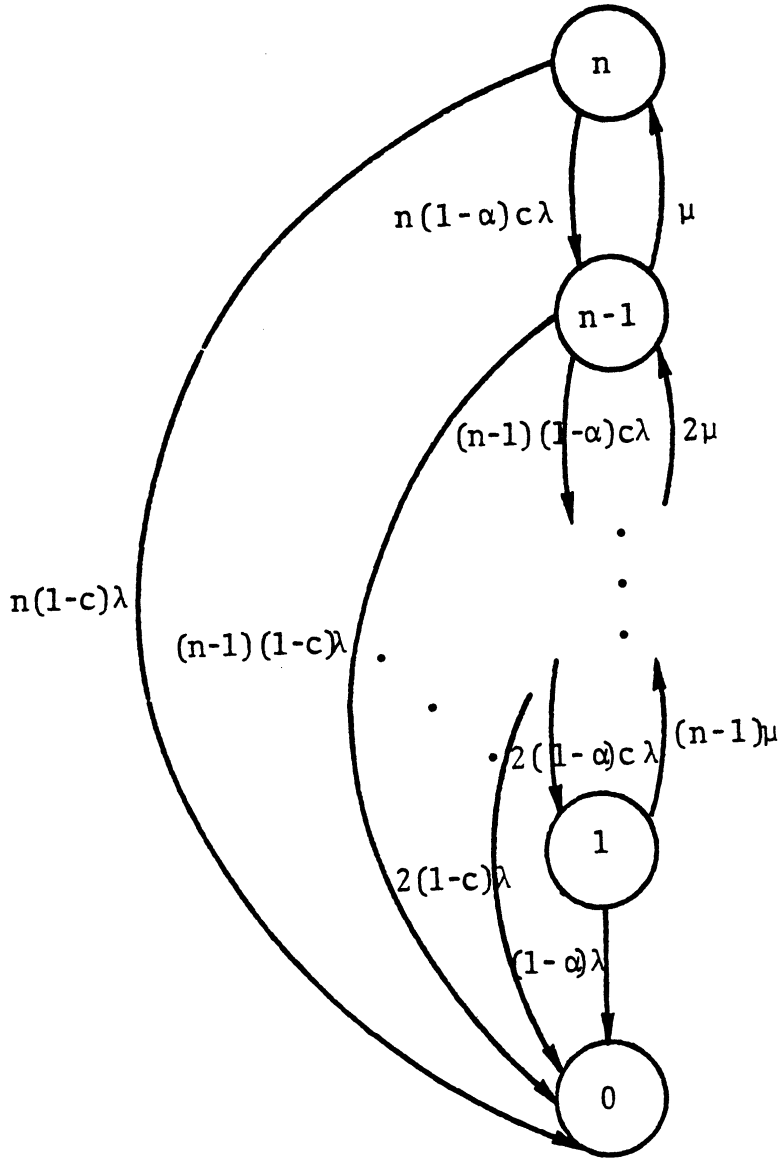


Figure 4.2

Single Resource Model for Systems with

Instantaneous Detections and Recoveries

4.2.2 Performance Variable

The performance and reliability requirements of a real-time system often differ from those of a general purpose computer because of the stringent constraints of fast response time. Thus, to characterize the performance and the reliability of a real-time system by a single performance variable, the performance variable must take into account both the resource availability as well as the promptness of the system response.

As an example, let us consider first a typical real-time environment encounted by a control computer. In a study concerning the design of fault-tolerant computers for aircraft, Ratner et al. [39] have identified 26 computational tasks most likely to be performed by the control computer of an advanced commercial aircraft. These computation tasks conceptually can be regarded as short programs stored in a common memory of a multiprocessor system and each task is scheduled to be executed periodically according to a predetermined frequency. However, the actual execution of a task may be delayed from its scheduled execution time because of the resource sharing, interface between tasks and the overhead for running system software. Since a prolonged starting-time delay may cause dangerous conditions to develop, the computational tasks are grouped into 5 priority classes and priority interrupt mechanisms are used to reduce the delay times of more critical tasks.

The above example clearly shows that the concept of the starting time delay is a useful tool for specifying the performance criteria of a real-time

control application: A task is regarded as failing to satisfy the real-time constraint if its starting-time delay, on a regular basis, exceeds a certain predetermined value. More precisely, let d be the estimated length of time that would have to elapse before an undesirable condition is noticed. Then the real-time constraints can typically be stated as: either the average starting-time delay or the percentile starting-time delay must be less than d. For example, the percentile starting-time delay can be stated as that the starting-time delay should not exceed d for 99% of the time when a task is scheduled to be executed. For simplicity, it will be assumed in the following discussions that the average starting-time delay is used to specify the real-time constraints.

To generalize the above notion of the starting-time delay, a user-oriented performance variable can be formulated as follows. First, let us assume that the tasks to be performed by the computer belong to one of a set of κ different priority classes. Then, relative to the utilization period $T=[0,t]$, the performance variable can be defined as a random variable Y taking value in the set $\{0,1,...,\kappa\}$ such that

$$Y = k \quad \text{iff}$$

k is the largest nonnegative integer less than or equal to κ such that all tasks from the first k priority groups are executed within the real-time constraints throughout T.

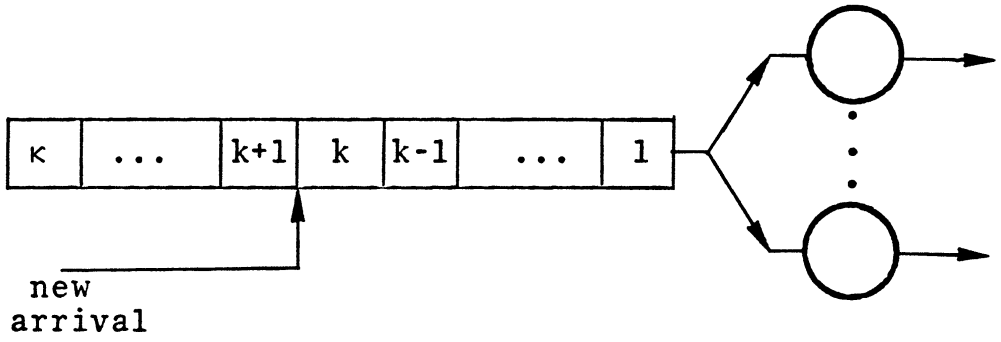
In other words, if we adopt the convention that the priority groups are numbered in reverse order (i.e., the smaller the number, the higher the priority), then the performance variable Y can be regarded as the degree of user satisfaction relative to how well the more critical tasks are executed by the computer to satisfy the real-time constraints. In particular, $Y = 0$ can be interpreted as a system crash, since, in this case, the computer does not even have enough resources to meet the demand of the most critical tasks (i.e., those from priority group 1). On the other hand, $Y=\kappa$ represents nondegraded performance, since all tasks are executed properly within their real-time constraints.

4.2.3 Operational Structure

To ease the evaluation of system performability, the connection between the performance variable Y and the base model X can be established more easily by introducing intermediate models to account for the internal structure of the computer. Under the assumptions described in Sections 4.2.1 and 4.2.2, a natural representation of the system's behavior at the operational

level is the behavior of the computer's control programs. Typically, to resolve conflicting demands on system resources, the control software of a multiprocessor system must provide a scheduler (also called a supervisor or executive) that allocates system resources to application tasks and handles interfaces between tasks. When the computer is degradable, the control software must also provide a mechanism (called a reconfiguration mechanism) which, upon detection of faults, appropriately changes the scheduler to facilitate error recoveries. In other words, given a general resource model as described in Figure 4.1, it is possible to associate each state of the model with a scheduler. Accordingly, depending on how well the application tasks are performed within each resource state according to a given scheduler, various operational rates can be identified to reflect different degrees of user satisfaction.

To measure the effectiveness of the scheduling algorithms associated with the resource states, it is assumed that each scheduler is modeled by a resource sharing priority queuing model (see [21], for example). For each resource state of a resource model, it is further assumed that arriving tasks form a single queue according to the "head-of-the-line" (HOL) queuing discipline (see Figure 4.3), that is, an arrival from priority class k joins the queue behind all "customers" from priority class k (and higher) and in front of all "customers" from priority class k+1 (and lower). Moreover, the value of



Head-of-the-Line

Priority Queue

one's priority remains constant in time. Thus, while the customers with the highest priorities are selected for service ahead of those with lower priorities, customers from the same priority class are served on a first-come, first-served (FCFS) basis. Finally, we also note that two possible refinements in priority mechanism can be distinguished depending on whether the execution of a low-priority task is interrupted when a task of higher priority arrives.

Since we are concerned with the ability of a system in satisfying the real-time constraints, the effectiveness of each scheduler can be measured in terms of the "expected waiting times" of the corresponding queuing model. More precisely, for each state q of a general resource model and for each priority class k, let τ_k^q be a random variable denoting the time spent waiting in the queue of a priority k "customer" with respect to the queuing model of resource state q (see [40], p. 189). Then the expected value of τ_k^q is a close approximation of the average starting-time delay when i) the communication delays between tasks are negligible, and ii) the transition rates of the given resource model are much lower than the arrival and the service rates of the computational tasks. The first condition can typically be satisfied by treating each task as an atomic unit for resource allocation. The second condition is usually satisfied automatically because failures and repairs of a computer occur much less frequently than the iteration rates of the computational tasks (thus the expected waiting times rapidly approach steady-states once the system enters a particular state).

The above relationships between the average starting-time delays and the expected waiting times provide us with a basis for partitioning the resource states into operational modes. First, we note that the ability of a resource state q in satisfying the real-time constraints of a priority k task can be expressed as

$$E[\tau_k^q] < d \tag{4.4}$$

where d is the predetermined time length as described in the previous subsection. The use of average starting-time delays to specify the real-time constraints is reasonable because, in general, the effect of a starting-time delay on the behavior of the system is proportional to the duration of the delay.

In addition to the real-time constraints, the partitioning of system states into operational modes must also take into account the effects of unsafe faults. For example, consider a multiprocessor system with triplicated components. The occurrence of a undetected double fault in a triad will cause the system to fail regardless of the duration of starting-time delays. Accordingly, if we assume that the probabilistic nature of the system resources is specified as a Markov process X with state space Q (see (4.1)), then an operational structure can be given as a function

$$f:Q \rightarrow \{0,1,\ldots,\kappa\}$$

where, for each $q \epsilon Q$,

$$f(q) = \begin{cases} k & \text{if unsafe faults in q are tolerated and,} \\ & \text{for some } 1 \leq k \leq \kappa,\ E[\tau_k^q] < d \text{ and } E[\tau_k^{q+1}] \geq d, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

Note that, in the above formulation, we have assumed that tasks of higher priorities have shorter expected waiting times, i.e.,

$$k < k' \text{ implies } E[\tau_k^q] \leq E[\tau_{k'}^q] .$$

The assumption is satisfied when the scheduling algorithm (associated with q) uses the usual priority queuing disciplines in resource allocation. Having established the operational structure, the performance variable described in Section 4.2.2 can now be expressed as

$$Y = \min\{f(X_t) | t \epsilon T\} .$$

Finally, to conclude the construction of the operational model, we note that the analysis of a priority queuing system is generally more difficult than that of a nonpriority system. In particular, for the multiple channel (multiple server) case, it is usually required to assume no service-time

distinctions between priorities or else the mathematics becomes intractable. However, as illustrated in the following section, even under the above stringent conditions, the scheduling algorithms of a single resource model can still be modeled satisfactorily using a priority M/M/m queue (see [40], pp. 193-194).

For multiple resource models, the scheduler associated with each resource state can be modeled by an open or a closed queuing network (also called network of queue; see [4] and [21], for example). In addition to the basic assumptions governing the arrival and service rates of each "service station" (see [4], pp. 161-163), it is also required to assume that the priority queuing discipline at each service station is work conserving in the sense that the priority interrupts will not impose extra work on the server. Solutions for the expected waiting times at each service station can be obtained by incorporating solutions for priority M/M/m queues in the multiple resource models.

4.3 Evaluation of Two Degradable Multiprocessor Systems

In this section, the performances of two degradable multiprocessor systems are evaluated and compared to determine the tradeoffs between two design approaches. Both of the computers to be evaluated are assumed to be multiprocessor systems containing 4 identical processors and a common memory module. They differ in the way that the system resources are allocated to perform computations. One computer (S_1) is assumed to operate in simplex mode, i.e., the processors are allowed to operate independently. The other

computer (S_2) is assumed to operate in duplex mode, i.e., the processors are paired into duplex subsystems to enhance the system reliability. While the simplex mode of operation generally can improve the performance/cost ratio of the system, the duplex mode of operation provides better detection and fault coverage relative to the simplex mode of operation.

With respect to hardware faults, we assume that the processor modules fail independently and permanently with a constant failure rate λ_2 (failures per hour). The memory module is assumed to have a constant failure rate λ_1 and fails independently with respect to other subsystems. We further assume that the system's ability to recover from a failure is accounted for by the coverage factors c_1 and c_2, respectively, for S_1 and S_2. Since simplex subsystems rely on error detecting codes and self-checking logic for error detection and location, the coverage of S_1 is generally lower than that of S_2. This is because, in the simplex mode of operation, undetected single faults may cause the system to crash and the detection latency (i.e., the time duration between the first error and the first detected error) of a simplex subsystem is generally longer than that of a replicated subsystem. Accordingly, it will be assumed that

$$c_1 \leq c_2. \tag{4.6}$$

Under the above assumptions, both S_1 and S_2 can be represented as

a Markov process according to the general resource model as described in (4.1). However, since neither the simplex mode of operation nor the duplex mode of operation can tolerate unsafe faults, the model can be reduced to the special case whose state space is given by (4.2). In particular, each of S_1 and S_2 can be conveniently represented by a 10-state Markovian base model X_i (i=1 or 2), where the state space Q_i (i=1 or 2) can be represented as

$$Q_i = \{ (k,j) \mid k=0,1 \text{ and } j=0,1,2,3,4 \}. \tag{4.7}$$

For each state (k,j) in Q_i, k denotes the operational status of the shared memory module (0=failed and 1=working) and j denotes the number of working processor units. Thus, for instance, a fault-free configuration is encoded as state (1,4). The state-transition diagram of the model is depicted in Figure 4.4.

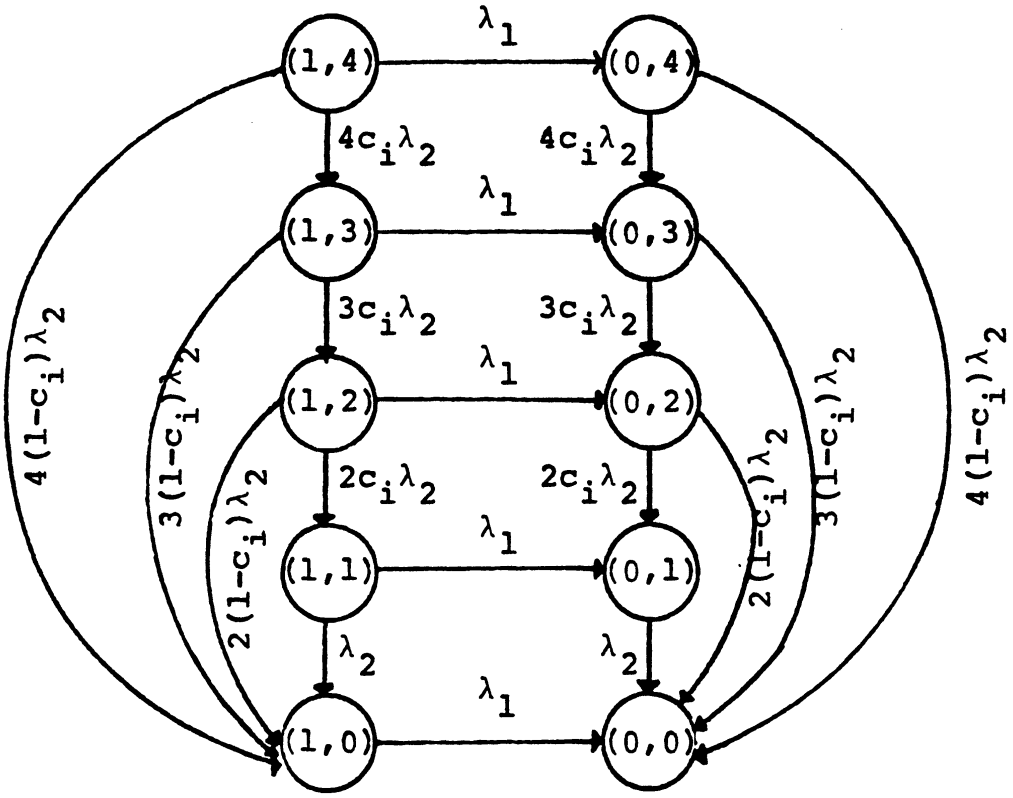


Figure 4.4

State-Transition Diagram

of the Base Model for S_1 and S_2

To provide a useful comparison between S_1 and S_2, their performance must be evaluated with respect to the same work environment. In this regard, let us consider a typical real-time control application where its control computer uses priority interrupt mechanisms to meet the stringent constraints of fast response time. For simplicity, let us assume that each computational task of the system is assigned a priority of 1 or 2 denoting, respectively, high or low-priority. Normally, all tasks are executed iteratively to generate sample-time updates of control variables. However, when the computer's resources decrease, because of failures, to a point that only some of the tasks can be completed in time, tasks from high-priority group are given preferential treatment over tasks from low-priority group. Based on the above assumptions of the work environment, a user-oriented performance variable can then be formulated as

$$Y_i = \begin{cases} 2 & \text{if the system is operating as prescribed,} \\ 1 & \text{if only high priority tasks meet the average response time requirement,} \\ 0 & \text{if high priority tasks can not meet the average response time requirement.} \end{cases} \tag{4.8}$$

In other words, the performance of S_i conveys the following information:

Value of Y_i	Interpretation
2	normal
1	degraded
0	failure

On closer examination of the relationship between Y_i and X_i, we find it is necessary to introduce an intermediate model between them because X_i is not detailed enough to support the user's view of system performance Y_i. One way to introduce such an intermediate model is to identify an operational model by taking into account the workload environment of the computer. In this regard, let us assume that tasks from priority group i (i=1 or 2) arrive in a Poisson stream at 1 task per millisecond. We also assume that each task, regardless of its priority, requires a service time exponentially distributed with mean service time 1/3 milliseconds. Furthermore, we assume that the HOL queuing discipline is used, but there is no preemption [5]. Then, if the behavior of the system in a state q is modeled as a queuing system with m servers, the average starting-time delay of tasks from priority group i can be approximated by the expected waiting time of the M/M/m priority queue.

As for the operational structure, the states of the base model X_i can now be partitioned into three operational modes according to their average

starting-time delay. In particular, suppose that the average starting-time delays of both priority groups shall not exceed d milliseconds. Then, the operational structure of S_i (i= 1 or 2) can be expressed as a function

$$f_i:Q_i \rightarrow \mathbf{R}$$

such that, for each $q \epsilon Q_i$,

$$f_i(q) = \begin{cases} 2 & \text{if } E[\tau_1^q] \leq d \text{ and } E[\tau_2^q] \leq d, \\ 1 & \text{if } E[\tau_1^q] \leq d \text{ and } E[\tau_2^q] > d, \\ 0 & \text{otherwise.} \end{cases} \tag{4.9}$$

If d = 1/2 millisecond, the operational structures of S_1 and S_2 can be tabulated as follow:

State q of S_i (i = 0 or 1)	Operational rate $f_1(q)$ of S_1	Operational rate $f_2(q)$ of S_2
(1,4)	2	2
(1,3)	2	2
(1,2)	2	1
(1,1)	1	1
(1,0)	0	0
(0,4)	0	0
(0,3)	0	0
(0,2)	0	0
(0,1)	0	0
(0,0)	0	0

The operational rate assignments in the table are determined by the degree of subsystem redundancies. For instance, suppose that the system is in state (1,2), i.e., the memory module is operational and two processors are working. Then, since in the duplex mode of operation both processors must perform the same function in parallel, the behavior of S_2 in state (1,2) can be modeled as a M/M/1 priority queue. Thus, using the existing formula for computing the expected queueing times (see [40], for example), the average starting-time delays can be approximated by

$$T_1^q = 1/3 \text{ milliseconds and } T_2^q = 1 \text{ milliseconds}$$

where q=(1,2). Since, in this case,

$$T_1^q \leq d \text{ and } T_2^q > d,$$

we have $f_2(q)=1$ by equation (4.9). On the other hand, since the behavior of S_1 in state (1,2) can be modeled as a M/M/2 priority queue, the average starting-time delays can be estimated as

$$T_1^q = 1/30 \text{ milliseconds and } T_2^q = 1/20 \text{ milliseconds}$$

where q=(1,2). Accordingly, it follows by (4.9) that $f_1(q)=2$.

Given the above performance variables Y_i and the operational structures f_i, it can easily be verified that, for each X_i (i= 1 or 2),

$$Y_i = \min \{ f_i(X_t) \mid t \epsilon T \}$$

where T is the utilization period. Thus the solution methods described in the previous chapter can now be used to solve the system performability (i.e., the probability distribution function of Y_i).

Suppose that the system is initially fault-free, i.e., $Pr[X_0=(1,4)]=1$. Then, relative to the utilization period $T=[0,t]$, the performability of S_1 is given by

$$\begin{aligned} perf_1(2) &= Pr[\, Y_1=2 \,] \\ &= e^{-(4\lambda_2+\lambda_1)t} + 4c_1[e^{-(3\lambda_2+\lambda_1)t} - e^{-(4\lambda_2+\lambda_1)t}] \\ &\quad + 6c_1{}^2[e^{-(2\lambda_2+\lambda_1)t} - 2e^{-(3\lambda_2+\lambda_1)t} + e^{-(4\lambda_2+\lambda_1)t}], \end{aligned}$$

$$\begin{aligned} perf_1(1) &= Pr[\, Y_1=1 \,] \\ &= 4c_1^3[e^{-(\lambda_2+\lambda_1)t} - 3e^{-(2\lambda_2+\lambda_1)t} \\ &\quad + 3e^{-(3\lambda_2+\lambda_1)t} - e^{-(4\lambda_2+\lambda_1)t}], \end{aligned}$$

and

$$perf_1(0) = Pr[\ Y_1=0\ ]$$

$$= 1 - perf_1(2) - perf_1(1).$$

Similarly, the performability of S_2 is given by

$$perf_2(2) = Pr[\ Y_2=2\ ]$$

$$= e^{-(4\lambda_2+\lambda_1)t} + 4c_2[e^{-(3\lambda_2+\lambda_1)t} - e^{-(4\lambda_2+\lambda_1)t}],$$

$$perf_2(1) = Pr[\ Y_2=1\ ]$$

$$= 4c_2{}^3e^{-(\lambda_2+\lambda_1)t} + 6(c_2{}^2-2c_2{}^3)e^{-(2\lambda_2+\lambda_1)t}$$

$$+ 12(c_2{}^3-c_2{}^2)e^{-(3\lambda_2+\lambda_1)t} + (6c_2{}^2-4c_2{}^3)e^{-(4\lambda_2+\lambda_1)t},$$

and

$$perf_2(0) = Pr[\ Y_2=0\ ]$$

$$= 1 - perf_2(2) - perf_2(1).$$

When expressed as functions of the duration of the utilization, the above equations can be represented, respectively, as in Figures 4.5 and 4.6 for the indicated parameter values of S_1 and S_2. When we compare Figure 4.5 with Figure 4.6, S_2 clearly results in better system reliability than S_1 in the sense that

$$perf_2(0) < perf_1(0).$$

Moreover, S_2 also has a much higher degraded performance even though its nondegraded performance is slightly lower than that of S_1. In general, we may conclude that, if the system's ability to recover from faults is low, then the performance of the system can be made to degrade more gracefully by allowing the subsystems to operates in duplex mode.

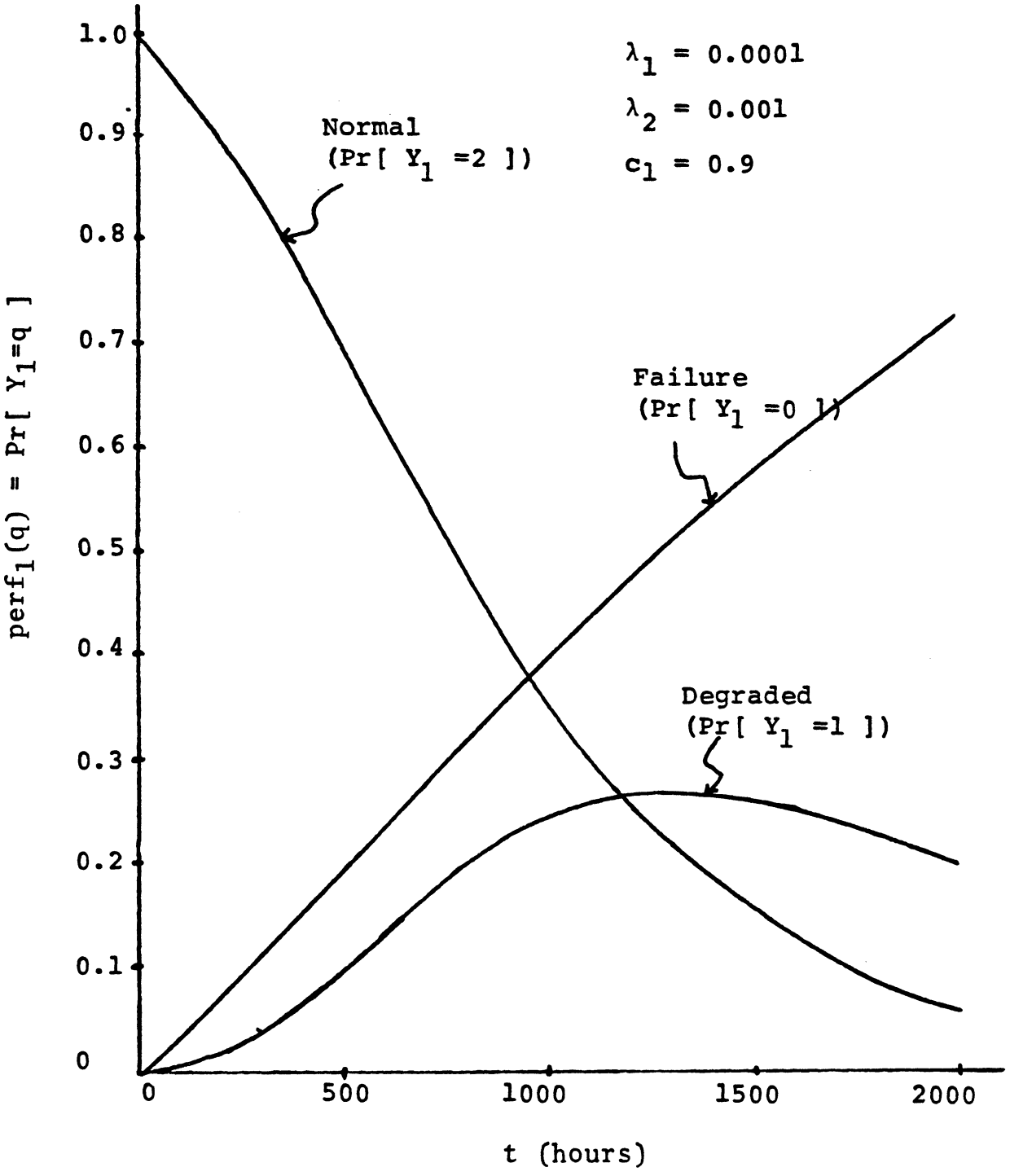


Figure 4.5

Performability of S_1

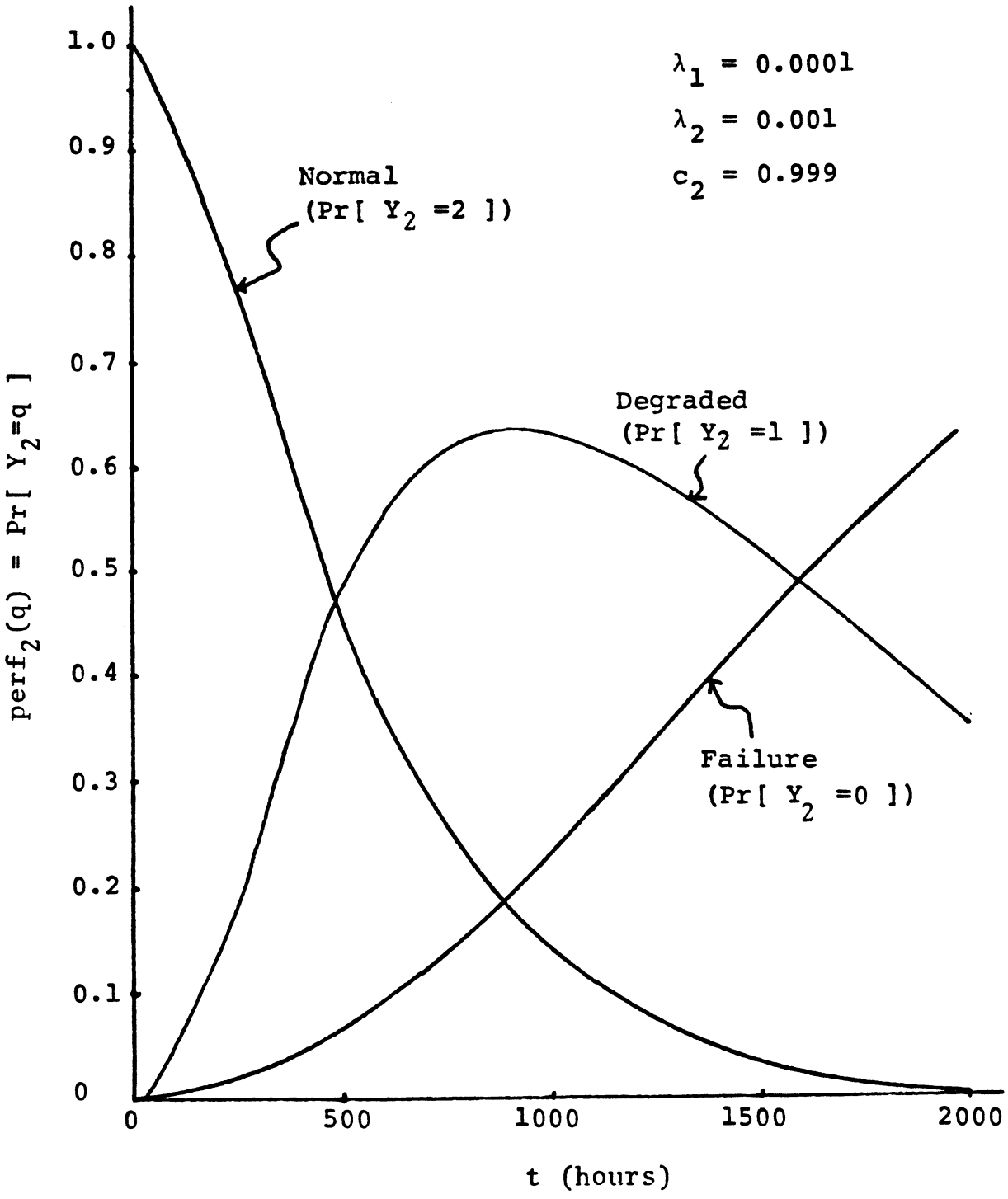


Figure 4.6

Performability of S_2

CHAPTER 5

PHASED MODELS

5.1 Phased-Missions

5.1.1 Introduction

The performability models considered in the previous chapters assume that the environment of the system is invariant in time in the sense that the underlying processes are time-homogeneous and the operational structures of the system remain the same throughout the utilization period. Although this assumption is appropriate for certain applications, there are many cases where the user's demands on the computing system can change appreciably during different phases of its utilization. This is particularly true for real-time control applications in which the computing system is required to execute different sets of computational tasks during different phases of a control process.

One approach to dealing with a time-varying environment is to decompose the system's utilization period into consecutive time periods (usually referred to as a decomposition of the system's "mission" into "phases"; see [42]-[45]). Demands on the system are then allowed to vary from phase to phase; within a given phase, however, they are assumed to be time invariant.

This permits intraphase behaviors to be evaluated in terms of conventional time-homogeneous models, but raises the interesting question of how the intraphase results are combined. This is the essential question addressed in investigations of phased-mission reliability evaluation methods (e.g., [42]-[45]) where the problem has been constrained as follows. It is assumed, first, that a success criterion (formulated, say, by a structure function; see [18] for example) can be established for each phase, where the criterion is independent of what occurs during other phases. It is required further that successful performance of the system be identified with success during all phases, that is, the system performs successfully if and only if, for each phase, the corresponding success criterion is satisfied throughout that phase.

Although the above constraints are reasonable for certain types of systems, they exclude systems where successful performance involves nontrivial interactions among the phases of the mission. In more exact terms, it has been shown (see [46] Theorem 6) that "structure-based" formulations of success are possible if and only if the phases are functionally independent in a precisely defined manner. What we wish to do, therefore, is to examine the utility of phased-mission evaluation methods in a less restrictive context.

In addition to removing the above constraints, we extend the domain of application to include evaluation of computing system performability. Moreover, by representing intraphase models in terms of operational models, we are able to obtain useful results even without the typical no-repair

assumption of the traditional phased-mission reliability methods.

Finally, unlike the models used in phased-mission reliability evaluation methods, we permit the intraphase models to differ from phase to phase. Thus, the modeling of a particular phase can be tailored not only to the computational demands of each phase but also to the relevant properties of the total system that influence performance during the phase.

5.1.2 Formulation

Intuitively, phased-missions are real-time control processes whose utilization period can be decomposed into phases. During each phase, the system is required to execute a predetermined set of computational tasks. A typical example of a phased-mission is an "unmanned space mission" during which the spacecraft's on-board computer must complete different phases of the mission. The analysis of such a system is usually complicated because of the time-varying nature of the system's performance criteria.

To generalize the notion of a phased-mission in the context of performability modeling, we assume that (i) the behavior of each single phase can be characterized by a single performance variable regardless of the interactions among phases, and (ii) interactions among phases can be characterized without reference to the detailed behavior of each phase. More precisely, let us suppose that the utilization period T is the continuous interval $T = [0,h]$. Suppose further that T is divided into a finite number of consecutive phases (time intervals) $T_1 = [t_0,t_1]$, $T_2 = [t_1,t_2]$,..., $T_m = [t_{m-1},t_m]$

where $0 = t_0 < t_1 < \cdots < t_m = h$. During each phase T_k, we assume that the system can be modeled by a performability model (X^k, γ^k) where

$$X^k = \{X_t^k | t \epsilon T_k\}$$

is a continuous-time stochastic process such that, for each t in T_k, X_t^k is a random variable taking values in the *phase k state space* Q_k ($X^k: \Omega \rightarrow Q_k$), and

$$\gamma^k: U_k \rightarrow A_k$$

is a function that maps the *phase k trajectory space* U_k to the phase k accomplishment set A_k. X^k is referred to as the *intraphase process* (of phase k) and γ^k is called the *phase k capability function*. When each phase can be represented by an intraphase performability model, a performability model (X, γ) of S is referred to as a *phased model* if it can be constructed by the following steps:

$$\text{(i) } X = \bigcup_{k=1}^{m} X^k = \bigcup_{k=1}^{m} \{X_t^k | t \epsilon T_k\} \tag{5.1}$$

(ii) there exists a function (referred to as an *organizing structure*)

$$\Psi: A_1 \times A_2 \times \cdots \times A_m \rightarrow A \tag{5.2}$$

such that, for all $u \epsilon U$,

$$\gamma(u) = \Psi(\gamma^1(u_1),...,\gamma^m(u_m))$$

where u_k is the restriction of u to T_k (k= 1,2,...,m).

On examining X we see that it is similar to a base model except that, for each time instant t_k ($1 \leq k \leq m$), the state of the system is represented by two random variables $X_{t_k}^k$ and $X_{t_k}^{k+1}$ whose values, respectively, are the final state of the k^{th} phase and the initial state of the $k+1^{th}$ phase. Since we permit the state sets of the intraphase models to differ from phase to phase, $X_{t_k}^k$ and $X_{t_k}^{k+1}$ can also be different. However, if we consider an augmented utilization period

$$\hat{T} = T \bigcup \{t_k' | k=1,2,...,m-1\}$$

(where t_k' can be interpreted as the initial time of phase k+1), then X can be expressed as

$$X = \{X_t | t \epsilon \hat{T}\}$$

where

$$X_t = \begin{cases} & \text{if } t = 0 \\ X_0^1 & \\ X_t^k & \text{if } t \epsilon (t_{k-1}, t_k) \\ X_{t_k}^{k+1} & \\ & \text{if } t = t_k'. \end{cases} \tag{5.3}$$

If, further, we regard the state space of X as the union

$$Q = \bigcup_{k=1}^{m} Q_k ,$$

then X is a base model in the sense defined in Chapter 2. When X is so constructed from intraphase processes, we will refer to it as a *phased base model*.

Generally, there are two types of dependencies among phases that can affect the performability evaluation of phased models. The first type, encountered when computing intraphase performabilities, are caused by statistical dependencies among phases. For example, if we assume that the k^{th} intraphase process X^k is a Markov process with a given transition probability function, then the performability of the system during phase k can be computed once the initial distribution of X^k is known. However, in general, the initial distribution of X^k is determined by that of the first phase together with the behavior of the previous phases in realizing a specific level of performance of the total system.

The second type of dependence occurs when combining intraphase performabilities to determine system performability. This type of dependence is determined by the algebraic relationship among phases, i.e., those relationships which do not involve probability concepts. In other words, the relationship is analogous to structure functions that are concerned with the structural representation of multicomponent systems (see [47], for example). Clearly, a complete analysis of phased models will require a detailed knowledge of both types of dependencies and their effects on the performability of the system.

In the following section, we first study the above algebraic relationship among phases via an extended definition of structure functions. In Section 5.3, we then consider the probabilistic aspects of the dependencies among phases. In both cases, we only assume that the behavior of the system during each phase can be summarized by a performance variable Y_k defined by the phase k performability model (X^k,γ^k). Finally, in Section 5.4, computational methods and formulas are derived, when Y_k can be defined as the minimum value assumed by a functional of X^k.

5.2 Structural Properties of Phased Models

In system theory, the structure of a system is generally taken to be the interactions among subsystems to perform certain specific tasks. The interaction may involve the physical interconnection of the subsystems or, more generally, "functional dependence" among subsystems [46]. Thus, if we

regard the phases of a phased model as subsystems, then the structure of a phased model can be effectively regarded as the interaction among phases in the realization of various degrees of system performance.

Since the relationship among phases can be represented as an organizing structure, and since the accomplishment sets of the phases are totally ordered sets (see (2.2)), there is a natural connection between the structure of a phased model and the mappings of partially ordered sets. More precisely, for each phase k, let us denote the phase k performance variable as

$$Y_k: \Omega \rightarrow A_k$$

defined by the phase k performability model (X^k, γ^k) according to (2.12). Then, by (5.2), the performance variable of the total system S can be represented as a random variable

$$Y: \Omega \rightarrow A$$

where, for each $\omega \in \Omega$,

$$Y(\omega) = \Psi(Y_1(\omega), \ldots, Y_m(\omega)) .$$

Thus, the structural relationship between $Y_1, Y_2, \ldots, Y_m$ can be characterized in terms of the properties of the mapping $\Psi: A_1 \times \cdots \times A_m \rightarrow A$. The product of

sets $A_1 \times \cdots \times A_m$ is a partially ordered set because, by extending the ordering relation of the individual phases, an ordering relation for $A_1 \times \cdots \times A_m$ can be defined as, for all $(a_1,a_2,...,a_m)$ and $(b_1,b_2,...,b_m)$ in $A_1 \times \cdots \times A_m$,

$$(a_1,a_2, \ldots ,a_m) \leq (b_1,b_2,...,b_m)$$

$$\text{iff } a_k \leq b_k \text{ for all } k = 1,2,...,m \; . \tag{5.4}$$

Our interest will be restricted to the case when $\Psi: A_1 \times \cdots \times A_m \rightarrow A$ is *order-preserving* in the sense that, for all $(a_1,a_2, \ldots ,a_m)$ and $(b_1,b_2,...,b_m)$ in $A_1 \times \cdots \times A_m$,

$$(a_1,a_2, \ldots ,a_m) \leq (b_1,b_2, \ldots ,b_m)$$

$$\text{implies } \Psi(a_1,a_2,...,a_m) \leq \Psi(b_1,b_2,...,b_m) \; .$$

Order-preserving mappings as defined above can be used to characterize systems whose performance do not deteriorate due to the performance improvements of the subsystems. Thus, the notion of an order-preserving mapping is a proper generalization of the notion of a coherent structure function [18] because the coordinates of $A_1 \times \cdots \times A_m$ are not restricted to binary valued sets.

Our investigation efforts will be focused on the properties of Ψ

which permit us to simplify the evaluation of system performability. In particular, given a subset B of A, we wish to consider methods for representing the set $\Psi^{-1}(B)$ without enumerating its elements. The representation methods are important because, as shown in the following discussions, such representations are amenable to iterative methods of evaluation.

We note first that the effects of Ψ on $A_1 \times \cdots \times A_m$ is that it imposes an order structure on the equivalence kernal of the order-preserving mapping. First, let us define, for all $a \epsilon A$,

$$C(a) = \{q | q \epsilon A_1 \times \cdots \times A_m \text{ and } \Psi(q) > a\} \tag{5.5}$$

and

$$D(a) = \{q | q \epsilon A_1 \times \cdots \times A_m \text{ and } \Psi(q) \geq a\} . \tag{5.6}$$

In words, D(a) is the set of elements in $A_1 \times \cdots \times A_m$ that result in at least level a accomplishment. Clearly, when Ψ is an order-preserving mapping, for all $a, b \epsilon A$,

$$a \leq b \text{ implies}$$

$$C(a) \supseteq C(b) \quad \text{and} \quad D(a) \supseteq D(b) . \tag{5.7}$$

Accordingly, if we let

$$\mathbf{C} = \{C(a) | a \epsilon A\} \tag{5.8}$$

and

$$\mathbf{D} = \{D(a) | a \epsilon A\} , \tag{5.9}$$

then, because A is a totally ordered set, it follows that C and D are totally ordered sets with respect to set inclusion. Moreover, if the elements in A are expressed as a sequence

$$\cdots < a_1 < a_2 < \cdots < a_i < \cdots$$

where, for all i, a_{i+1} *covers* a_i in the sense that $a_{i+1} > a_i$ and for no $a \epsilon A$, $a_{i+1} > a > a_i$, then the corresponding elements in C and D can be expressed in a like manner, i.e.,

$$\cdots \supseteq C(a_1) \supseteq C(a_2) \supseteq \cdots \supseteq C(a_i) \supseteq \cdots$$

and

$$\cdots \supseteq D(a_1) \supseteq D(a_2) \supseteq \cdots \supseteq D(a_i) \supseteq \cdots \quad .$$

Hence, if we denote the difference between two sets X and Y by X-Y, then it can easily be shown that

Theorem 5.1:

Let A be a totally ordered and countable set and let $\Psi:A_1\times\cdots\times A_m\rightarrow A$ be an order-preserving mapping. Then, for all $a,b\epsilon A$, a covers b implies, for all $q,r\epsilon A_1\times\cdots\times A_m$,

(1) $\Psi(q) = \Psi(r) = a$ if and only if both q and r belong to C(b)-C(a),

(2) $\Psi(q) = \Psi(r) = b$ if and only if both q and r belong to D(b)-D(a).

The above results imply that, when evaluating system performability based on a phased model, the sets C(a) and D(a) where $a\epsilon A$ can be used as building blocks for describing events that characterize system performance. In particular, given an order-preserving mapping $\Psi:A_1\times\cdots\times A_m\rightarrow A$ and a closed interval $B=[a,b]\subseteq A$, we show in the following theorems that $\Psi^{-1}(B)$ can be specified by D(a) and C(b). The set $\Psi^{-1}(B)$ has practical significance in performability modeling because its probability quantifies the ability of S to perform within the specified limits a and b.

First, we note that Cartesian subsets of $A_1\times\cdots\times A_m$ are amenable to iterative methods of evaluation. A subset $V\subseteq A_1\times\cdots\times A_m$ is called a *Cartesian* subset if $V=\xi_1(V)\times\cdots\times\xi_m(V)$ where $\xi_k(V)$ is the projection of V onto its k^{th} coordinate. To illustrate the use of Cartesian sets in the evaluation of performabilities, let us consider the evaluation of a specific Cartesian subset

V. We first let

$$B_k = \{q \epsilon A_1 \times \cdots \times A_m | \xi_k(q) \epsilon \xi_k(V)\}$$

be the set of elements in $A_1 \times \cdots \times A_m$ that assume values in $\xi_k(V)$ at the k^{th} coordinate. Then, the probability of B_k can be expressed as a one-dimensional distribution of the phase k performance variables Y_k, i.e.,

$$\Pr[B_k] \triangleq \Pr[(Y_1, Y_2, \ldots, Y_m) \epsilon B_k]$$

$$\triangleq \Pr[Y_k \epsilon \xi_k(V)] .$$

Accordingly, when V is a Cartesian set, V clearly can be represented as the intersection of those elementary sets B_k, i.e.,

$$V = \xi_1(V) \times \cdots \times \xi_m(V)$$

$$= \bigcap_{k=1}^{m} B_k .$$

By iteratively applying the definition of conditional probability, then

$$\Pr[V] = \Pr[B_m | \bigcap_{k=1}^{m-1} B_k] \cdot \Pr[B_{m-1} | \bigcap_{k=1}^{m-2} B_k]$$

$$\cdots \Pr[B_2|B_1] \cdot \Pr[B_1]$$

$$= \Pr[Y_m \epsilon \xi_m(V) | Y_{m-1} \epsilon \xi_{m-1}(V), \ldots, Y_1 \epsilon \xi_1(V)]$$

$$\cdots \Pr[Y_2 \epsilon \xi_2(V) | Y_1 \epsilon \xi_1(V)] \cdot \Pr[Y_1 \epsilon \xi_1(V)] . \tag{5.10}$$

Since each term in the product involves only elementary sets B_k, we show in the following section that Pr[V] can be determined iteratively using matrix multiplications.

An important class of Cartesian subsets of $A_1 \times \cdots \times A_m$ is the sets of "intervals." For all $q,r \epsilon A_1 \times \cdots \times A_m$ where $q=(a_1,a_2,\ldots,a_m)$, $r=(b_1,b_2,\ldots,b_m)$ and $q \leq r$, a *closed interval* [q,r] is defined to be

$$[q,r] = \{q' \epsilon A_1 \times \cdots \times A_m | q \leq q' \leq r\} .$$

It follows that

$$[q,r] = \xi_1([q,r]) \times \cdots \times \xi_m([q,r])$$

$$= [a_1,b_1] \times \cdots \times [a_m,b_m] ,$$

hence [q,r] is Cartesian. The *open interval* (q,r), *half-open intervals* (q,r] and

[q,r) can be defined in a like manner. Moreover, if we assume that $A_1 \times \cdots \times A_m$ has *greatest* and *least* elements I and 0, satisfying

$$q \geq 0 \quad \text{and} \quad I \geq q$$

for all q in $A_1 \times \cdots \times A_m$, then $A_1 \times \cdots \times A_m$ can be expressed as a closed interval [0,I].

Given an interval B of A, an important problem then is to find the "representation" of $\Psi^{-1}(B)$ in terms of intervals of $A_1 \times \cdots \times A_m$. The problem is interesting not only because the representation permits us to evaluate $Pr[\Psi^{-1}(B)]$ using the iterative algorithm described above but also because the representation reduces the number of elements needed to describe the set $\Psi^{-1}(B)$.

To express $\Psi^{-1}(B)$ as intervals, we note first that since Ψ is order-preserving, for all $q,r \in A_1 \times \cdots \times A_m$, we have

$$q \leq x \leq r \quad \text{implies} \quad \Psi(q) \leq \Psi(x) \leq \Psi(r) \tag{5.11}$$

(for all $x \in A_1 \times \cdots \times A_m$). Thus, if B is an interval of A and $q,r \in \Psi^{-1}(B)$, it follows that

$$[q,r] \subseteq \Psi^{-1}(B) .$$

Moreover, for each $a \epsilon A$, let C(a) and D(a) be the sets as defined in (5.5) and (5.6), and denote the set complement of C(a) by $\sim C(a)$. If we define

$$M(a) \triangleq \{q \epsilon A_1 \times \cdots \times A_m | q \text{ is a maximal element of } \sim C(a)\}, \tag{5.12}$$

and

$$m(a) \triangleq \{q \epsilon A_1 \times \cdots \times A_m | q \text{ is a minimal element of } D(a)\}, \tag{5.13}$$

then, applying (5.11), it follows that

Lemma:
If $\Psi: A_1 \times \cdots \times A_m \rightarrow A$ is order-preserving, then

$$\Psi^{-1}([a,b]) \supseteq \bigcup_{\substack{q \epsilon m(a) \\ r \epsilon M(b)}} [q,r] . \tag{5.14}$$

In general, the relation $\supseteq$ in (5.14) can not be replaced by an equality. However, we found that the equality holds when $A_1 \times \cdots \times A_m$ satisfies the chain conditions [48]; A partially ordered set L is said to satisfy the *ascending chain condition* when every nonempty subset of L has a maximal element. Similarly, L is said to satisfy the *descending chain condition* when every nonempty subset of L has a minimal element. L is said to satisfy the *chain*

conditions when it satisfies both chain conditions. When the chain conditions are imposed on the above lemma, we then have the following result:

Theorem 5.2:

If $\Psi: A_1 \times \cdots \times A_m \rightarrow A$ is order-preserving and $A_1 \times \cdots \times A_m$ satisfies the chain conditions, then for all $a,b \in A$,

$$\Psi^{-1}([a,b]) = \bigcup_{\substack{q \in m(a) \\ r \in M(b)}} [q,r] \,. \tag{5.15}$$

Proof:

We only need to show that

$$\Psi^{-1}([a,b]) \subseteq \bigcup_{\substack{q \in m(a) \\ r \in M(b)}} [q,r] \,,$$

i.e., for all $x \in A_1 \times \cdots \times A_m$, we have to show that $a \leq \Psi(x) \leq b$ implies $q \leq x \leq r$, for some $q \in m(a)$ and $r \in M(b)$. First, for each fixed $x \in A_1 \times \cdots \times A_m$, let

$$K_1 = \{y \in A_1 \times \cdots \times A_m \mid \Psi(y) \leq b \text{ and } x \leq y\}$$

and

$$K_2 = \{y \in A_1 \times \cdots \times A_m \mid a \leq \Psi(y) \text{ and } y \leq x\} \,.$$

Then K_1 and k_2 are clearly nonempty since $x \epsilon K_1$ and $x \epsilon K_2$. Moreover, since $A_1 \times \cdots \times A_m$ satisfies the chain conditions, K_1 must satisfy the ascending chain condition and K_2 must satisfy the descending chain condition. Hence K_1 contains a maximal element, say r, and K_2 contains a minimal element, say q, that is $q \leq x \leq r$, for some $q \epsilon K_2$ and $r \epsilon K_1$. Finally, we note that r is a maximal element of K_1 implies $r \epsilon M(b)$ and q is a minimal element of K_2 implies $q \epsilon m(a)$, i.e., $x \epsilon [q,r]$ for some $q \epsilon m(a)$ and $r \epsilon M(b)$.

Note that, for each $a \epsilon A$, the set M(a) is an *unordered set* in the sense that, for all q_1 and q_2 in M(a), neither $q_1 \leq q_2$ nor $q_1 \geq q_2$ unless $q_1 = q_2$. Hence, the number of elements in M(a) is bounded by the *width* of $A_1 \times \cdots \times A_m$ defined to be a natural number n if and only if there is an unordered subset K of $A_1 \times \cdots \times A_m$ of n elements such that all unordered subsets of $A_1 \times \cdots \times A_m$ have no more than n elements [48]. Similarly, for all $a \epsilon A$, the cardinality of m(a) is no larger than the width of $A_1 \times \cdots \times A_m$.

Since a finite partially ordered set always satisfy the chain conditions and the width of a finite partially ordered set is finite, we also obtain the following result:

Corollary:

If $\Psi: A_1 \times \cdots \times A_m \rightarrow A$ is order-preserving and $A_1 \times \cdots \times A_m$ is finite, then for all intervals B in A, $\Psi^{-1}(B)$ can be expressed as the union of a finite number of intervals in $A_1 \times \cdots \times A_m$.

In addition to the conditions of Theorem 5.2, Equation (5.15) also holds when $\Psi: A_1 \times \cdots \times A_m \rightarrow A$ is a lattice homomorphism. A partially ordered set L is called a *lattice* when any two of whose elements x and y have a least upper bound denoted by $x \vee y$, and a greatest lower bound denoted by $x \wedge y$ [48]. Clearly, every totally ordered set L is a lattice because, for any $q, r \in L$,

$$q \vee r = \begin{cases} q & \text{if } q \leq r \\ r & \text{otherwise,} \end{cases}$$

Moreover, since $A_1, A_2, \ldots, A_m$ are totally ordered sets, the least upper bound and the greatest lower bound of any two elements $(a_1, a_2, \ldots, a_m)$ and $(b_1, b_2, \ldots, b_m)$ in $A_1 \times \cdots \times A_m$ can be defined as

$$(a_1, a_2, \ldots, a_m) \vee (b_1, b_2, \ldots, b_m)$$

$$\triangleq (a_1 \wedge b_1, a_2 \wedge b_2, \ldots, a_m \wedge b_m) . \tag{5.16}$$

and

$$(a_1, a_2, \ldots, a_m) \wedge (b_1, b_2, \ldots, b_m)$$

$$\triangleq (a_1 \wedge b_1, a_2 \wedge b_2 \ldots, a_m \wedge b_m) . \tag{5.17}$$

Now, since $A_1 \times \cdots \times A_m$ is a partially ordered set, it follows immediately from (5.16) and (5.17) that $A_1 \times \cdots \times A_m$ is also a lattice.

When the mapping $\Psi: A_1 \times \cdots \times A_m \rightarrow A$ is a *lattice homomorphism* in the sense that, for all $q, r \in A_1 \times \cdots \times A_m$,

$$\Psi(q \vee r) = \Psi(q) \vee \Psi(r)$$

and

$$\Psi(q \wedge r) = \Psi(q) \wedge \Psi(r) ,$$

we are able to show that

Theorem 5.3:

If $\Psi: A_1 \times \cdots \times A_m \rightarrow A$ is a lattice homomorphism and $A_1 \times \cdots \times A_m$ satisfies the chain conditions, then, for any interval B in A, $\Psi^{-1}(B)$ can be expressed as an interval of $A_1 \times \cdots \times A_m$ in one and only one way.

Proof:

We note first that a lattice homomorphism is order-preserving because, when $q \geq r$,

$$\Psi(q \wedge r) = \Psi(q) \wedge \Psi(r) = \Psi(r)$$

and

$$\Psi(q \vee r) = \Psi(q) \vee \Psi(r) = \Psi(q) .$$

Moreover, given an interval B in A, $\Psi^{-1}(B)$ contains at most one maximal element. To provide this statement, let us suppose that $\Psi^{-1}(B)$ contains two maximal elements q and r. Since A is a totally ordered set, we have either $\Psi(q) \geq \Psi(r)$ or $\Psi(q) \leq \Psi(r)$. If we assume $\Psi(q) \geq \Psi(r)$, then

$$\Psi(q \vee r) = \Psi(q) \vee \Psi(r) = \Psi(q) .$$

But this implies q can not be a maximal element of $\Psi^{-1}(B)$ unless $q \vee r = q$ or, equivalently, $q \geq r$. However, since r is also a maximal element of $\Psi^{-1}(B)$, $q \geq r$ implies $q = r$. In other words, $\Psi^{-1}(B)$ contains at most one maximal element. By similar argument, we can also show that $\Psi^{-1}(B)$ contains at most one minimal element.

If we assume that $\Psi^{-1}(B)$ is nonempty then, since $A_1 \times \cdots \times A_m$ satisfies the chain conditions, $\Psi^{-1}(B)$ contains a unique maximal element, say q_2, and a unique minimal element, say q_1. Now since Ψ is order-preserving, we have

$$\Psi^{-1}(B) \supseteq [q_1,q_2] .$$

Thus, it remains to be shown that

$$\Psi^{-1}(B) \subseteq [q_1,q_2] ,$$

i.e., $x \epsilon \Psi^{-1}(B)$ implies $q_1 \leq x \leq q_2$. This follows immediately since q_2 is also a maximal element of the set $\{y \epsilon \Psi^{-1}(B) | y \geq x\}$ and q_1 is a minimal element of the set $\{y \epsilon \Psi^{-1}(B) | y \leq x\}$.

Theorem 5.2 and 5.3, in our opinion, have established a feasible method for the evaluation of system performability based on the notion of a phased model. To illustrate the application of the method, let us consider the following hypothetical two-phase model. During each phase, it is assumed that the phase k performance variable is given by

$$Y_k: \Omega \rightarrow A_k \qquad (k=1 \text{ or } 2)$$

where $A_k = \{0,1,2\}$ and

$$Y_k = \begin{cases} 2 & \text{if the system operates in the nondegraded mode throughout phase k,} \\ 1 & \text{if the sysem enters degraded mode during phase k while remains operational throughout the phase,} \\ 0 & \text{otherwise.} \end{cases}$$

It is further assumed that the performance variable of the total system S is given by

$$Y: \Omega \rightarrow A$$

where $A=\{0,1,2\}$ and

$$Y = \begin{cases} 2 & \text{if the system operates in the nondegraded mode throughout all phases,} \\ 1 & \text{if the system remains operational for at least one phase} \\ 0 & \text{otherwise.} \end{cases}$$

Then, the organizing structure can be expressed as an order-preserving mapping

$$\Psi: \{0,1,2\} \times \{0,1,2\} \rightarrow \{0,1,2\}$$

as illustrated in Figure 5.1. Note that in the figure the domain and the

codomain of the mapping are both represented by diagrams where two nodes q and r are connected by a downward line when q covers r.

Suppose that the user is interested in evaluating the probability of the event Y =1 or 0, i.e., the probability of encountering degraded performance or system failure. Applying Theorem 5.2, the event $\Psi^{-1}([0,1])$ can be expressed as

$$\Psi^{-1}([0,1]) = [(0,0),(2,1)] \cup [(0,0),(1,2)]$$

where [(0,0),(2,1)] and [(0,0),(1,2)] are intervals of the partially ordered set {0,1,2}×{0,1,2}. Note that the intersection of [(0,0),(2,1)] and [(0,0),(1,2)] is also an interval [(0,0),(1,1)]. Accordingly, the probability of $\Psi^{-1}([0,1])$ can be expressed as

$$\begin{aligned}
&\Pr[\Psi^{-1}([0,1])] \\
&= \Pr[[0,0),(2,1)] \cup [(0,0),(1,2)]] \\
&= \Pr[[(0,0),(2,1)]] + \Pr[[(0,0),(1,2)]] \\
&\qquad - \Pr[[(0,0),(1,1)]] .
\end{aligned}$$

Since an interval of {0,1,2}×{0,1,2} is a Cartesian subset, each of the last three terms of the above equality can be evaluated iteratively using the solution method described in Section 5.3 for Cartesian subsets.

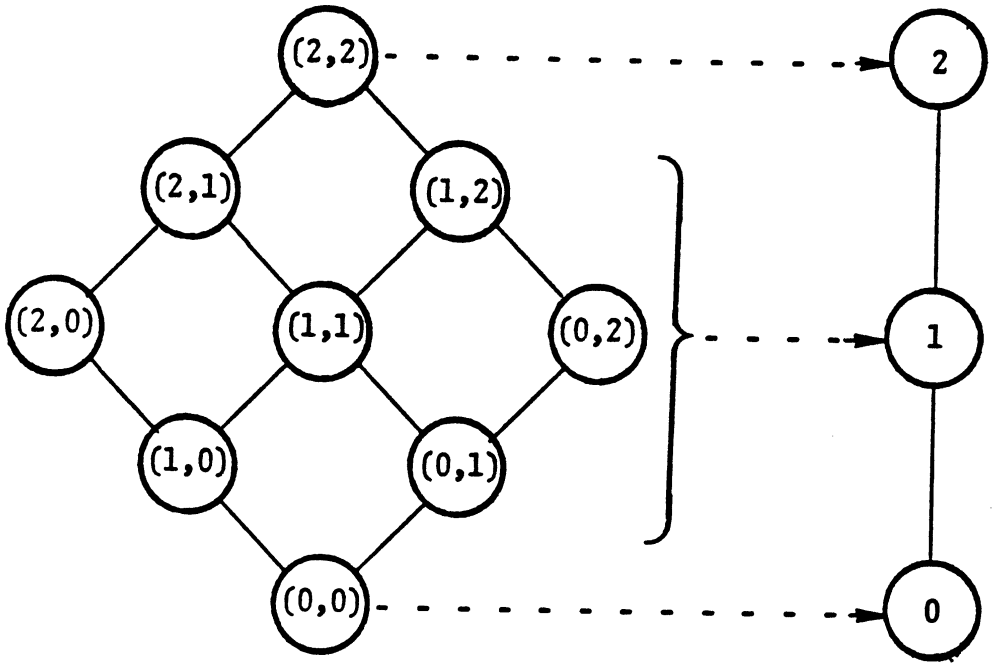


Figure 5.1

An Order-Preserving Mapping

$\Psi:\{0,1,2\}^2 \rightarrow \{0,1,2\}$

In general, given an order-preserving mapping $\Psi: A_1 \times \cdots \times A_m \rightarrow A$ and an interval B of A, let us suppose $\Psi^{-1}(B)$ can be expressed as the set union of a finite number of intervals $I_1, I_2, \ldots, I_N$, i.e.,

$$\Psi^{-1}(B) = I_1 \cup I_2 \cup \cdots \cup I_N .$$

To generalize the above evaluation method, let us define

$$S_1 = \sum_i \Pr[I_i]$$

$$S_2 = \sum_{i,j} \Pr[I_i \cap I_j]$$

$$S_3 = \sum_{i,j,k} \Pr[I_i \cap I_j \cap I_k]$$

$$\vdots$$

$$S_N = \Pr[I_1 \cap I_2 \cap \cdots \cap I_N]$$

where $1 \leq i < j < k < \ldots \leq N$ so that in the sums each combination appears once and only once; hence S_ℓ has $\binom{N}{\ell}$ terms. Then, since the intersection of intervals is an interval, each of the sums can be evaluated by repeatedly applying the computational algorithms described in the following section.

Hence, by the method of inclusion and exclusion (see [51], p. 89), the probability of $\Psi^{-1}(B)$ can be computed by the well-known formula

$$\Pr[\Psi^{-1}(B)] = S_1 - S_2 + S_3 - S_4 + \cdots \pm S_N . \tag{5.18}$$

5.3 Probability Computation of Cartesian Trajectory Sets

Given a phased model (X,γ) satisfying assumptions (5.1) and (5.2) the performability model can be simplified as follows. The simplified base model is taken to be the imbedded discrete-time process

$$\hat{X}_S = \{Y_k | k=1,2,...,m\}$$

where, for each $k=1,2,...,m$, Y_k is the phase k performance variable. The trajectory space of X can be effectively regarded as the product space

$$\bar{U} = A_1 \times A_2 \times \cdots \times A_m$$

where A_k is the accomplishment set of phase k. The corresponding simplification of γ is the organizing structure

$$\Psi : \bar{U} \rightarrow A$$

as defined in (5.2). Then, it follows that, for all a in A, the probability that S performs at level a is

$$\text{perf}(a) = \Pr[\gamma^{-1}(a)] = \Pr[\Psi^{-1}(a)]$$

and hence the performability model $(\hat{X}_S,\Psi)$ can be used to evaluate the performability of S. We will thus refer to (X,Ψ) as being equivalent to the model (X,γ).

Generally, given an equivalent performability model $(\hat{X}_S,\Psi)$, the evaluation of $\Pr[\Psi^{-1}(a)]$ requires a detailed knowledge of how intraphase processes cooperate to accomplish level a, i.e., a thorough understanding of their functional dependencies (see [46]). The difficulties are further aggravated by statistical dependencies between phases. However, we show in the following discussions that when a trajectory set $V\subseteq\bar{U}$ is Cartesian in the sense that, for every phase k, there exists $R_k\subseteq A_k$ such that $V=R_1\times R_2\times\cdots\times R_m$, then Pr[V] can be determined iteratively using matrix multiplications. Moreover, given this ability to compute the probabilities of Cartesian sets, the probabilities of more general sets can be determined by decomposing them into Cartesian components (see (5.18)). Hence, the problem reduces to that of computing the probabilities of Cartesian trajectory sets.

If, for each phase k, let n_k be the number of states in Q_k. Then, for a Cartesian trajectory set $V=R_1\times R_2\times\cdots\times R_m$, the conditional

intraphase transition matrix of the k^{th} phase is the $n_k \times n_k$ matrix $P_{V,k}$ where, for all $i,j \epsilon Q_k$,

$$P_{V,k}(i,j) = Pr[Y_k \epsilon R_k, X_{t_k}^k = j | X_{t_{k-1}}^k = i, Y_{k-1} \epsilon R_{k-1}, \ldots, Y_1 \epsilon R_1] \qquad (5.19)$$

where $X_{t_{k-1}}^k$ and $X_{t_k}^k$ respectively are the initial and the final states of phase k intraphase process (see (5.1)). In other words, $P_{V,k}(i,j)$ is the probability of having performance levels R_k during phase k while the intraphase process initiates in state i and ends up in state j, conditional by the first k-1 components of V. Similarly, for all but the first phase, the conditional *interphase transition* matrix is the $n_{k-1} \times n_k$ matrix $H_{V,k}$ where, for all $i \epsilon Q_{k-1}$ and all $j \epsilon Q_k$,

$$H_{V,k}(i,j) = Pr[X_{t_{k-1}}^k = j | X_{t_{k-1}}^{k-1} = i, Y_{k-1} \epsilon R_{k-1}, \ldots, Y_1 \epsilon R_1] \, . \qquad (5.20)$$

In other words, $H_{V,k}(i,j)$ is the probability that the k^{th} phase initiates in state j given that the final state of the $k-1^{th}$ phase is i, conditioned by the first k-1 components of V. Finally, for consistency, we let $H_{V,1}$ be the $n_1 \times n_1$ identity matrix. In terms of the above matrices, we are able to establish the following matrix formula for computing the probability of a Cartesian trajectory set V. Given X, let p denotes the initial state distribution, i.e., $p = [p_1 \; p_2 \; \cdots \; p_{n_1}]$ where $p_i = Pr[X_0^1 = i]$, and let F_k denote the $n_k \times 1$ column matrix with "1" in each entry. Then, by induction on k, it can be established that

Theorem 5.4:

If $V = R_1 \times \cdots \times R_k \times Q_{k+1} \times \cdots \times Q_m$ then

$$\Pr[V] = p \cdot [\prod_{\ell=1}^{k} H_{V,\ell} \cdot P_{V,\ell}] \cdot F_k \tag{5.21}$$

Proof:

For $k=1$,

$$p \cdot H_{V,1} \cdot P_{V,1} = p \cdot P_{V,1} = [a_1 \ \cdots \ a_j \ \cdots \ a_{n_1}]$$

where

$$a_j = \sum_{i \epsilon Q_1} \Pr[X_0^1 = i] \cdot \Pr[Y_1 \epsilon R_1, X_{t_1}^1 = j | X_0^1 = i]$$

$$= \sum_{i \epsilon Q_1} \Pr[Y_1 \epsilon R_1, X_{t_1}^1 = j, X_0^1 = i]$$

$$= \Pr[Y_1 \epsilon R_1, X_{t_1}^1 = j] \ .$$

Multiplied by F_1,

$$p \cdot H_{v,1} \cdot P_{v,1} \cdot F_1$$

$$= \sum_{j \epsilon Q_1} Pr[Y_1 \epsilon R_1, X_{t_1}^1 = j] = Pr[Y_1 \epsilon R_1]$$

$$= Pr[Y_1 \epsilon R_1, Y_2 \epsilon Q_2, \ldots, Y_m \epsilon Q_m]$$

$$= Pr[V] .$$

Suppose that the formula holds for $k<m$, then

$$p \cdot [\prod_{\ell=1}^{k+1} H_{v,\ell} \cdot P_{v,\ell}] \cdot F_{k+1}$$

$$= p \cdot [\prod_{\ell=1}^{k} H_{v,\ell} \cdot P_{v,\ell}] \cdot H_{v,k+1} \cdot P_{v,k+1} \cdot F_{k+1}$$

$$= A_1 \cdot H_{v,k+1} \cdot P_{v,k+1} \cdot F_{k+1}$$

where

$$A_1 = [b_1 \cdots b_j \cdots b_{n_1}]$$

and

$$b_j = Pr[X_{t_k}^k = j, Y_k \epsilon R_k, \ldots, Y_1 \epsilon T_1]$$

by applying the equation for k.

When we iteratively compute the matrix product, beginning from the left, the first two terms become

$$A_2 = A_1 \cdot H_{V,k+1} = [c_1 \ \cdots \ c_j \ \cdots \ c_{n_{k+1}}]$$

where

$$c_j = \sum_{i \epsilon Q_k} b_i \cdot H_{V,k+1}(i,j)$$

$$= \sum_{i \epsilon Q_k} \Pr[X_{t_k}^k = i, Y_k \epsilon R_k, \ldots, Y_1 \epsilon R_1]$$

$$\Pr[X_{t_k}^{k+1} = j | X_{t_k}^k = i, Y_k \epsilon R_k, \ldots, Y_1 \epsilon R_1]$$

$$= \Pr[X_{t_k}^{k+1} = j, Y_k \epsilon R_k, \ldots, Y_1 \epsilon R_1] \ .$$

The next partial product is the result of multiplying A_2 by the transition matrix $P_{V,k+1}$ which yields:

$$A_3 = A_2 \cdot P_{V,k+1} = [d_1 \ \cdots \ d_j \ \cdots \ d_{n_{k+1}}]$$

where

$$d_j = \sum_{i \epsilon Q_{k+1}} c_i \cdot P_{V,k+1}(i,j)$$

$$= \sum_{i \epsilon Q_{k+1}} \Pr[X_{t_k}^{k+1}=i, Y_k \epsilon R_k, \ldots, Y_1 \epsilon R_1]$$

$$\cdot \Pr[Y_{k+1} \epsilon R_{k+1}, X_{t_{k+1}}^{k+1}=j | X_{t_k}^{k+1}=i, Y_k \epsilon R_k, \ldots, Y_1 \epsilon R_1]$$

$$= \Pr[X_{t_{k+1}}^{k+1}=j, Y_{k+1} \epsilon R_{k+1}, \ldots, Y_1 \epsilon R_1] .$$

The product is completed by multiplying A_3 by the summing vector F_{k+1}, that is,

$$p \cdot [\prod_{\ell=1}^{k+1} H_{V,\ell} \cdot P_{V,\ell}] \cdot F_{k+1}$$

$$= A_3 \cdot F_{k+1}$$

$$= \sum_{j \epsilon Q_{k+1}} \Pr[X_{t_{k+1}}^{k+1}=j, Y_{k+1} \epsilon R_{k+1}, \ldots, Y_1 \epsilon R_1]$$

$$= \Pr[Y_{k+1} \epsilon R_{k+1}, Y_k \epsilon R_k \ldots, Y_1 \epsilon R_1]$$

$$= \Pr[Y_1 \epsilon R_1, \ldots, Y_{k+1} \epsilon R_{k+1}, Y_{k+2} \epsilon Q_{k+2}, \ldots, Y_m \epsilon Q_m]$$

$$= \Pr[R_1 \times \cdots \times R_{k+1} \times Q_{k+2} \times \cdots \times Q_m] .$$

Accordingly, the equation holds for all $k \leq m$, which completes the proof of Theorem 5.4.

In particular, for $k=m$, we have

Corollary:

For any Cartesian set $V=R_1 \times R_2 \times \cdots \times R_m$,

$$\Pr[V] = p \cdot [\prod_{k=1}^{m} H_{V,k} \cdot P_{V,k}] \cdot F_m \,. \tag{5.22}$$

Although Equation (5.22) provides us with a general formula for computing the probability of a Cartesian set, its disadvantages derive from the fact that the $H_{V,k}$ and $P_{V,k}$ matrices may be difficult to obtain in practical applications. In particular, these matrices will generally depend on V as well as X and, moreover, will generally depend on the history of X before phase k. However, the latter objections disappear when the transition probabilities are "memoryless." More precisely, let the (unconditional) *intraphase transition* matrix of the k^{th} phase be the $n_k \times n_k$ matrix $\bar{P}_{V,k}$ where, for all $i,j \in Q_k$,

$$\bar{P}_{V,k}(i,j) = \Pr[X_{t_k}^{k}=j,\ Y_k \in R_k | X_{t_{k-1}}^{k}=i] \tag{5.23}$$

i.e., the probability of having performance levels R_k during phase k while the intraphase process initiates in state i and ends up in state j. Similarly, let the (unconditional) interphase transition matrix be the $n_{k-1} \times n_k$ matrix H_k where, for all $i \in Q_{k-1}$ and $j \in Q_k$,

$$H_k(i,j) = \Pr[X^k_{t_{k-1}}=j | X^{k-1}_{t_{k-1}}=i] \tag{5.24}$$

i.e., the probability that the k^{th} intraphase process initiates in state j given that the $k-1^{th}$ intraphase process ends up in state i. Then the intraphase transitions of (X,γ) are *memoryless for V at phase k* if

$$P_{V,k} = \bar{P}_{V,k} .$$

Similarly, the interphase transitions of (X,γ) are *memoryless for V at phase k* if

$$H_{V,k} = H_k .$$

Accordingly, when transitions are memoryless through phase k, by the definitions and Theorem 5.4, we obtain

Theorem 5.5:

If $V = R_1 \times R_2 \times \cdots \times R_k \times Q_{k+1} \times \cdots \times Q_m$ and the intraphase and interphase transitions of X are memoryless for V through phase k, then

$$\Pr[V] = p \cdot [\prod_{\ell=1}^{k} H_\ell \cdot \bar{P}_{V,\ell}] \cdot F_k . \tag{5.25}$$

Corollary:

For any Cartesian set V, if the intraphase and interphase transitions of X are memoryless for V for all phases, then

$$\Pr[V] = p\cdot[\prod_{\ell=1}^{k} H_\ell\cdot\bar{P}_{V,\ell}]\cdot F_k \,. \tag{5.26}$$

When V is a Cartesian set and $R_\ell = Q_\ell$, for $\ell = 1,2,\ldots,k-1$, then the intraphase and interphase transitions of X are memoryless for V through phase k. Accordingly, applying Theorem 5.5, we obtain the following formula for the probability of the trajectory set $V = Q_1\times\cdots\times Q_{k-1}\times R_k\times Q_{k+1}\times Q_m$ which, alternatively, is the probability of the event $Y_k\epsilon R_k$.

Theorem 5.6:

If $V = Q_1\times\cdots\times Q_{k-1}\times R_k\times Q_{k+1}\times\cdots\times Q_m$, then

$$\Pr[V] = p\cdot[\prod_{\ell=1}^{k} H_\ell\cdot\bar{P}_{V,\ell}]\cdot F_k \,. \tag{5.27}$$

By Theorems 5.5 and 5.6, when certain intraphase and interphase transitions are memoryless for V, the probability of a Cartesian set V is easily obtainable. However, such results may still be difficult to use because, even though the transitions are memoryless for V, they may not be memoryless for other Cartesian sets. Accordingly, we have sought to identify stronger conditions under which the formulas will hold for all Cartesian trajectory sets.

First, by extending previous definitions, the intraphase (interphase) transitions of $(X,\dot{\gamma})$ are *memoryless at phase k* if they are memoryless for all Cartesian sets V at phase k; the intraphase (interphase) transitions of (X,γ) are *memoryless* if they are memoryless at all phases. The advantages of memoryless transitions are obvious, for by their definitions and the corollary to Theorem 5.4, we have

Theorem 5.7:

If (X,γ) is a phased model and the intraphase and interphase transitions of X are memoryless, then, for all Cartesian sets V,

$$\Pr[V] = p\cdot[\prod_{k=1}^{m} H_k\cdot\bar{P}_{V,k}]\cdot F_m \ . \tag{5.28}$$

Moreover, we find that the memoryless property is relatively easy to characterize, that is, we are able to show the following characterization conditions for the memoryless property. Note that the conditions do not involve any specific Cartesian sets.

Theorem 5.8:

(1) The intraphase transitions of X are memoryless at phase k if and only if, for all $i,j\epsilon Q_k$ and all a_ℓ in A_ℓ ($\ell = 1,2,...,k$),

$$\Pr[X_{t_k}^k=j, Y_k=a_k | X_{t_{k-1}}^k=i, Y_{k-1}=a_{k-1},\ldots,Y_1=a_1]$$

$$= \Pr[X_{t_k}^k=j, Y_k=a_k | X_{t_{k-1}}^k=i] \ . \tag{5.29}$$

(2) The interphase transitions of X are memoryless at phase k if and only if, for all $i\epsilon Q_{k-1}$, $j\epsilon Q_k$ and all $a_\ell \epsilon A_\ell$ ($\ell = 1,2,\ldots,k-1$),

$$\Pr[X_{t_{k-1}}^k=j | X_{t_{k-1}}^{k-1}=i, Y_{k-1}=a_{k-1},\ldots,Y_1=a_1]$$

$$= \Pr[X_{t_{k-1}}^k=j | X_{t_{k-1}}^{k-1}=i] \ . \tag{5.30}$$

Proof:

Suppose $P_{V,k}$ is memoryless for all Cartesian sets $V = R_1 \times R_2 \times \cdots \times R_m$. By taking R_ℓ to be the singleton set $\{a_\ell\}$ ($\ell = 1,2,\ldots,k$),

$$P_{V,k}(i,j) = \Pr[X_{t_k}^k=j, Y_k=a_k | X_{t_{k-1}}^k=i, Y_{k-1}=a_{k-1},\ldots,Y_1=a_1]$$

$$= \Pr[X_{t_k}^k=j, Y_k=a_k | X_{t_{k-1}}^k=i]$$

$$= \bar{P}_{V,k}(i,j) \ .$$

Now, suppose that, for all $a_\ell \epsilon A_\ell$ ($\ell = 1,2,\ldots,k$),

$$\Pr[X^k_{t_k}=j, Y_k=a_k | X^k_{t_{k-1}}=i, Y_{k-1}=a_{k-1}, \ldots, Y_1=a_1]$$

$$= \Pr[X^k_{t_k}=j, Y_k=a_k | X^k_{t_{k-1}}=i] \, .$$

Then, for any Cartesian set $V = R_1 \times R_2 \times \cdots \times R_m$,

$$P_{V,k}(i,j) = \Pr[X^k_{t_k}=j, Y_k \epsilon R_k | X^k_{t_{k-1}}=i, Y_{k-1} \epsilon R_{k-1}, \ldots, Y_1 \epsilon R_1]$$

$$= \frac{\sum_{a_1 \epsilon R_1, \ldots, a_k \epsilon R_k} c_{ij}(a_1, \cdots, a_k) \cdot d_{ik}(a_1, \ldots, a_k)}{\Pr[X^k_{t_{k-1}}=i, \; Y_{k-1} \epsilon R_{k-1}, \ldots, Y_1 \epsilon R_1]}$$

where

$$c_{ij}(a_1, \ldots, a_k) = \Pr[X^k_{t_k}=j, Y_k=a_k | X^k_{t_{k-1}}=i, Y_{k-1}=a_{k-1}, \ldots, Y_1=a_1]$$

and

$$d_{ik}(a_1, \ldots, a_k) = \Pr[X^k_{t_{k-1}}=i, Y_{k-1}=a_k, \ldots, Y_1=a_1] \, .$$

Thus, by the assumption, $P_{V,k}(i,j)$ is equal to

$$\frac{\sum_{a_1 \epsilon R_1, \ldots, a_k \epsilon R_k} \Pr[X^k_{t_k}=j, Y_k=a_k | X^k_{t_{k-1}}=i] \cdot \Pr[X^k_{t_{k-1}}=i, Y_{k-1}=a_{k-1}, \ldots, Y_1=a_1]}{\Pr[X^k_{t_{k-1}}=i, Y_{k-1} \epsilon R_{k-1}, \ldots, Y_1 \epsilon R_1]} \, .$$

Factoring out the term $Pr[X^k_{t_k}=j, Y_k=a_k | X^k_{t_{k-1}}=i]$, we have

$$P_{V,k}(i,j) = Pr[X^k_{t_k}=j, Y_k=a_k | X^k_{t_{k-1}}=i] \cdot 1$$

$$= \bar{P}_{V,k}(i,j)$$

which completes the proof for part (1) of the theorem. Part(2) is proven in a like manner.

5.4 Operational Models as Intraphase Processes

To illustrate the application of the above general results, we consider in this section performability evaluation methods assuming that the phased base model of a phased model (see (5.3)) is Markovian and that the intraphase performance variable are defined as the minimum operational rates experienced by the system during a particular phase (see (3.23)). The representation of intraphase processes as operational models permits us to calculate the intraphase transition probabilities using Equations (3.45) and (3.49). By formulating the intraphase performance variables in terms of functionals, we also obtain a considerable gain in expressive power of a phased model, particularly in representing the effects of intraphase "repair." Moreover, since the operational models are allowed to vary from phase to phase, the modeling of a particular phase can be tailored to the computational requirements of the phase. In other words, this special class of phased models can be regarded as the time varying versions of the models introduced in Chapter 3.

5.4.1 Derivation of Intraphase Transition Probabilities

Recall that, in order to represent the phased base model as a one-parameter family of random variables

$$X = \{X_t | t \epsilon \hat{T}\} ,$$

the augmented utilization period is taken to be

$$\hat{T} = T \cup \{t_k' | k=1,2,...,m-1\}$$

where $t=[0,h]$ is the original utilization period. Hence, to describe X as a Markov process, $\hat{T}$ must be specified as a totally ordered set by inheriting the ordering relation of T and by assuming $t_k < t_k'$, $t_k' < x$, $y < t_k'$ for all $k=1,2,...,m-1$, and all $x,y \epsilon T$ such that $t_k < x$ and $y < t_k$. By Theorem 5.8, the Markov assumptions imply that the intraphase transition and the interphase transitions of X are memoryless. Hence, applying (5.28), for all Cartesian sets $V \subseteq A_1 \times \cdots \times A_m$,

$$Pr[V] = p \cdot [\prod_{k=1}^{m} H_k \cdot \bar{P}_{V,k}] \cdot F_m . \qquad (5.31)$$

In other words, the probability of V can be expressed in terms of the initial distribution of the first phase, $p=[p_1\ p_2\ \cdots\ p_{n_1}]$, the interphase transition

probabilities

$$H_k(i,j) = Pr[X^k_{t_{k-1}}=j | X^{k-1}_{t_{k-1}}=i]$$

where $k=1,2,...,m$, $i \epsilon Q_{k-1}$ and $j \epsilon Q_k$, and the intraphase transition probabilities

$$\bar{P}_{V,k}(i,j) = Pr[X^k_{t_k}=j, Y_k \epsilon R_k | X^k_{t_{k-1}}=i]$$

where $k=1,2,...,m$ and $i,j \epsilon Q_k$. Assuming p and $H_k(i,j)$ can be determined from the known properties of the system, then the problem of computing Pr[V] is reduced to that of computing the intraphase transition probabilities.

To compute the intraphase transition probabilities, we assume that each phase of the mission is represented by an operational model. More precisely, for each phase k, the intraphase process

$$X^k = \{X^k_t | t \epsilon T_k\}$$

is a time-homogeneous Markov process and the phase k performance variable Y_k is given by

$$Y_k = \min\{f_k(X_t) | t \epsilon T_k\}$$

where f_k is an operational structure defined on the state space of X^k. Clearly, the intraphase transition probability $\bar{P}_{V,k}(i,j)$ can be determined by

$$\Pr[X^k_{t_k}=j, Y_k=q | X^k_{t_{k-1}}=i] \qquad (q \epsilon Q_k)$$

which, in turn, can be determined by the conditional probabilities $m^q_{ij}(t)$ described in Section 3.2. More precisely, for each $1 \leq k \leq m$ and $q \epsilon \bar{Q}_k = \{1,2,...,n_k\}$ let us define a $n_k \times n_k$ dimensional matrix

$$M_{k,q} = [m^q_{ij}]$$

where for all $i,j \epsilon Q_k$,

$$m^q_{ij} = \Pr[X^k_{t_k}=j, Y_k=q | X_{t^k_{k-1}}=i] \ .$$

Then, since X^k is a time-homogeneous Markov process, m^q_{ij} can be computed by applying either (3.45) or (3.49). Moreover, it follows that the intraphase transition probabilities can then be obtained from $M_{k,q}$ by matrix additions, i.e.,

$$\bar{P}_{V,k}(i,j) = \sum_{q \epsilon R_k} M_{k,q} \ .$$

5.4.2 Evaluation of a Two-Phase Mission

In constructing a phased model that can support an evaluation of system performability, we must specify the intraphase processes and an organizing structure with respect to a specific computer and its computational environment. The intraphase processes together with the organizing structure determine an equivalent performability model that can be evaluated using solution methods developed in Section 5.3.

To illustrate these modeling and evaluation methods, a comprehensive phased model has been examined in [49], involving the SIFT computer with an environment taken to be the control of a transoceanic air transport mission. The model represents the internal structure of the SIFT computer as well as conditions of its environment in terms of Markov processes (see [49], Figure 3 and Table III). State trajectories of the equivalent base model are then related to accomplishment levels of the mission via a capability function (i.e., an organizing structure) which is formulated in terms of a three-level model hierarchy (see [49], Figure 2). After the capability function is formulated, solution methods are then applied to determine the performability of the total system.

Although the performability modeling and evaluation effort of the SIFT computer has shown the essential aspects of the phase model method, it has emphasized the construction of realistic higher level models. Simple Markovian models for nonrepairable systems are used to reduce the complexity

of the performability calculation. Hence, to show the solution algorithms in more detail, we consider in this subsection a phased model that uses operational models as intraphase processes.

We consider a total system S=(C,E) comprised of a control computer C operated in the environment E of a two-phase mission. The computer initially operates as a multiprocessor system with three identical components. However, system reconfiguration can occur in the computer due to phase change, hardware faults, or software faults. During the first phase $T_1=[t_0,t_1]$, all three subsystems are required to perform all computational tasks successfully. But, when less than three subsystems are available, the system can still survive by executing a reduced set of computational tasks. Depending on the amount of resources available, the system exhibits the following levels of accomplishment during phase 1:

Phase 1 accomplishment levels	Interpretation
3	Full performance
2	Noncritical performance degradation
1	Critical performance degradation
0	Failure

During the second phase $T_2=[t_1,t_2]$, the system is reconfigured into a TMR

system with software recovery to obtain a high degree of reliability (see Section 3.4). Hence, the system exhibits the following three accomplishment levels:

Phase 2 accomplishment levels	Interpretation
2	Full performance
1	Degraded performance
0	Failure

To describe aspects of the system performance that the users consider important, we assume that users are interested in distinguishing only three levels of mission performance $A=\{2,1,0\}$, where the accomplishment levels convey the following information:

Mission accomplishment levels	Interpretation
2	Full performance
1	Degraded performance
0	Failure

We further assume the following characteristics of the mission:

1) To achieve level 2 mission accomplishment, if the performance

degraded noncritically during the first phase, then performance degradation is not permitted during the second phase. If phase 1 performance is not degraded, then phase 2 performance is allowed to degrade.

2) Mission performance is degraded if phase 1 performance is noncritically degraded and phase 2 performance is degraded. A critically-degraded performance in phase 1 will result in level 1 mission accomplishment only if phase 2 performance is not degraded.

3) Mission accomplishment level is 0 if the system enters the failure mode during any phase.

Under the above assumptions, the organizing structure of the phased model (i.e., the capability function of the equivalent performability model) can be tabulated as in Table 5.1. Clearly, the organizing structure $\Psi:A_1 \times A_2 \rightarrow A$ is order-preserving and satisfies the conditions of Theorem 5.2.

Accomplishment Levels		
Phase 1	Phase 2	Mission (Ψ)
3	2	2
3	1	2
3	0	0
2	2	2
2	1	1
2	0	0
1	2	1
1	1	0
1	0	0
0	2	0
0	1	0
0	0	0

Table 5.1

An Organizing Structure

Hence, for each mission accomplishment level $a \epsilon A$, $\Psi^{-1}(a)$ can be expressed as a finite union of Cartesian subsets of $A_1 \times A_2$. In particular, one such representation is

$$\Psi^{-1}(2) = \{3\}\times\{1,2\} \cup \{2\}\times\{2\} \tag{5.32}$$

$$\Psi^{-1}(1) = \{2\}\times\{1\} \cup \{1\}\times\{2\} \tag{5.33}$$

$$\Psi^{-1}(0) = \{0\}\times\{0,1,2\} \cup \{0,1\}\times\{0,1\} \cup \{0\}\times\{0,1,2,3\} \tag{5.34}$$

Note that each Cartesian set in the above equalities is an interval as defined in Section 5.2.

To specify the equivalent base model, it is necessary that the state spaces of the intraphase processes be refined enough to support the evaluation of system performability. This condition can be satisfied if the phased base model (5.3) is chosen to be a time-homogeneous Markov process with a state space detailed enough to distinguish different operational modes of the intraphase processes. In other words, even though the operational models vary from phase to phase, they share the same underlying Markov process throughout the whole mission.

Assume that the computer has the same failure characteristics as the one considered in Section 3.4. Then, if we denote the model parameters by

λ = component failure rate

σ = software failure rate

c = hardware error coverage

d = software error coverage

$d\mu$ = rate of software recovery,

a common time-homogeneous Markov process for both phases can be specified as in Figure 5.2. Each state of the graph (except state 0) represents a specific number of subsystems that are free from hardware faults; a prime (′) is appended to the number if the system is attempting recovery from a software error. State 0 represents any other configurations. Using this Markov process, the probabilistic nature of phase 1 and phase 2 can be represented, respectively, by operational models induced by the operational structures as illustrated in Table 5.2. Note that the operational rates are chosen to convey the same information as the accomplishment levels of each phase so that the intraphase performance variables Y_k (k=1 or 2) can be defined as

$$Y_k = \min\{f_k(X_t) | t \epsilon T_k\}$$

where f_k is the operational structure of the k^{th} phase.

To compute the performability of the total systems, note also that the use of a common Markov process for all phases implies that each interphase transition matrix is an identity matrix. Accordingly, once the initial

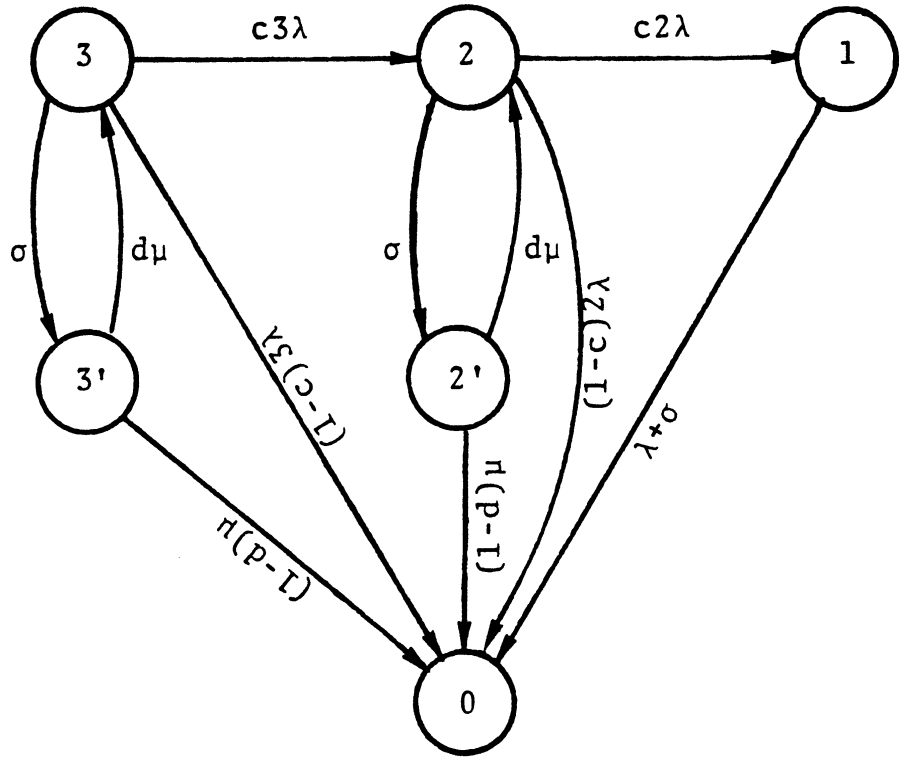


Figure 5.2

Underlying Markov Process

of a Phased Model

Phase	State	Operational rate
1	3	3
	3′	2
	2′	1
	1	1
	0	0
2	3	2
	3′	1
	2	2
	2′	0
	1	0
	0	0

Table 5.2

Operational Structures

of a Phased Model

distribution of the first phase model is known, the probabilities of Cartesian sets can be determined by repeatedly applying (5.31). First, let us assume the following parameter values:

Parameter	Value
λ	5×10^{-4}
σ	10^{-2}
μ	10^{3}
c	.99999
d	.9

Then, if we assume that the duration of the two phases are both 10 hours, we have, by (5.31),

$$M_{1,3} = \begin{bmatrix} .89137 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$M_{1,2} = \begin{bmatrix} .08393 & .01402 & .00001 & 0 & 0 & 0 \\ 0 & .89583 & 0 & 0 & 0 & 0 \\ .87777 & .01262 & .00001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$M_{1,1} = \begin{bmatrix} 0 & .00065 & 0 & .00001 & .00005 & 0 \\ 0 & .08436 & 0 & .00001 & .00005 & 0 \\ 0 & .00058 & 0 & 0 & .00846 & 0 \\ 0 & .88218 & 0 & .00001 & .00846 & 0 \\ 0 & 0 & 0 & 0 & .90032 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$M_{2,2} = \begin{bmatrix} .89137 & .01338 & 0 & 0 & 0 & 0 \\ 0 & .89583 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

and

$$M_{2,1} = \begin{bmatrix} .08393 & .00065 & .00001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ .87777 & .01262 & .00001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Accordingly, the trajectory sets in (5.32) and (5.33) can be evaluated by applying (5.28) and (5.18), i.e.,

$$\Pr[\Psi^{-1}(2)] = \Pr[\{3\}\times\{1,2\}] + \Pr[\{2\}\times\{2\}]$$

$$= p\cdot M_{1,3}\cdot(M_{2,1}+M_{2,2})\cdot F + p\cdot M_{1,2}\cdot M_{2,2}\cdot F$$

and

$$\Pr[\Psi^{-1}(1)] = \Pr[\{2\}\times\{1\}] + \Pr[\{1\}\times\{2\}]$$

$$= p\cdot M_{1,2}\cdot M_{2,1}\cdot F + p\cdot M_{1,1}\cdot M_{2,2}\cdot F$$

where p is the initial distribution of the first phase and F is a 6×1 dimensional matrix with a "1" on every entry. In particular, if the computer is fault-free at the beginning, i.e., p=[1 0 0 0 0 0], the performability of the phased mission is as follows:

$$\text{perf}(2) = \Pr[\Psi^{-1}(2)] = .97036$$

$$\text{perf}(1) = \Pr[\Psi^{-1}(1)] = .00769$$

$$\text{perf}(0) = \Pr[\Psi^{-1}(0)] = .02195 \quad .$$

CHAPTER 6

CONCLUSION AND FURTHER RESEARCH

The objective of this research has been to develop a general stochastic process model for evaluating the performability of degradable computing systems. This objective was established to fulfill the need of evaluating the unified performance and reliability of distributed multiprocessor systems. To accomplish the objective, a precise formulation of system performance is developed in a broad context and the concept is then applied to analyze the performance of degradable computing systems. Furthermore, a simple and useful user-oriented performance variable is identified and shown to be a proper generalization of the traditional notions of system performance and reliability.

In addition to the above modeling framework, a specific two-level hierarchical model is developed. The model is constructed according to a hierarchical decomposition of a system's behavior: Priority queueing models are used to analyze the system's detailed program behavior and the results are combined via a Markov reward process to characterize the overall system performance. Although the modeling approach resembles the top-down structured approach of software development, the decomposition considered

here is based on a more precise classification of a system's short term and long term equilibrium behavior. Accordingly, the modeling approach permits the evaluation of a computing system's hardware and software as a whole, and it becomes possible to deal with the performance and the reliability of a computing system simultaneously to measure the extent to which the user can benefit from tasks accomplished by the computer.

Finally, a time-varying version of the model is considered to analyze the performance of phased missions. By representing intraphase models in terms of operational models, we are able to obtain useful results even without the typical no-repair assumption of the traditional phased-mission reliability methods. Moreover, since the model considered does not require the structure function representation of system success, the approach thus represents an important generalization of traditional fault-tree analysis.

Although the investigation efforts documented in this thesis were carried to the point where the research objectives described in Section 1.2 were satisfactorily accomplished, there remain several problems that must be resolved before the performability modeling techniques can become a major tool in the design and analysis of computing systems.

First, to extend the usefulness of operational models, more efforts can be made to formulate various user-oriented performance variables that are suitable for a wide variety of computer applications. Since solution methods for these performance variables may differ considerably from those obtained in this

study, new solution techniques should also be explored.

Several generalizations of the phased model are possible depending on how the notion of phasing is relaxed. For example, by allowing the duration of each phase to vary, the model can be extended to a larger class of systems with time-varying environments. One may also extend the phased model by allowing the decision on selecting a succeeding phase to be made at the time of a phase change to improve the mission performance.

Another important problem that may have significant influence on the modeling of degradable computing systems is the modeling of software faults. The problem becomes even more interesting when both software and hardware faults are considered simultaneously. Note that the model considered in Chapter 4 measures the effect of hardware faults while taking into account the behavior of the system software. On the other hand, software reliability models (see [55]-[56], for example) are typically concerned with the effect of software faults on the system performance assuming that the hardware is fault-free. Clearly, useful performance measures can be obtained by combining performability with the results of software reliability analysis.

Finally, we note that the hierarchical decomposition method considered in Chapter 4 may also be extended to the performance modeling of computing systems in general in addition to the performance modeling of degradable computing systems. By classifying the physical and logical resources of a computing system according to their frequency of accesses, various models

can be constructed to facilitate the step-by-step approximation of system performance. To make the approach useful, however, it then becomes necessary to have a better understanding of the error bound of the approximation.

APPENDIX

MARKOVIAN FUNCTIONALS OF MARKOV PROCESSES

This appendix is reference material for Chapter 3. Conditions under which operational models become Markovian are stated in terms of slightly modified forms of what can be found in the literature. The utilization of such conditions together with their limitations are illustrated through examples.

When modeling computing systems as operational models, there are many situations in which the state space of the underlying base model may be much larger than needed to distinguish operational rates via the operational structure. Accordingly, to simplify the evaluation of system performability, one question that arises naturally is whether the operational models can be described as Markov processes and, if so, whether they are time-homogeneous. More precisely, let us suppose the total system is modeled by a time-homogeneous Markov process

$$X = \{X_s | 0 \leq s \leq t\}$$

with a denumerable state space Q and, relative to an operational structure $f:Q \rightarrow R$,

$$Z = \{f(X_s) | 0 \leq s \leq t\} .$$

Since the above question does not involve the actual values of f, Z may be regarded here as a "lumped" [52] version of X, where states i and j are in the same lump if and only if f(i)=f(j). In other words, lumps coincide with the operational modes of S. If f is 1-1 then Z is obviously both Markovian and time-homogeneous since, in this case the lumping is trivial. If i is properly a many-to-one function, the answer is no longer obvious, and indeed the question needs further clarification.

To begin, let us suppose that the underlying process X is specified by its generator matrix A and an initial distribution p. Then our original inquiry can be reduced to the following questions:

Q1) Given A, p and f, is Z Markovian?

In many applications, however, one wants the freedom to alter the initial distribution p without losing the Markov property. In this case we are asking:

Q2) Given A and f, is Z Markovian for arbitrary p?

Finally, we can raise our sights even higher and ask:

Q3) Given A and f, is Z Markovian for arbitrary p and,
moreover, is the transition function of Z
independent of p?

Adopting the terminology of [52] (which investigates the discrete-time versions of Q1 and Q3), if the answer to Q1 is "yes" then the process X specified by A and p is *weakly lumpable* (with respect to f). A "yes" answer to Q2 is stronger but, generally, these Markov processes will not be time-homogeneous. If the answer to Q3 is "yes" then, for all initial distributions p, the Markov processes Z have the same transition function and, by the homogeneity of X, it follows that this function is invariant under time shifts, i.e., the processes are time-homogeneous. In this case we say that the processes X specified by A are *strongly lumpable* (with respect to f).

Addressing first the question of weak lumpability (Q1), if Z is to be a Markov process, we must insure it has the "memoryless" property, that is, any sequence of past observations of Z provides the same information as the last of those observations. To formalize this requirement, if $0 \leq t_1 < t_2 < \cdots < t_k$ is a sequence of observation times and $q_i \epsilon \bar{Q} = f(Q)$ is the state of Z observed at time t_i, for each underlying state $j \epsilon Q$, let

$$M_j(t_1, \ldots, t_k; q_1, \ldots, q_k) = \Pr[X_{t_k}=j | Z_{t_1}=q_1, \ldots, Z_{t_k}=q_k] \qquad (A.1)$$

then the $1 \times |Q|$ matrix

$$M(t_1, \ldots, t_k; q_1, \ldots, q_k) = [M_j(\, ; \,)]$$

is the probability distribution of the states of X at time t_k, as conditioned by these observations of Z. In particular, since $Z_{t_k}=q_k$, it follows that $M_j(\ ;\ )$ is nonzero only if $f(j)=q_k$, where $\{M_j(\ ;\ )|f(j)=q_k\}$ gives the probability distribution of states inside the lump $f^{-1}(q_k)$. Moreover, since X is a Markov process, $M_j(\ ;\ )$ permits us to represent conditional probabilities in terms of the transition function of X, i.e., it can be shown that, for all $0\leq t_1<t_2<\cdots<t_k=t$ and $s\geq 0$,

$$M_j(t_1, \ldots, t_k; q_1, \ldots, q_k)\cdot P(s)$$

$$= [\pi_1\ \pi_2\ \cdots] \tag{A.2}$$

where, for each $i\epsilon Q$,

$$\pi_i = \Pr[X_{t+s}=i|Z_{t_1}=q_1, \ldots, Z_t=q_k]\ .$$

To translate distributions of X back up into distributions of Z relative to some specified ordering of the lumps, let Q_ℓ denote the ℓ^{th} lump, i.e., the collection of sets

$$\{Q_\ell|1\leq\ell\leq|\bar{Q}|\}$$

is the partition of Q induced by f. Accordingly, if $\pi=[\pi_1\ \pi_2\ \cdots]$ is a

probability distribution over the states of X, we let $\bar{\pi}$ denote the corresponding distribution, that is

$$\bar{\pi} = [\bar{\pi}_1 \ \bar{\pi}_2 \ \cdots] \tag{A.3}$$

where

$$\bar{\pi}_j = \sum_{i \epsilon Q_j} \pi_i \qquad (1 \leq j \leq |\bar{Q}|) \ .$$

In terms of the above notation, weak lumpability can then be characterized similar to its discrete time analog.

Theorem A.1:

Let X be a time-homogeneous Markov process with transition function P and a fixed initial state probability distribution p. Let Z be a functional of X and, for all $t \geq 0$, define

$$\Delta_p(t) = \{M(t_1,\ldots,t_k;q_1, \ldots, q_k) | 0 \leq t_1 < \cdots < t_k = t \text{ and } q_1, \ldots, q_k \epsilon \bar{Q}\} \ .$$

Then, Z is a Markov process if and only if, for all $s \geq 0$ and for all $\pi, \pi' \epsilon \Delta_p(t)$,

$$\bar{\pi} = \bar{\pi}' \quad \text{implies} \quad \overline{\pi \cdot P(s)} = \overline{\pi' \cdot P(s)} \ . \tag{A.4}$$

Theorem A.1 can be proved in a way similar to that of its discrete-time analog (see [52]; pp. 132-134). To illustrate its application, let us suppose that the system in question is a multicomputer comprised of three identical computer modules. Suppose further that modules fail independently and that each fails permanently with a constant failure rate λ. Then we can take the base model X to be the Markov process depicted by the state-transition-rate diagram of Figure A.1.

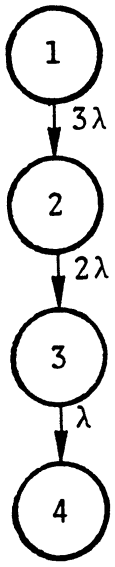


Figure A.1

Markov Model of a Multicomputer

As for operational rates, let us assume they are normalized so that, at full capacity the rate is 1, and with the loss of one or two modules the rate is 1/2; loss of a third module results in total failure. Accordingly, the operational structure here is the function

i	f(i)
1	1
2	1/2
3	1/2
4	0

and hence the functional Z takes values in the state set $\bar{Q}=\{1,1/2,0\}$. On taking the inverse of f, these states correspond as follows to lumps of Q:

$$1 \leftrightarrow \{1\}$$

$$1/2 \leftrightarrow \{2,3\}$$

$$0 \leftrightarrow \{4\}.$$

If we now examine the probabilistic nature of Z, we find that the conditional probability $Pr[Z_{t+s}=0|Z_t=1/2]$ depends on the time that Z enters state 1/2 from state 1 if the latter event is possible (i.e., if the probability of initially being in state 1 is nonzero). Thus, for example, if X is initially in state 1 with probability 1, i.e., p=[1 0 0 0] is the initial state probability distribution, then we have such a dependence (on the past history of Z) and therefore Z is not a Markov process. On the other hand, let us suppose the initial distribution is p=[0 0 1 0], which is not a likely choice from a functional point of view, but it serves to illustrate the role of p. In this case $\Delta_p(t)$ (as defined in the statement of Theorem A.1) is the same for any time t in T, i.e., it is the set

$$\Delta_p(t) = \{[0010],[0001]\} .$$

Accordingly, the conditions of Theorem A.1 are vacuously satisfied, and therefore Z is a Markov process for this choice of p. Moreover, it should be obvious that Z, is this case, is time-homogeneous. Other distributions, such as p=[0 1 0 0] can be shown to result in Markov processes that are not time-homogeneous.

Regarding the second question (Q2), a necessary and sufficient condition can be obtained by extending the previous theorem to arbitrary initial distributions. More precisely, it can be shown that Z is a Markov process whatever the initial distribution if and only if, for all $t,s \geq 0$ and any initial distribution p, condition (A.4) holds for all π, π' in $\Delta_p(t)$. Although the characterization is useful from a conceptual point of view, it is difficult to use in practical applications. A more desirable form of this result can be found in [53] (pp. 1113-1114, Theorem 4) which assumes that X has a finite state space. (The theorem was generalized later in [54] to allow for arbitrary state space.) Stating the desired form of this result in terms of the notation defined above we have:

Theorem A.2:

Let X be a time-homogeneous Markov process with generator matrix $A=[a_{ij}]$ and let Z be a functional of X determined by f. Then Z is a Markov process, whatever the initial distribution of X, if and only if for each

$q \epsilon \bar{Q}$ taken separately either

(i) For all $i,j \epsilon Q$ such that $f(i) \neq q$ and $f(j) = q$,

$$a_{ij} = 0$$

or

(ii) For all $r \epsilon \bar{Q}$ such that $r \neq q$, the sum

$$\sum_{f(j)=r} a_{ij}$$

is the same for all $i \epsilon Q$ such that $f(i) = q$.

Although the conditions of Theorem A.2 guarantee that Z is a Markov process relative to any initial distribution p for X, note that the specific nature of Z (as specified by its transition function) will generally depend on p. Moreover, the process Z need not be time-homogeneous.

To illustrate Theorem A.2 and the above observations, let us again consider the Markov process X having the state-transition-rate diagram given by Figure A.1. Then, the generator matrix of X is the 4×4 matrix

$$A = \begin{bmatrix} -3\lambda & 3\lambda & 0 & 0 \\ 0 & -2\lambda & 2\lambda & 0 \\ 0 & 0 & -\lambda & \lambda \\ 0 & 0 & 0 & 0 \end{bmatrix} .$$

Suppose, however, that the operational structure here is one that corresponds to triplication with voting (TMR), i.e., the function

i	f(i)
1	1
2	1
3	0
4	0

Then $\bar{Q}=\{1,0\}$ and, applying Theorem A.2, we see that state 1 (i.e., lump $\{1,2\}$) satisfies condition (i) and state 0 (i.e., lump $\{3,4\}$) satisfies condition (ii). Hence, the functional Z is a Markov process. To determine the probabilistic nature of Z, let us rename state 0 (in $\bar{Q}$) as state 2 (permitting the use of standard matrix notation) and let $\bar{P}(s,t)$ denote the transition function of Z, i.e.,

$$\bar{P}(s,t) = [\bar{p}_{ij}(s,t)] \qquad (s \leq t)$$

where

$$\bar{p}_{ij}(s,t) = Pr[Z_t=j | Z_s=i] \ .$$

Then, relative to an initial distribution $p=[p_1\ p_2\ p_3\ p_4]$ for X, if we let

$$d = \frac{p_1}{p_1+p_2} ,$$

it can be shown that the matrix $\bar{P}(s,t)$ has the following entries:

$$\begin{aligned}
\bar{p}_{11}(s,t) &= \frac{e^{-2\lambda(t-s)}(1+2d(1-e^{-\lambda t}))}{(1-2d(1-e^{-\lambda s}))} , \\
\bar{p}_{12}(s,t) &= 1 - \bar{p}_{11}(s,t) , \\
\bar{p}_{21}(s,t) &= 0 , \\
\bar{p}_{22}(s,t) &= 1 .
\end{aligned} \tag{A.5}$$

From the above equations, we see that the transition function $\bar{P}(s,t)$ depends on d and, hence, on the initial distribution $p=[p_1\ p_2\ p_3\ p_4]$. Moreover, we observe that Z is time-homogeneous (i.e., the values of $\bar{P}(s,t)$ depend only on the time difference t-s) only when $d=0$. In other words, by the definition of d, Z is time-homogeneous if and only if $p_1=0$, i.e., there is a zero probability that the underlying process X is initially in state 1 (all three modules fault-free). However, with our interpretation of Z as a TMR model, this special case is pathological, and hence for most practical purposes Z will not be time-homogeneous.

Finally, turning to the question of strong lumpability (see Q3 above), the answer can be characterized by removing condition (i) of Theorem A.2 and modifying the proof to accommodate this change. More precisely, we have

Theorem A.3:

Let X be a time-homogeneous Markov process with generator matrix $A=[a_{ij}]$ and let Z be a functional of X determined by f. Then Z is a Markov process, whatever the initial distribution p of X and with a transition function that is independent of p, if and only if for each $q \epsilon \bar{Q}$ the following is satisfied:

For all $r \epsilon \bar{Q}$ such that $r \neq q$, the sum

$$\sum_{f(j)=r} a_{ij} \tag{A.6}$$

is the same for all $i \epsilon Q$ such that $f(i)=q$. To illustrate Theorem A.3, suppose X is specified by the generator matrix

$$A = \begin{bmatrix} -3\lambda & \lambda & \lambda & \lambda & 0 & 0 & 0 \\ 0 & -2\lambda & 0 & 0 & \lambda & \lambda & 0 \\ 0 & 0 & -2\lambda & 0 & \lambda & 0 & \lambda \\ 0 & 0 & 0 & -2\lambda & 0 & \lambda & \lambda \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.7}$$

and f is the function

q	f(q)
1	1
2	2
3	3
4	3
5	4
6	4
7	4

.

Testing condition (A.6) for states 1 and 2 in $\bar{Q}$, we see that it holds trivially since these states correspond to singleton lumps. As for state $3\epsilon\bar{Q}$ (corresponding to lump {3,4}), with respect to states $1,2\epsilon\bar{Q}$ the sums are zero for both i=3 and i=4; with respect to state $4\epsilon\bar{Q}$ the sum is 2λ for both i=3 and i=4. Thus condition (A.6) holds for state 3. Finally, (A.6) is likewise satisfied for state $4\epsilon\bar{Q}$ and we conclude that Z is a Markov process with a transition function that is independent of p.

In general, if X is strongly lumpable (as characterized by Theorem A.3) it is easily shown that Z must inherit the time-homogeneity of X. In other words, a strongly lumped processes will always be time-homogeneous and, accordingly, it can be specified by a constant generator matrix. More

precisely, let us rename the states in $\bar{Q}$ (if not already so named) with the integers from 1 to $|\bar{Q}|$, the generator matrix $\bar{A}=[\bar{a}_{qr}]$ of Z can be constructed directly from A, where entry $\bar{a}_{qr}$ $(q \neq r)$ is given by the invariant sum of condition (A.6) for any i such that $f(i)=q$. (The diagonal entries $\bar{a}_{qq}$ are then determined by the condition that rows must sum to zero.) Thus, for the example just considered (see (A.7) and (A.8)), the generator matrix of Z is the 4X4 matrix

$$A = \begin{bmatrix} -3\lambda & \lambda & 2\lambda & 0 \\ 0 & -2\lambda & 0 & 2\lambda \\ 0 & 0 & -2\lambda & 2\lambda \\ 0 & 0 & 0 & 0 \end{bmatrix} .$$

As illustrated through the above examples, a strongly lumped process is clearly the most desirable type of operational model. On the other hand, by Theorem A.3, it is evident that such models require a relatively restricted "match" between the probabilistic nature of the base model (as specified by A) and the operational structure f.

The conditions of Theorem A.2 are somewhat weaker although, when satisfied, the transition rates of the resulting Markov functional are generally time-varying and dependent on the initial distribution of the underlying process. Of significance here is that even without strong lumpability one can obtain operational models that are Markovian and admit to feasible, closed-form analytic solutions (see Equations A.5, for example). What must be

used here, then, are solution techniques for arbitrary (discrete-state) Markov processes, as opposed to more special (and much more familiar) techniques that apply only to time-homogeneous Markov processes.

Finally, regarding weak lumpability (Theorem A.1), the requirement here is even less restrictive. However, depending on A, p, and f, it may be difficult to decide whether the condition of Theorem A.1 is satisfied. Moreover, we currently know of no general means of solving such models without resorting to detailed computations at the base model level. The utility of weak lumpability is also curtailed by the fact that the initial state distribution of the base model is fixed. This may be satisfactory in certain applications but one often wishes to examine the influence of different initial distributions. In such cases, one must derive a solution for each of the given distributions provided, of course, that each admits to weak lumpability.

Theorems A.1-A.3 thus provide formal support of what we and others in the field have observed through experience: at higher, more user-oriented levels of abstraction, it is difficult to maintain a Markovian representation of system behavior. As a consequence, we should seek means for accommodating operational models (functionals) that are not Markovian. The latter task is less formidable than it might appear if we bear in mind that, when evaluating a system S, an operational model Z plays an intermediate role in support of a specific performance variable Y. Thus, our knowledge of Z can be restricted to that required to solve the probability distribution of Y, i.e., the

performability of S. The latter observation serves as the guiding principle for the work described in Chapter 3.

REFERENCES

References

[1] J. F. Meyer, "On evaluating the performability of degradable computing systems," *Proc. 1978 Int'l Symp. on Fault-Tolerant Computing,* Toulouse, France, pp. 44-49, June 1978.

[2] J. F. Meyer, "On evaluating the performability of degradable computing systems," *IEEE Trans. Comput.*, vol. C-29, no. 8, pp. 720-731, Aug. 1980.

[3] D. Ferrari. *Computer Systems Performance Evaluation.* Englewood Cliffs, NJ: Prentice-Hall, 1978.

[4] H. Kobayashi, *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology.* Reading, MA: Addison-Wesley, 1978.

[5] W. G. Bouricius, W. C. Carter, and P. R. Schneider, "Reliability modeling techniques for self-repairing computing systems," *Proc. 24th ACM National Conference,* pp. 295-305, Aug. 1969.

[6] F. P. Mathur, "On reliability modeling and analysis of ultra-reliable fault-tolerant digital systems," *IEEE Trans. Comput.*, vol. C-20, no. 11, pp. 1376-1382, Nov. 1971.

[7] Y.-W. Ng and A. Avizienis, "A reliability model for gracefully degrading and repairable fault-tolerant systems," *Proc. 1977 Int'l Symp. on Fault-*

Tolerant Computing, Los Angeles, CA, pp. 22-28, June 1978.

[8] 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, no. 6, pp. 548-560, June 1978.

[9] M. D. Beaudry, "Performance related reliability measures for computing systems," *Proc. 1977 Int'l Symp. on Fault-Tolerant Computing,* Los Angeles, CA, pp. 16-21, June 1977.

[10] H. Mine and K. Hatayama, "Performance evaluation of a fault-tolerant computing system," *Proc. 1979 Int'l Symp. on Fault-Tolerant Computing,* Madison, WI, pp. 59-62, June 1979.

[11] R. S. Howard, *Dynamic Probabilistic Systems, Vol. II: Semi-Markov and Decision Processes.* New York, NY: John Wiley, 1971.

[12] F. A. Gay and M. L. Ketelsen, "Performance evaluation for gracefully degrading systems," *Proc. 1979 Int'l Symp. on Fault-Tolerant Computing,* Madison, WI, pp. 51-58, June 1979.

[13] J. M. De Souza, "A unified method for the benefit analysis of fault-tolerance," *Proc. 1980 Int'l Symp. on Fault-Tolerant Computing,* Kyoto, Japan, pp. 201-203, Oct. 1980.

[14] X. Castillo and D. P. Siewiorek, "A performance-reliability model for computing systems," *Proc. 1980 Int'l Symp. on Fault-Tolerant Computing,* Kyoto, Japan, pp. 187-192, Oct. 1980.

[15] W. B. Davenport, Jr., *Probability and Random process.* New York, NY: McGraw-Hill, 1970.

[16] E. Wong, *Stochastic Processes in Information and Dynamical Systems.* New York, NY: McGraw-Hill, 1971.

[17] J. L. Doob, *Stochastic Processes.* New York, NY: John Wiley, 1953.

[18] R. E. Barlow and F. Proschan, *Statistical Theory of Reliability and Life Testing.* New York, NY: Holt, Rhinehart and Winston, 1975.

[19] J. F. Meyer, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Semiannual Status Rep. 3, NASA Rep. CR158992 (NTIS Rep. N79-17564/2GA), Jan. 1978.

[20] J. F. Meyer, "Models and techniques for evaluating the effectiveness of aircraft computing systems," Semiannual Status Rep. 4, NASA Rep. CR158993 (NTIS Rep. N79-17563/4GA), July 1978.

[21] L. Kleinrock, *Queuing Systems, Volume II: Computer Applications.* New York, NY: John Wiley, 1976.

[22] K. L. Chung, *Markov Chains with Stationary Transition Probabilities.* Berlin: Springer-Verlag, 1960.

[23] E. Cinlar, *Introduction to Stochastic Processes.* Englewood Cliffs, NJ: Prentice-Hall, 1975.

[24] Discussions with F. J. Beutler.

[25] 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, no. 10, pp. 1221-1239, Oct. 1978.

[26] A. Avizienis and L. Chen, "On the implementation of N-version programming for software fault-tolerance during program execution," *Proc. 1977 COMPSAC (Int'l Computer Software and Applications Conf.),* Chicago, IL, pp. 149-155, Nov. 1977.

[27] T. F. Arnold, "The concept of coverage and its effect on the reliability model of a repairable system, *IEEE Trans. Comput.*, vol. C-22, no. 3, pp. 251-254, March 1973.

[28] C. Weitzman, *Distributed Micro/Minicomputer Systems.* Englewood Cliffs, NJ: Prentice-Hall, 1980.

[29] D. Katsuki, E. S. Elsam, W. F. Mann, E. S. Roberts, J. C. Robinson, F. S. Skowronski, and E. W. Wolf, "Pluribus - An operational fault-tolerant multiprocessor," *Proc. IEEE,* vol. 66, no. 10, pp. 1146-1159, Oct. 1978.

[30] J. H. Wensley, L. Lamport, J. Goldberg, M. W. Green, K. N. Levitt, P. M. Melliar-Smith, R. E. Shostak, and C. B. Wenstock, "SIFT: Design and analysis of a fault-tolerant computer for aircraft control," *Proc. IEEE,* vol. 66, no. 10, pp. 1240-1255, Oct. 1978.

[31] A. Avizienis, "Fault-tolerance: The survival attribute of digital systems," *Proc. IEEE,* vol. 66, no. 10, pp. 1109-1125, Oct. 1978.

[32] H. Hecht, "Fault-tolerant software for real-time applications," *ACM Computing Surveys,* vol. 8, no. 4, pp. 391-407, Dec. 1976.

[33] D. P. Siewiorek, V. Kini, H. Mashburn, S. McConnel, and M. Tsao, "A case study of C.mmp, Cm* and C.vmp: Part I - Experiences with fault tolerance in multiprocessor systems," *Proc. IEEE,* vol, 66, no. 10, pp. 1178-1199, Oct. 1978.

[34] H. B. Baskin, B. R. Borgerson, and R. Roberts, "PRIME - A modular architecture for terminal-oriented systems," *1972 Spring Joint Comput. Conf., AFIPS Conf. Proc. Vol. 40.* Washington, DC: Sparton, pp. 431-437.

[35] J. J. Shedletsky and E. J. McCluskey, "The error latency of a fault in a sequential digital circuit," *IEEE Trans. Comput.,* vol. C-25, no.6, pp. 655-659, June 1976.

[36] B. R. Borgerson and R. F. Fretitas, "A reliability model for gracefully degrading and standby-sparing systems," *IEEE Trans. Comput.,* vol. C-24, no.5, pp. 517-525, May 1975.

[37] J. Losq, "Effect of failure on gracefully degradable systems," *Proc. 1977 Int'l Symp. on Fault-Tolerant Computing,* Los Angeles, CA, pp. 29-34, June 1977.

[38] J. C. Laprie, "Reliability and availability of repairable structures," *Proc. 1975 Int'l Symp. on Fault-Tolerant Computing,* Paris, France, pp. 87-92,

June 1975.

[39] R. S. Ratner, E. B. Shapiro, H. M. Zeidler, S. E. Wahlstrom, C. B. Clark, and J. Goldberg, "Design of a fault tolerant airborne digital computer, Volume II - Computational Requirements and Technology," NASA Contract NAS1-10920, Stanford Research Institute, Menlo Park, CA, Oct. 1973.

[40] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory.* New York, NY: John Wiley, 1974.

[41] W. N. Toy, "Fault-tolerant design of local ESS processors," *Proc. IEEE.* vol. 66, no. 10, pp. 1126-1145, Oct. 1978.

[42] H. S. Winokur, Jr., and L. J. Goldstein, "Analysis of mission-oriented systems," *IEEE Trans. Reliability,* vol. R-18, no. 4, pp. 144-148, Nov. 1969.

[43] J. L. Bricker, "A unified method for analyzing mission reliability for fault tolerant computer systems," *IEEE Trans. Reliability,* vol. R-22, no. 2, pp. 72-77, June 1973.

[44] J. D. Esary and H. Ziehms, "Reliability analysis of phased missions," *Reliability and Fault-Tree Analysis,* Philadelphia, PA: SIAM, pp. 213-236, 1975.

[45] G. R. Burdick, J. B. Fussell, D. M. Rasmuson, and J. R. Wilson, "Phased mission analysis: A review of new developments and applications," *IEEE*

Trans. Reliability, vol. R-26, no. 1, pp. 43-49, April 1977.

[46] R. A. Ballance and J. F. Meyer, "Functional dependence and its application to system evaluation," *Proc. 1978 Conf. on Information Sciences and Systems,* The Johns Hopkins University, Baltimore, MD, March 1978.

[47] Z. W. Birnbaum, J. D. Esary and S. C. Saunders, "Multicomponent systems and structures and their reliability," *Technometrics,* vol. 3, no. 1, pp. 55-77, Feb. 1961.

[48] G. Birkhoff, *Lattice Theory.* Providence, RI: American Math. Society, 1966.

[49] J. F. Meyer, D. G. Furchtgott, and L. T. Wu, "Performability evaluation of the SIFT computer," *IEEE Trans. Comput.,* vol. C-29, no.6, pp. 501-509, June 1980.

[50] G. Gratzer, *General Lattice Theory.* Stuttgart, Germany: Birkhauser, 1978.

[51] W. Feller, *An Introduction to Probability Theory and Its Applications, Volume I.* New York, NY: John Wiley, 1957.

[52] J. G. Kemeny and J. L. Snell, *Finite Markov Chains.* Princeton, NJ: Van Nostrand, 1960.

[53] C. J. Burke and M. Rosenblatt, "A Markov function of a Markov chain," *Annals of Math. Statistics,* vol. 29, pp. 1112-1122, 1958.

[54] M. Rosenblatt, *Markov Processes. Structure and Asymptotic Behavior.* New York, NY: Springer-Verlag, 1971.

[55] J. D. Musa, "A theory of software reliability and its application," *IEEE Trans. Software Engineering,* vol. SE-1, no. 3, Sept. 1975.

[56] B. Littlewood, "A semi-Markov model for software reliability with failure costs," *Proc. Symp. Comput. Software Eng.,* New York, NY, pp. 281-300, April 1976.