Performance Evaluation 14 (1992) 139–156
North-Holland

Performability: a retrospective and some pointers to the future *

John F. Meyer

Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109, USA

Abstract

Meyer, J.F., Performability: a retrospective and some pointers to the future, Performance Evaluation 14 (1992) 139–156.

As computing and communication systems become physically and logically more complex, their evaluation calls for continued innovation with regard to measure definition, model construction/solution, and tool development. In particular, the performance of such systems is often degradable, i.e., internal or external faults can reduce the quality of a delivered service even though that service, according to its specification, remains proper (failure-free). The need to accommodate this property, using model-based evaluation methods, was the *raison d'être* for the concept of performability. To set the stage for additional progress in its development, we present a retrospective of associated theory, techniques, and applications resulting from work in this area over the past decade and a half. Based on what has been learned, some pointers are made to future directions which might further enhance the effectiveness of these methods and broaden their scope of applicability.

Keywords: performability, degradable performance, dependability, fault tolerance, model-based evaluation.

1. Introduction

Contemporary computers and communication systems represent a fusion of concepts, techniques, and technologies from the fields of both computing and communication. It is not surprising, therefore, that means of evaluating such systems have likewise evolved from methods originating in one field or the other, merged in various ways to satisfy differing evaluation needs. As these systems become more physically and logically complex, challenges regarding their evaluation fail to subside, calling for continued innovation in areas of measure definition, model construction/solution, and tool development. This is especially so for networked systems, ranging from local area networks (LANs) to the type integrated broadband communication networks (IBCNs) envisioned for next-generation telecommunication systems.

Generally, when evaluating a system, one seeks to relate and quantify aspects of what the system is and does with respect to what the system is required to be and do. Moreover, since what a system does (e.g., how well it performs) depends on what it is (e.g., how its resources are altered by faults), both need to be addressed in the evaluation process. Just what is evaluated or, more precisely, the types of measures employed can be classified according to certain assumptions regarding "is" and "does". In the context of computing systems and with respect to a specified user-oriented or system-oriented service, *performance* typically refers to "quality of service, provided the system is correct." *Dependability* (according to current use of this term; see [6,59], for example) is that property of a system which allows "reliance to be justifiably placed on the the service it delivers." Such service is *proper* if it is delivered as specified; otherwise it is *improper*. System *failure* is identified with a transition from proper service to improper service. Dependability thus includes attributes of reliability and availability as special cases. Specifically, a reliability measure quantifies the "continuous delivery of proper service"; an availability measure quanti-

* This work was supported in part by the Office of Naval Research under Contract No. N00014-85-K-0531.

0166-5316/92/$05.00 © 1992 – Elsevier Science Publishers B.V. All rights reserved

fies the "alternation between deliveries of proper and improper service."

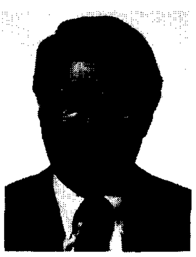
This basic distinction between performance and dependability [1] has been particularly useful in development of evaluation techniques suited to each concept. As a consequence, both performance evaluation and dependability evaluation have evolved as important technical disciplines within the fields of computing and communication. However, if separate evaluations of system performance and system dependability are to suffice in determining overall "quality of service" (e.g., QOS, as this term is defined and abbreviated by the telecommunication industry; see [48], for example), one must place certain constraints on how properties affecting performance interact with those affecting dependability. For example, suppose that a system's capacity to serve is binary (either "up" or "down") and proper service (see above) coincides with that delivered when the system is up. In this case, performance measures the quality of proper service and dependability measures the system's ability to remain up (and thus deliver that service) in the presence of faults. Accordingly, results of each type of evaluation, when taken together, can provide a rather complete assessment of overall service quality.

Generally, however, individual assessments of system performance and dependability are not so easily combined, particularly if performance in the presence of faults is *degradable*, i.e., fault-caused errors can reduce the quality of a delivered service even though that service, according to its specification, remains proper (failure-free). The need to accommodate this property, using model-based evaluation methods, was the *raison d'être* for the concept of performability. As a prelude to the body of the paper, a more precisely stated example of degradable performance is described in Section 2. It is followed by a retrospective (Section 3) of theory, techniques, and applications that have evolved from work on model-based performability evaluation over the past 15 years. Based on what has been learned, some pointers are made to future directions (Section 4) which might further enhance the effectiveness of these methods and broaden their scope of applicability.

[1] The term "reliability" often has a second, more generic meaning that is similar to the definition of "dependability". However, in the presentation that follows, we prefer to consistently employ the latter term, even when referring to past work (prior to the mid-1980s) that preceded its current use.

2. Degradable performance

What separate evaluations of performance and dependability may fail to provide can be explained more carefully via a specific, yet representative example. To begin, we assume (as is usual in model-based evaluation) that measures of performance and dependability are defined in terms of random variables representing specific aspects of what is to be evaluated. In particular, let us suppose that the system in question is a telephone switching network and Y_t is a random variable that represents some aspect of service quality, as observed up to time t during some designated period T. For example, $T = [s, u]$ is a "busy period" and, for any time $t (s < t \leqslant u)$, Y_t is the fraction of incoming calls that have been successfully processed since start-time s. Accordingly, if we let Y denote overall service quality then $Y = Y_u$, the fraction of calls successfully pro-



John F. Meyer is a Professor in the Department of Electrical Engineering and Computer Science at the University of Michigan in Ann Arbor, MI, USA. He has been active in computer and systems research for over 30 years and has published widely in the areas of fault-tolerant computing and system evaluation. He joined the Michigan faculty in 1967 and, in addition to his university appointment, has held visiting research positions at laboratories in the USA, England, France, Italy, and Japan. Prior to 1967, he was a Research Engineer at the California Institute of Technology Jet Propulsion Laboratory where his contributions included the first patent issued to the National Aeronautics and Space Administration. His current research interests concern the development and application of probabilistic models for evaluating the performability (performance-dependability) of distributed computing and communication systems. He is a Fellow of the IEEE, a member of the ACM, and has served the IEEE Computer Society in various capacities, including chair of the Technical Committee on Fault-Tolerant Computing and membership on the Society's Board of Governors. He is currently a vice-chair of IFIP Working Group 10.4 on Dependable Computing and Fault Tolerance.

cessed throughout the busy period T. Regarding service loss, let us suppose that failure is defined relative to some some threshold of service quality $\sigma (0 < \sigma \leqslant 1)$, i.e., at any time $t \in (s, u]$, network service is proper if $Y_t \geqslant \sigma$; otherwise it is improper, hence, and the network has failed (the time of failure being the instant t that that value of Y_t last dropped below σ). Given the general notions of performance and dependability discussed in the introduction, let us now examine how each might be measured for the example just described.

Regarding performance, we must first agree on what is meant by a "correct" system, since some call losses may be due strictly to congestion (e.g., blocking), some strictly to breakdowns in system resources, and some to a combination of both (and perhaps other) factors. A typical choice here is to regard the system as being correct if its structure is correct, i.e., its hardware and software resources are fault-free. However, unless a network is designed to be non-blocking, blocking losses can occur in the absence of structural faults and, hence, are an inherent part of correct network behavior. Accordingly, the performance in question is the random variable Y_P, where Y_P is Y, conditioned by the event that the system remains fault-free throughout the busy period T. A complete measure of Y_P is given by its probability distribution function. In practice, however, one might settle for a less refined measure such as the expected value $E[Y_P]$.

As for dependability, let us suppose that continued proper service throughout T is the user's concern. Then, according to our assumed meaning of what constitutes proper service, network dependability can be expressed by the indicator random variable Y_D, where $Y_D = 1$ if $Y_t \geqslant \sigma$, for all $t \in (s, u]$; $Y_D = 0$ otherwise. The corresponding dependability measure is the usual measure of reliability, i.e., the probability that $Y_D = 1$ or, equivalently, the probability of no failures occurring during the busy period T. Note, however, that a reliability evaluation in this instance must address more than just the effects of structure faults, since call blocking, for example, can likewise contribute to a failure.

Given these measures of performance and dependability, separate evaluations of each may yield only a partial assessment of service quality. In particular, suppose there are structure faults which reduce the quality of service without causing failure, i.e., the network's performance is degradable. Then, obviously, such degradation cannot be accounted for by the performance variable Y_P, since its measure is conditioned by the event that the network is correct (fault-free) throughout T. Likewise, it is not accounted for by the dependability variable Y_D since the latter concerns service quality only to extent that it is proper (failure-free) throughout T.

Generally, if a system exhibits degradable performance (in which case the system itself is often referred to as being "degradable" or "gracefully degrading"), a binary (up-down) classification of operational integrity is too coarse. Instead, a degradable system's operational integrity should be viewed as a multivalued variable representing the extent to which the system is faulty, e.g., which resources are faulty and, among them, which ones have failed, which ones are in the process of fault recovery, which ones contain latent faults, etc. With such variations, the usual concept of computer performance is too restrictive and, although dependability measures are compatible with this view (indeed, the special nature of degradable computing systems was first investigated in a reliability context [14]), they account for service quality degradation only at the boundary between proper and improper service.

As noted in the concluding paragraph of Section 1, the need to fill this gap, in a manner suited to model-based methods (either analysis or simulation), was the principal reason for introducing the concept of performability. The section that follows reviews associated theory, techniques, and applications that have resulted from work in this area since the mid-1970s.

3. A retrospective

A general framework for model-based performability evaluation was first published in 1978 [70], with a somewhat more refined description appearing in 1980 [71]. This framework grew from some ideas concerning "partial success" which had been formulated, but not openly published, a number of years before [65], and from a notion of "computation-based reliability" which was examined in the mid-1970s [66,68]. It was also motivated by the recognition, inherent in work done at that time by Borgerson and Freitas [14], that degradable systems required special attention with regard to the kind of measures and models that might be used in their evaluation.

The application aim, at the outset, was the evaluation of ultra-reliable aircraft control computers being developed for the U.S. Space Agency (NASA) by both the C.S. Draper Laboratory [44] and SRI International [103]. One intended feature of these systems was an ability to shed workload, beginning with the least critical tasks, if a loss of computing resources, due to faults, demanded it. Accordingly, these systems could provide varying degrees of service over a specified period of use (e.g., the duration of a flight of the aircraft) and, hence, exhibited the type of degradable performance referred to in the previous section.

Prior to settling on the concepts and terminology introduced in [70], we initially viewed the unification of performance and dependability (or reliability, as it was referred to then; see footnote 1) as a measure of system "effectiveness" [67,69], where its formulation depended on an intermediate association of "worth" (reward, benefit, utility) with each possible level of accomplishment. (This view was consistent with the definition proposed in [65] but differed with respect to lower-level details.) However, as the desired amount of generality became clearer, it was decided, in late 1976, that performance-dependability aspects of effectiveness should be separated from the worths that one might associate with their outcomes. The resulting concept was more refined and, consequently, could still be employed in higher-level, worth-oriented evaluations of system effectiveness. In words, it measured an object system's "ability to perform" in a designated environment, whence the term "performability" which we adopted at that time.

In a more formal probability-theoretic setting, performability and its associated concepts are defined as follows [70,71]. Let S denote the *total system* in question where, generally, S consists of an *object system* C (the computing or communication system being evaluated) and its *environment* E (workload, external faults, etc.). Then the *performance* of S over a specified *utilization period* T is a random variable Y taking values in a set A; elements of A are the *accomplishment levels* (performance outcomes) that might possibly be attained by S. T is the time period of use over which system performance is summarized (by the value of Y). Formally, T is an interval of numbers (time instants) that is either continuous or discrete and either bounded (e.g, the busy period $[s, u]$ considered for the switching network example in the previous section) or, for systems which exhibit meaningful steady-state behavior, unbounded from above (e.g, $T=[0,\infty)$ or $T=\{0,1,2,\dots\}$).

Note that the interpretation of "performance" here, as compared with its traditional use in a computing context (see Section 1) is more general. It connotes any designated aspect of total system behavior relative to which the object system's ability to perform is being measured. Accordingly, choices of Y are virtually limitless, ranging from a high-level representation of service quality with a continuum of accomplishment levels, down to a binary-valued variable that distinguishes whether or not a specified service is performed properly throughout T (e.g., the dependability variable Y_{D} illustrated in the previous section). Accordingly, performance, in this framework, has the generic meaning of "what a system accomplishes during its use"; its ability to so perform, expressed by probabilities, is its performability. As a measure, the latter can be generally defined as follows.

For a system S with performance Y taking values in accomplishment set A, the *performability* of S is the probability measure *Perf* (denoted p_s in [70,71]) induced by Y where, for any measurable set B of accomplishment levels ($B \subseteq A$),

$$Perf(B) = \mathrm{P}[Y \in B] = \text{the probability that } S \text{ performs at a level in } B.$$

Although, conceptually, this measure applies to any set B for which the event "$Y \in B$" has a probability, in practice these sets are typically intervals of accomplishment expressing performance requirements. Thus, for example, if $A=(-\infty,\infty)$ and $B=[a,\infty)$ then *Perf(B)* is the probability that S performs at or above level a.

Determination of these probabilities is based on an underlying stochastic process $X=\{X_t \mid t \in I\}$, where the index (time) set I must include the utilization period T (i.e., $T \subseteq I$) associated with the performance variable Y. Thus X may be continuous-time or discrete-time, depending on the nature of just what is being evaluated. X is referred to as a *base model* [2] of S where, for any $t \in I$, the value of the random variable X_t is the state of the total system S at time t. Hence, when restricted to the period T associated with Y, this process conveys the dynamics of an object system's structure, internal state, and environment during that period. By its definition, the base model must also "support" a solution of performability in the sense that, for any accomplishment set B of interest, *Perf(B)* is indeed determinable, at least theoretically, from the probabilistic nature of X restricted to T. This is insured via the concept of a *capability function* which maps trajectories of X into corresponding values of Y. A base model X together with a performance variable Y is a *performability model* of S. When a performability model is solved analytically, the base model must be characterized explicitly in some suitable form, e.g., a state-transition-rate matrix in the case of a continuous-time, time-homogeneous, finite-state Markov process. If performability is estimated via simulation techniques then X refers to the behavior of a some simulation model of S.

In general terms, model-based performability evaluation thus involves (1) construction of a performability model for the system and measure in question, and (2) evaluation of the measure via solution of the model. More precisely, performability model *construction* consists of specifying the performance variable Y (relative to which *Perf* is defined) and determining a base model X that supports its solution (in the sense described in the previous paragraph). In many cases, as evidenced by the evolution of techniques for this purpose over the past 15 years (see below), construction of X will often invoke some form of intervening model, e.g., a graphical model whose state behavior is then identified with X. Performability model *solution* is a procedure which yields performability values *Perf(B)* for accomplishment sets B that are of interest to the user. Generally, knowledge of the probability distribution function (PDF) of Y suffices to determine such values and, hence, one can regard a performability model as "fully solved" once the PDF of Y is determined. Although closed-form solutions of this PDF are sometimes attainable, it must typically be determined via numerical or simulation techniques. Also, in many applications, only certain types of accomplishment sets have useful interpretations; hence, a complete solution is often not called for.

In the subsections that follow, we attempt to trace, mainly through references to published literature, the evolution of work on unified performance-dependability measures, model construction/solution techniques (both methods and tools), and applications thereof. Although, technically, notions of strict (correct) performance or dependability are special cases of performability, our coverage here, as one might expect, is limited to developments that define, support, or apply truly unified measures of the type just described. For convenience, we choose to divide a 15-year history of this work into three consecutive 5-year periods, beginning with the period that marked the conception and refinement of the framework just described.

[2] In the original formulation of this framework, X was restricted to the utilization period T. However, as several colleagues were kind enough to suggest, T is a user-oriented, rather than system-oriented, consideration and should thus not constrain one's perception of total system behavior, as expressed by the base model X.

3.1. 1976–1980

Work during this epoch dealt with a variety of topics, including alternative formulations of combined performance-dependability measures motivated by various system and/or application considerations. An early contribution in this regard was Beaudry's treatment of "performance-related reliability" [8,9] where, by associating a fixed computation rate with each structure state, constant fault arrival rates (in a Markov reliability model) are translated into "faults per unit of computation". Accordingly, for example, a reliability measure such as "mean time to failure", when applied to the translated model, becomes the performance-related measure "mean computation to failure". Moreover, techniques for evaluating the former apply equally as well to the latter, since the translated model is likewise Markovian. In the more general context of queueing systems, problems of degradable performance (although not referred to there as such) were also beginning to receive attention, for example, the investigation by Neuts and Lucantoni [89] of a multiserver queue subject to breakdowns and repairs. Here, using transform methods developed by Mitrani and Avi-Itzhak [81], it was shown that fault-caused congestion (queue length buildups) can have adverse effects that may linger well beyond the completion of repair.

Most of the effort, however, stemmed from interests in fault-tolerant computing. These contributions, ranging from evaluation methods to specific applications, included the work of Losq ([63]; degradable systems composed of degradable resources), Troy ([101]; efficiency evaluation of dynamic reconfiguration algorithms), Gay and Ketelsen ([36]; performance evaluation of degradable systems), Mine and Hatayama ([79]; job-related reliability), De Souza ([27]; benefit analysis of fault tolerance), Castillo and Siewiorek ([16]; performance-reliability models for computing systems), Chou and Abraham ([20]; performance-availability models of shared resource multiprocessors), and Osaki and Nishio ([91]; reliability of information).

Our own work during this period, some of which was reported in connection with the basic definitions described earlier in this section, focused initially on evaluation with respect to discrete-valued performance. In this case, for any level of accomplishment $a \in A$, the ability to perform exactly at that level is measurable (i.e., $Perf(\{a\})$ is defined). Moreover, if A is finite, it is possible to consider each level individually and move top-down through a model hierarchy which is founded on a base model X. To account for variations in user demands during a bounded period T, construction of X can employ the notion of a *phased model* [105] where T is decomposed into finite number of consecutive time periods (phases); for each phase, the system's intraphase behavior is represented by a continuous-time, finite-state Markov process. Different phases, however, can be modeled by different processes, subject to constraints which permit the determination of (conditional) interphase transition probabilities. This may be viewed as a generalization of what, in reliability evaluation, is referred to as "phased mission" analysis (see [32], for example). However, more complicated construction and solution techniques are required to accommodate an interesting kind of "functional dependence" [7] that exists between phases, given knowledge of how well the system was able to perform during T (i.e., the value of Y). This approach was first described and illustrated in 1978 [70]; its initial application focus was the performability evaluation of fault-tolerant multiprocessors for aircraft control [35,75,76].

3.2. 1981–1985

During this period, model-based performability evaluation began to mature along lines that characterize the current scope of work in this area. Perhaps the most influential in this regard was the introduction of solution methods based on "reward models" (see [45], for example). In constructing such a model to support a performability solution, the base model $X = (X_t \mid t \in I\}$, is augmented by a reward structure which associates reward *rates* with state occupancies and reward *impulses* with state transitions. (Generally, such rates and impulses are expressed by real numbers; when negative, they have the interpretation of a "penalty" or a "cost".) The stochastic process X, together with the reward structure, is a *reward model* for the performance (reward) variable Y. Such a model is *rate-based* if there is no impulse assignment or, equivalently, every transition is assigned an impulse value of 0; *impulse-based* reward models are defined in an analogous manner. In the case of rate-based models, the reward structure is typically described by a real-valued function r defined on the states of X, where $r(q)$ is interpreted as the rate at which reward is accumulated in state q. In this setting and relative to a designated utilization period $T = [0, t]$, an interesting performance (reward) variable is the total reward accumulated during T, i.e., the random variable

$$Y_t = \int_0^t r(X_s) \, \mathrm{d}s$$

For rate-based models of this type, a full solution of performability (i.e., the PDF of Y_t) was initially discussed in [72] for a special class of degradable system models. The base model in this case is a time-homogeneous Markov process and the results include a closed-form solution of performability for a specific dual-processor example. To avoid "stiffness" resulting from the large discrepancy between task arrival rates (high) and fault arrival rates (low), the base model is decomposed into a structure submodel and a set of performance submodels (one for each structure state). Such a decomposition has its roots in theory developed by Courtois [24] and yields an approximate performability model wherein the reward rate assigned to a structure state q is the steady-state performance given by its corresponding performance submodel. (Note that the same kind of approximation was implicit in the earlier work of Beaudry [8,9].) This approach was later extended [33] to provide a method, albeit computationally expensive, for determining the PDF of Y_t relative to semi-Markov reward models that are acyclic and nonrecoverable. (Acyclic models typically represent "nonrepairable systems"; a rate-based reward model is "nonrecoverable" [104] if reward rates experienced are nonincreasing with time, i.e., if u and v are times such that $u \leqslant v$ then the event $r(X_v) > r(X_u)$ has probability 0.) These methods were implemented in a software tool called METAPHOR [34], the first such tool developed specifically for the purpose of performability evaluation.

This period also marked the beginning of the development and application of stochastic Petri nets. Specifically, their use in performance [82] and dependability [11] evaluation motivated the consideration of other features which might make them better suited to performability evaluation. One such class of graphical models, referred to as *stochastic activity networks* (SANs), was developed with this purpose in mind, i.e., to provide a more effective and efficient means of determining the base model component of a performability model [73,77,83]. SANs, along other stochastic net models of this type, have since proved to be the substrate for practical, automated means of performability evaluation (see the subsection that follows).

A variety of other contributions were made during the 1981–1985 period which further expanded the scope of performability-related activity. Some of these were extensions of work cited in the previous subsection; others were initial examinations of new techniques and/or applications. Included here are the results of Castillo and Siewiorek ([17]; connections between workload, performance, and reliability), Huslende ([47]; combined performance-reliability evaluation for degradable systems), Munarin ([86]; performance/reliability analysis of gracefully degrading systems), Arlat and Laprie ([3]; performance-related dependability evaluation), and Krishna and Shin ([56]; performance measures for multiprocessor controllers).

Applications concerned with aspects of communication also began to emerge, e.g., the work of Beeler ([10]; degradable performance in packet switching networks), Li and Sylvester ([61]; performance of networks with unreliable components), and Das and Bhuyan ([25,26]; bandwidth availability of multiple-bus multiprocessors). Finally, in a queueing-theoretic setting, there was some continued interest of the type referred to in the previous section, e.g, the contributions of King and Mitrani [54,80] regarding the effects of breakdowns on the performance of multiserver systems.

3.3. 1986–1990

This period, taking us up to the present, was one of truly accelerated progress with respect to the development of performability model construction/solution techniques, their implementation in model-based performability evaluation tools, and their application to various types of computing and communication systems.

Regarding solution methods, much of this effort dealt with performability models comprised of some form of rate-based Markov reward model along with accumulated reward Y_t (see Section 3.2) as the performance variable in question. Given some value of t, the solution sought is the PDF $F_{Y_t}(y)$, i.e., for any accomplishment level y, the probability

$$F_{Y_t}(y) = \mathrm{P}[Y_t \leqslant y].$$

Specifically, for acyclic, nonrecoverable Markov reward models, Goyal and Tantawi [39] developed a closed-form solution of $F_{Y_t}(y)$. Solution methods, using transform techniques, were also developed. For example, Donatiello and Iyer [29] employed Laplace transforms to derive a closed-form time-domain solution of performability for acyclic Markov reward models; here, unlike the nonrecoverable models considered in [33,39,72], there is no restriction on the nature of the reward rate assignment. In the context of fault-tolerant satellite systems and, again, for acyclic rate-based reward models, Ciciani and Grassi [23] likewise obtained a closed-form solution of the PDF of Y_t If only the expected value $\mathrm{E}[Y_t]$ is desired, solution procedures are typically less complex, due mainly to the linear nature of expectation. Contributions here include those of Marie et al. [64] who developed closed-form solutions of $E[Y_t]$ for the acyclic rate-based case. Sanders and Meyer [95] considered a somewhat more general class of reward models wherein all states of the underlying process X are either transient or absorbing (acyclic models are thus a special case) and the reward structure has impulse as well as rate assignments. A closed-form solution of $E[Y_t]$ was then obtained by determining solutions over two unbounded periods, $[0, \infty)$ and $[t, \infty)$, and then subtracting the second from the first.

In performability models of systems with some form of repair capability, the underlying process model, representing variations in system structure, is no longer acyclic. Moreover, a state that lies in a cycle can be either recurrent (in the case of indefinite repair) or transient (e.g., if repair is limited or fault coverage is not perfect). For models of this type, solutions of the PDF $F_{Y_t}(y)$ are much more difficult to obtain. One approach, applicable to general rate-based Markov reward models, is to use double Laplace transforms that involve transformation of both the time variable t and the accomplishment (accumulated reward) level y. Methods of this type, employing various means of inverting the transformed solution, were investigated by Iyer et al. [51], Kulkarni et al. [57], and Smith et al. [100]. In a similar vein, Ammar et al. [2] derived the PDF of Y_t using Laguerre transformations. An alternative numerical method for calculating values of $F_{Y_t}(y)$ was developed by de Souza e Silva and Gail [28]. It uses "randomization" (see [41], for example) and, although approximate in nature, its accuracy can be specified at the outset. It is applicable to general (cyclic or acyclic) time-homogeneous Markov processes and, via different formulations, can accommodate either impulse-based or rate-based reward models. These formulas are initially derived for the special case of availability evaluation. Their subsequent generalization, which employs colors as a metaphor, provides an excellent example of how performability (where generally many colors are needed) differs from dependability (where black & white suffice). Other recent investigations of such solution methods have examined their extension to semi-Markov reward models (Ciardo et al. [21]) and described their computational aspects in greater detail (Pattipati and Shah [92]).

There was also evidence of progress in the development of solution techniques for steady-state performability measures, although these have received considerably less attention than the type of cumulative measures just discussed. This work, as noted earlier, is typically based on queueing network models of systems for which repair actions can be repeated indefinitely, thus yielding meaningful steady-state behaviors. Results obtained in this regard include the contributions of Müller-Clostermann ([84,85]; degradable queueing networks), Geist et al. ([37]; perceived effect of breakdowns/repairs), and van Dijk ([102]; queueing networks with breakdowns). Queueing systems with "vacations" are likewise relevant here, since a vacation may be interpreted as the repair period that follows a breakdown. Doshi provided a survey of such models [30] and, recently, generalized certain decomposition results for the case of a single server [31].

Other techniques were motivated by problems encountered in performability model construction. Specifying the reward structure of a reward model, for example, is really part of the construction process, even though the reason for its employment is to facilitate model solution. Moreover, if the state space of a base model is large, actual determination of appropriate rates (for states) and impulses (for state transitions) may be a tedious, if not impossible, procedure to carry out. This observation, in connection with the use of SANs (see Section 3.2), led to the notion of a "SAN-based" reward model [98] wherein reward rates are associated directly with the markings of designated places and reward impulses are associated with the completion of activities. A similar approach, using GSPNs, has been investigated by Bobbio [13].

Another important solution-related aspect of model construction concerns just how the notion of state is defined for a base model, even when the latter is being determined by some form of stochastic Petri net. Although the marking of the net is the usual choice, this often results in an excessively large state space that precludes its use for solution purposes. What is called for instead are less refined notions of state that directly yield the type of reductions obtained via lumping (exact) or aggregation (approximate) techniques, without having to first generate a state space that's intractably large (prior to being lumped or aggregated). One such approach, referred to as "variable driven" or "reduced base model" construction [96,97,99], employs a concept of state that relies on knowledge of the performance variable as well as symmetries in the net model. Although developed in the context of SAN-based reward models (see above), it constitutes an effective and exact state space reduction technique that could likewise be applied to other types of stochastic Petri nets. Alternative means of exploiting net symmetries for this purpose have also emerged, notably through the use of constructs such as high level SPNs [62] and colored GSPNs [19].

These past 5 years have also witnessed progress in the development of performability modeling/evaluation tools. In particular, the initial version of METASAN [3] [94] emerged at the outset of this period. It is written in C, employs SANs to construct base models (process models on which solutions are based), and has separate facilities for describing (1) the total system model, and (2) the performance variables in question along with the types of performability solutions required; (1) is specified via an input language called Sanscript; options regarding (2) include both transient and steady-state variables solved by either analysis (if the base model is Markov) or simulation. A number of other performability tools were also produced during this period [4], thereby considerably expanding the access to practical means of model-based performability evaluation. The citations and brief descriptions that follow in no way reflect the enormous amount of creativity and hard work that was devoted to these developments.

SAVE [38]: Written in Fortran 77; base models are specified directly as Markov processes or via special constructs; analytic solutions; originally introduced as an availability evaluation tool but has since been extended.

[3] METASAN is a registered trademark of the Industrial Technology Institute.

[4] In some cases, initial work on their development preceded this period; what we are referring to here, more precisely, is when they were first reported in the open literature.

METFAC [15]: Written in Fortran 77; Markov base models are generated by a production rule system; analytic solutions.

SHARPE [93]: Written in C; base models are specified directly as Markov or semi-Markov processes; analytic solutions.

SPNP [22]: Written in C; employs GSPNs [1] to construct Markov base models; analytic solutions.

Proper [42]: Written in C but uses commercially available compilers for certain purposes; Markov base models are specified via a language called PDL; analytic solutions.

DyQNtool [43]: Written in several languages corresponding to different components of the tool; base models are constructed from extended GSPNs; reward rates are derived from product form queueing networks; analytic solutions.

A common property shared by all these tools is the ability to augment base models with a reward structure, so as to implement at least one solution method of the type discussed earlier in this subsection. In most cases, the resulting reward model is rate-based and, moreover, the reward rate assignment is often hand-specified as part of the input to a designated solution algorithm. Among the tools summarized above, DyQNtool is the most advanced with regard to automating reward model construction; it has a separate window for this purpose, permitting designation of the source and type of performance values (obtained via queueing model analysis) that are to serve as reward rates.

Finally, the past 5 years have seen performability evaluation techniques applied to a variety of systems ranging from relatively small computers to large communication networks. Certain of these studies accompany some form of method or tool development (and, hence, may have been cited earlier that context); others emphasize the evaluation results, per se, and how they contribute to a better understanding of the object systems in question. Simply referencing these contributions does not reflect their quality or their impact on future work. However, in keeping with the spirit of this review, the following should help convey the considerable breadth of this activity.

As applied to computing systems, typically incorporating some form of fault tolerance, it includes evaluation studies conducted by Kulkarni et al. ([58]; multimode computer systems), Iyer et al. ([50]; configurable duplex systems), Nicola et al. ([90]; fault-tolerant systems, using queueing analysis), Cherkassky and Malek ([18]; parallel computer systems), Sanders and Meyer ([95]; distributed fault-tolerant systems), Smith et al. ([100]; an algorithm and a case study), Grassi et al. ([40]; multicomponent fault-tolerant systems), Hsueh et al. ([46]; a case study based on measurement data), Aupperle et al. ([5]; fault-tolerant systems with nonhomogeneous workloads), and Ammar et al. ([2]; parallel and distributed algorithms).

Other performability evaluation studies have dealt more specifically with aspects of interprocessor communication in multiprocessors or internode communication in local area networks. Contributions here include those of Muralidhar and Pimentel ([87]; token bus LANs), Najjar and Gaudiot ([88]; hypercube multiprocessors), Bisbee and Nelson ([12]; shuffle exchange networks), Aupperle and Meyer ([4]; balanced multibus networks), Islam and Ammar ([49]; hypercube multiprocessors), Koren and Koren ([55]; multistage interconnection networks), Meyer et al. ([78]; token bus LANs), and Karmarkar and Kuhl ([53]; multibus LANs).

In conclusion, there's evidence that performability evaluation is useful when applied to telecommunication systems, particularly the type of wide area communication networks that support multiple, integrated services (voice, data, video, etc.). Possible approaches in this regard include those reported by Levy and Wirth ([60]; a unifying approach to performance and reliability objectives), Meyer ([74]; performability evaluation of telecommunication networks), and Jones ([52]; communication system performability).

4. Some pointers to the future

In an era where new technological developments grow old quickly, 15 years is a reasonable duration of time for a technical discipline to mature. Per the review just presented, this appears to have taken place in the growth of model-based performability evaluation. Although not yet an adult, it has certainly reached an age where it may ask "Where do I go from here?" In response to this question, the following are some pointers to directions it might follow. It is highly unlikely, however, that these point to all that's worthwhile pursuing. Moreover, some of the paths indicated may be shorter than one might desire; others may contain pitfalls that delay progress but are nevertheless surmountable. The focus is on needs peculiar to performability although, naturally, certain aspects of these are shared by other types of model-based evaluation. On the other hand, issues that are obviously common to any type of modeling, e.g., important problems such as model validation, are not discussed.

4.1. Measures

Perhaps the most general type of performability measures are those for which performance (the variable Y) is identified with the quality of service (QOS) provided by the object system in a specified operational environment. More precisely, with respect to a designated service, the values of Y represent different degrees of satisfaction that might be experienced by a user of the system. The term "service" here may have a collective meaning that refers to a multiplicity of system services; similarly, "user" may mean a population of users with possibly differing perceptions of service quality.

The definition of QOS has received considerable attention in the context of telecommunication systems, both in general recommendations of of the CCITT (see [48], for example) and in specific recommendations for special types of systems such as integrated services digital networks (ISDNs) and public data networks. In particular, they recognize the important fact that QOS must reflect the combined influence of factors associated with both performance (in the strict sense) and dependability. In their general definition (Recommendation G.106) and at the highest level, this combination is expressed as the "collective effect" of various *service performances* which relate to administrative support of a telecommunication network as well to operational performance at the network-user interface. In the latter category, and closest to variables typically considered in performability evaluation, is the notion of *servability performance* which has components of *service accessibility performance* and *service retainability performance*. These depend, in turn, on lower level "item performances" which describe either the fault-free performance or the dependability of underlying resources.

Although the distinctions made here are peculiar to telecommunication systems, there are at least two aspects of such a QOS "tree" that point to future work on performability measures. One is its hierarchical nature, which appears to be useful both in distinguishing various contributions to QOS and determining how they relate to one another. Thus other application domains, including safety-critical applications for which dependability is the dominant concern, might well benefit from similar tree-structured notions of service quality. Given such a definition of QOS, a second important consideration is just how various lower level performances contribute to the resulting value of service quality at the top of the tree. In other words, how is their "collective effect" (per the CCITT definition) actually measured?

A natural temptation here is to first define measures for the lower level performances and then literally combine them, i.e., the value of the higher level measure is simply a vector of the values supplied by the lower level measures. However, just as measures of strict performance and dependability cannot be so accommodated when performance is degradable, such an approach will generally not suffice. The pitfall is the fact that these individual measures may fail to capture important dependencies among their corresponding variables. On the other hand, by regarding QOS as the variable Y of a performability model, the measure *Perf* expresses the "collective effect" in question, where its formulation is inherent in the way trajectories of the base model X determine values of Y. This is not to say that such a model is easily constructed and solved. Nevertheless, it does suggest that performability modeling can provide a systematic means of dealing with complex notions of service quality, pointing to the prospect of QOS concepts that can indeed be measured and evaluated.

Another type of performability measure that deserves greater attention is system-oriented, as opposed to user-oriented. It can be viewed as a performability generalization of availability, in the sense that the latter expresses the "readiness" of an object system to deliver proper service. Since usual measures of availability are based on a binary notion of system status (ready or not), they are unable to convey intermediate levels of readiness due to degraded operational states and the probabilistic nature of an anticipated environment. Innovative performability measures, accounting for such factors, could result in more refined evaluations of how ready a system is, either at some given time t (pointwise readiness) or during some specified period of time (interval readiness). Moreover, a variable Y_t that expresses readiness at some known time t can be generalized to accommodate random service demands. For example, if T_d is a random variable representing the time of a demand (and, hence, the time when the object system should be ready) then *Perf*, as defined with respect to the variable Y_{T_d}, measures the system's readiness to serve when called on to serve.

4.2. Model construction

In formal terms, as noted in Section 3, construction of a performability model consists of specifying a performance variable Y (relative to which *Perf* is defined) and determining a base model stochastic process X that supports its solution in the sense that desired values of *Perf* can indeed be obtained. Thus, along with X and Y, some additional information is typically required, e.g., a reward structure in the case of reward model solutions, to either explicitly or implicitly describe the capability function that links X to Y. All three of these objects are "end products" of the construction process in the sense that they are the constructs which, via analysis or simulation techniques, are dealt with directly in the subsequent solution phase. Accordingly, the nature of the construction problem depends, for the most part, on where the construction procedure begins. As with other types of model-based evaluation and as evidenced by recent work surveyed in the previous section, the desire here is to start construction at an interface which is as close as possible to a given application domain. This includes consideration of just who is conducting the evaluation as well as the particular nature the object system and its environment.

Since model-based evaluation is of obvious importance during various phases of system design, one undeniable avenue of pursuit is to bring the model construction interface closer to knowledge and abstractions that are familiar to system designers. This is especially needed in early phases, where outcomes of critical design decisions often have an indelible effect on the system that's ultimately realized. Accordingly, as a design evolves down a hierarchy from initial specification to final implementation, at each level of the hierarchy, input to the construction procedure should be in a form compatible with that level. Moreover, depending on the level, determination of an appropriate performability model (X, Y) may involve construction of one or more intermediate models that bridge the gap between design-oriented input at the top and the base model X at the bottom.

This calls for innovative, algorithmic means of translating design-oriented (D-O) models to evaluation-oriented (E-O) models and successively refining the latter. For example, at a given level of a design hierarchy, the input to a performability model construction procedure could be an object system model produced by a design tool, together with supplementary information concerning both Y and relevant aspects of the total system (e.g., fault and workload assumptions) not accounted for by the D-O model. One would then seek algorithms for translating such input into an E-O model at a similar level of abstraction. In addition, there is need to develop algorithms which convert an E-O model at one level of abstraction to a more refined E-O model at a lower level. Such refinement terminates with a characterization of the base model X in a form that is amenable to solution. Implementation of this final step can be found in most existing performability evaluation tools, e.g., for an E-O model that is some variant of a Markovian stochastic Petri net and for a solution that is analytic, an algorithm which determines the generator matrix of the Markov process X identified with the net's marking behavior.

Progress in this regard thus requires appropriate advances and integration of knowledge concerning design-evaluation interfaces, environment modeling, D-O to E-O model translation, and E-O model hierarchies. Included in the last category is the important issue of how the base model, at the bottom of a hierarchy, is linked back up to the performance variable(s) in question. In the case of reward variables based on reward models, this specializes to the problem of determining reward structures for such hierarchies. Even here, options abound with no obvious choices, e.g., to what extent are reward values assigned (as part of the input to the construction procedure), to what extent are they derived (through combinations of lower level values and/or solutions of lower level models), at what levels in the hierarchy do such assignments and derivations take place, etc.

4.3. Model simplification

Once the end products of a construction procedure are in forms suited to solution by analysis or simulation, there remain practical questions as to whether such objects can indeed be constructed and, if so, whether they suffice to support a solution that is feasible. Accordingly, as has been experienced in model-based evaluations of performance or dependability, there is a continuing need for novel techniques that simplify a model without compromising, beyond stated restrictions, the accuracy and timeliness of the results it provides. Moreover, such techniques should be considered at each level of an E-O model hierarchy, for if all simplification is deferred to the end of the construction procedure, the objects sought, particularly the base model, may be too complex or too costly to actually construct.

Perhaps the most familiar means of reducing model complexity is decomposition, based on either spatial distinctions, wherein submodels typically represent different parts of an object system or its environment, or time-scale distinctions of the type that underlie the use of rate-based reward models. Although such techniques have already received considerable attention, there appears to be room for further progress in this direction, particularly in the context of E-O model hierarchies.

If a solution method is analytic and, more specifically, employs numerical techniques applicable to finite-state Markovian base models, the principal concern is the size of a model's state space. To reduce this size, several approaches are known and each deserves continued investigation. One is the elimination of states with predictably very small probabilities of occurrence. The central issues here are how such predictions are made and how the resulting approximate model affects, either optimistically or pessimistically, performability values obtained from its solution. Others involve the use of methods that lump or aggregate states according to some appropriate notion of state equivalence. However, if these are applied at the base-model level then, as an instance of what was noted above, the model's state space (prior to reduction) may be too large to deal with effectively. As a consequence, more practical means of state space reduction have begun to emerge in recent years (see Section 3.3). Such techniques exploit knowledge of the performance variable and symmetries in the object system in the process of constructing higher level models. This, in turn, permits formulation of an appropriate concept of state for the base model which, in effect, aggregates information that's redundant with respect to the evaluation in question. Hence, when constructed, the resulting base model is already in a reduced form. This strategy appears to be very promising and should be explored more extensively.

Another possible means of state space reduction, specific to models that support reward variable solutions, requires a reward structure that's more general than those typically considered. More precisely, instead of requiring a deterministic assignment of reward rates and reward impulses, the entities assigned to states and transitions are random variables. Under prescribed conditions, this permits a lumping of states which, relative to the usual type of reward model, would have to remain distinguished because of differing fixed reward values. The implications of this approach, particularly its feasibility with regard to solution algorithms, are just beginning to receive some attention.

Other types of simplification are peculiar to discrete-event simulation models. For example, via knowledge of the performance variable and symmetries in the object system, it may be possible to identify events that have an equivalent effect on future behavior. If these are accounted for when the model is constructed, it could reduce the average length of a future-event list when the model is executed. This, in turn, would reduce the amount of time required to obtain desired estimates of the performability values in question. In cases where the values relate to rare events that occur with very low probabilities, usual methods of discrete-event simulation are precluded, due to the excessively long execution times needed to satisfy even modest accuracy and confidence requirements. However, advances in special construction methods such as "importance sampling" might eventually remove this barrier, thus expanding the role that simulation models can play in the context of performability evaluation.

4.4. Model solution

Turning now to solution methods, their nature obviously depends on the type of performability model to be solved, i.e., specific properties of the base model X, the performance variable Y, and the constructs which relate X to Y. Moreover, just how the model is constructed, i.e., its hierarchical elaboration, its decomposition within levels, and how it is otherwise simplified, can have considerable influence on the means by which the values of *Perf* are determined. In particular, such solution need not be confined to a single technique. Depending on its composition, parts may be solved by analytic means, some parts by simulation, and there may be mix of transient and steady-state techniques. Accordingly, many of the directions indicated above have corresponding pointers to future work on solution methods.

With respect to a particular technique, the principal concerns are the accuracy of the results obtained and the time required to obtain them. On the analytic side, accuracy is an issue only if the model or the solution method is approximate. In either case, one should seek means of determining bounds on the errors of approximation, as they are reflected in the performability values that are ultimately determined. Moreover, if both the model and the solution method are approximate, there is a need to understand the interaction between two types of approximation, the concern being an incompatibility which causes excessive errors when the two are used together. Approximate analytic solution methods are usually simpler than their exact counterparts and, hence, the results they produce can typically by obtained more quickly. For simulation models, accuracy is proportional to solution time and, as noted earlier, high accuracy, high confidence estimates of small probabilities will have to be based on models specially constructed for this purpose. Accordingly, what is called for here are solution techniques that account for the peculiarities of such constructions.

As evidenced by work during the past decade (see Sections 3.2 and 3.3), most of the theory relating to performability solutions has focused on analytic means of solving the PDFs or expected values of reward variables based on Markovian reward models. In spite of this, there is need for a much better understanding of the practical advantages and limitations of such algorithms, particularly those that apply to a large class of models (e.g., the numerical methods proposed by de Souza e Silva and Gail [28]). In parallel, effort should also be devoted to more restricted types of analytic solution methods such as product form solutions. Although constrained in their applicability, such techniques can sometimes provide evaluation results that would otherwise not be obtainable.

4.5. Tools

None of the performability modeling directions pointed to above can be meaningfully pursued in the absence of model-based evaluation tools that implement the types of construction, simplification, and solution methods suggested. Past experience in the development and use of such tools bears this out and, moreover, provides a solid footing for future work in this regard. Although user interfaces will certainly differ for differing application domains, what is inside a tool may be general enough to serve a wide variety of applications. For use in system design, perhaps the most crucial requirement is the ability to implement hierarchical modeling along lines discussed in Section 4.2. Accordingly, this points to the realization of appropriate design-evaluation interfaces, environment modeling aids, automated D-O to E-O model translators, and internal construction algorithms which, with user assistance, permit level-by-level elaboration of an E-O model hierarchy.

As an inherent part of this construction procedure and at each level of the hierarchy, such tools should implement simplification techniques which apply to that level (see Section 4.3) and record information required for simplification at lower levels. Similarly, during model solution, which proceeds back up the hierarchy, a tool should implement the analysis or simulation required at each level and make the linkages which permit higher level models to be solved in terms of results obtained from lower level models. At the top of the hierarchy, desired results of the evaluation is fed back to the designer via the design-evaluation interface. In addition to capability that is specific to a particular evaluation study, tools of the future should be capable retaining descriptions of earlier versions of a design as well as storing results of earlier evaluations. They should also be able to interact with other design and validation tools via a common interface. These call for a database and database management system that would be the backbone of an integrated "tool kit" for a particular application domain. The specifics of its design would depend on the division of responsibility between the tool kit and its intended users.

4.6. Applications

Although the retrospective presented in Section 3 was limited to computer and communication applications, the scope of both past and potential uses of model-based performability evaluation is considerably broader. Generally, it comprises most any type of object system that exhibits degradable performance in the presence of faults. Examples include flexible manufacturing systems, office systems, enterprise systems, intelligent vehicle-highway systems, planning systems, economic systems, and many others. Moreover, it is important to note that the object of a performability evaluation study can take the form of a "process" as well as a "product", e.g., a software development process, an automobile assembly process, etc. Indeed, interesting prospects for future consideration are object systems which are combinations of both. In other words, the object is a "product-in-process" where, via a performability evaluation, one might assess such things as the influence of process quality, as degraded by processing faults, on the quality of the product it produces.

Within the more specific domain of computer and communication systems, it is anticipated that applications of model-based performability will continue to become more widespread, particularly if tools of the type discussed above become a reality. Examples here range from large distributed systems, such as integrated broadband communication networks, down to hardware and software systems. The latter are particularly challenging since performance degradation, in this case, is due strictly to design faults and, moreover, is extremely sensitive to the nature of the operational environment. Software systems also appear to be viable candidates for the type of product-in-process evaluations mentioned above.

In conclusion, the future appears to be filled with opportunities for further progress. Much of what needs to be done will require a great deal of energy and creativity but, with it, should emerge an evaluation capability that is well above the current state-of-the-art.

References

[1] M. Ajmone Marsan, G. Conte, and G. Balbo, A class of generalized stochastic Petri nets for performance evaluation of multiprocessor systems, *ACM Trans. Comput. Systems* **2** (2) (May 1984) 93–122.

[2] H. Ammer, S.M.R. Islam and S. Deng, Performability analysis of parallel and distributed algorithms, in: *Proc. 3rd International Workshop on Petri Nets and Performance Models*, Kyoto, Japan (IEEE Computer Soc. Press, Silver Spring, MD, 1989) 221–227.

[3] J. Arlat and J.-C. Laprie, Performance-related dependability evaluation of supercomputer systems, in: *Proc. 13th International Symposium on Fault-Tolerant Computing*, Milano, Italy (IEEE Computer Soc. Press, Silver Spring, MD, 1983) 276–283.

[4] B.E. Aupperle and J.F. Meyer, Fault-tolerant BIBD networks, in: *Proc. 18th International Symposium on Fault-Tolerant Computing*, Tokyo, Japan (IEEE Computer Soc. Press, Silver Spring, MD, 1988) 306–311.

[5] B.E. Aupperle, J.F. Meyer and L. Wei, Evaluation of fault-tolerant systems with nonhomogeneous workloads, in: *Proc. 19th International Symposium on Fault-Tolerant Computing* Chicago, IL (IEEE Computer Soc. Press, Silver Spring, MD, 1989) 159–166.

[6] A. Avižienis and J.-C. Laprie, Dependable computing: From concepts to design diversity, *Proc. IEEE* **74** (5) (1986) 629–638.

[7] R.A. Ballance and J.F. Meyer, Functional dependence and its application to system evaluation, in: *Proc. 1978 Johns Hopkins Conference on Information Sciences and Systems* (EE Dept., Johns Hopkins Univ., Baltimore, MD, 1978) 280–285.

[8] M.D. Beaudry, Performance related reliability measures for computing systems, in: *Proc. 7th International Symposium Fault-Tolerant Computing*, Los Angeles, CA (*IEEE Computer Soc. Press Press*, Silver Spring, MD, 1977) 16–21.

[9] M.D. Beaudry, Performance-related reliability measures for computing systems, *IEEE Trans. Comput.* **27** (6) (1978) 540–547.

[10] M. Beeler, Degradable performance in packet switching networks, in: *COMPCON Fall*, Washington, DC (IEEE Computer Soc. Press, Silver Spring, MD, 1982) 437–443.

[11] B. Beyaert, G. Florin, P. Lonc and S. Natkin, Evaluation of computer system dependability using stochastic Petri nets, in: *Proc. 11th International Symposium on Fault-Tolerant Computing*, Portland, ME (IEEE Computer Society Press, Silver Spring, MD, 1981) 66–71.

[12] C.R. Bisbee and V.P. Nelson, Failure dependent bandwidth in shuffle-exchange networks, *IEEE Trans. Comput.* **37** (7) (1988) 853–858.

[13] A. Bobbio, Petri nets for generating Markov reward models for performance/reliability analysis of degradable systems, in: D. Potier and B. Puigjaner, eds., *Modelling Techniques and Tools for Computer Performance Evaluation* (Plenum, New York, NY, 1989) 353–365.

[14] B.R. Borgerson and R.F. Freitas, A reliability model for gracefully degrading and standby-sparing systems, *IEEE Trans. Comput.* **24** (5) (1975) 517–525.

[15] J.A. Carrasco and J. Figueras, METFAC: Design and implementation of a software tool for modeling and evaluation of complex fault-tolerant computing systems, in: *Proc. 16th International Symposium on Fault-Tolerant Computing*, Vienna, Austria (IEEE Computer Soc. Press, Silver Spring, MD, 1986) 424–429.

[16] X. Castillo and D.P. Siewiorek, A performance-reliability model for computing systems, in: *Proc. 10th International Symposium on Fault-Tolerant Computing*, Kyoto, Japan (IEEE Computer Soc. Press, Silver Spring, MD, 1980) 187–192.

[17] X. Castillo and D.P. Siewioret, Workload, performance, and reliability of digital computing systems, in: *Proc 11th International Symposium on Fault-Tolerant Computing*, Portland, ME (IEEE Computer Soc. Press, Silver Spring, MD, 1981) 84–89.

[18] V. Cherkassky and M. Malek, Graceful degradation of multiprocessor systems, in: *International Conference on Parallel Processing*, St. Charles, IL (EE Dept., Penn. State Univ., 1987) 885–888.

[19] G. Chiola and G. Franceschinis, Colored GSPN models and automatic symmetry detection, in: *Proc. 3rd International Workshop on Petri Nets and Performance Models*, Kyoto, Japan (IEEE Computer Soc. Press, Silver Spring, MD, 1989) 50–60.

[20] T.C.K. Chou and J.A. Abraham, Performance/availability model of shared resource multiprocessors, *IEEE Trans. Reliabil.* **29** (1) (1980) 70–76.

[21] G. Ciardo, R.A. Marie, B. Sericola and K.S. Trivedi, Performability analysis using semi-Markov reward processes, *IEEE Trans. Comput.* **39** (10) (1990) 1251–1264.

[22] G. Ciardo, J. Muppala and K.S. Trivedi, SPNP: a graphical tool for performance analysis, in: *Proc. 3rd International Workshop on Petri Nets and Performance Models*, Kyoto, Japan (IEEE Computer Soc. Press, Silver Spring, MD, 1989) 142–151.

[23] B. Ciciani and V. Grassi, Performability evaluation of fault-tolerant satellite systems, *IEEE Trans. Commun.* **35** (4) (1987) 403–409.

[24] P.-J. Courtois, *Decomposability, Queueing, and Computer Science Applications* (Academic Press, New York, NY, 1977).

[25] C.R. Das and L.N. Bhuyan, Bandwidth availability of multiple-bus multiprocessors, *IEEE Trans. Comput.* **34** (10) (1985) 918–926.

[26] C.R. Das and L.N. Bhuyan, Computation availability of

multiple-bus multiprocessors, in: *International Conference on Parallel Processing*, St. Charles, IL (IEEE Computer Soc. Press, Silver Spring, MD, 1985) 807–813.

[27] J.M. De Souza, A unified method for the benefit analysis of fault-tolerance, in: *Proc. 10th International Symposium on Fault-Tolerant Computing*, Kyoto, Japan (IEEE Computer Soc. Press, Silver Spring, MD, 1980) 201–201.

[28] E. de Souza e Silva and H.R. Gail, Calculating availability and performability measures of repairable computer systems using randomization, *J. ACM* **36** (1) (1989) 171–193.

[29] L. Donatiello and B.R. Iyer, Analysis of a composite performance reliability measure for fault-tolerant systems, *J. ACM* **34** (1) (1987) 179–199.

[30] B.T. Doshi, Queueing systems with vacations: a survey, *Queueing Systems* **1** (1986) 29–66.

[31] B.T. Doshi, Generalizations of stochastic decomposition results for single server queues with vacations, *Stochastic Models* **6** (2) (1990) 307–333.

[32] J. Esary and H. Ziehms, Reliability analysis of phased missions, in: R.E. Barlow, J.B. Fussell and N.D. Singpurwalla, eds., *Reliability and Fault Tree Analysis* (Society for Industrial and Applied Mathematics, Berkeley, CA, 1974) 213–236.

[33] D. Furchtgott and J.F. Meyer, A performability solution method for degradable nonrepairable systems, *IEEE Trans. Comput.* **33** (6) (1984) 550–554.

[34] D.G. Furchtgott, Performability models and solutions, Technical Report CRL-TR-8-84, Computing Research Laboratory, The University of Michigan, January 1984.

[35] D.G. Furchtgott and J.F. Meyer, Performability evaluation of fault-tolerant multiprocessors, in: *Digest 1978 Government Micro-Circuit Applications Conference* Monterey, CA (NTIS, Springfield, VA, 1978) 362–365.

[36] F.A. Gay and M.L. Ketelsen, Performance evaluation for gracefully degrading systems, in: *Proc. 9th International Symposium on Fault-Tolerant Computing*, Madison, WI (IEEE Computer Soc. Press, Silver Spring, MD, 1979) 51–58.

[37] R.M. Geist et al., The perceived effect of breakdown and repair on the performance of multiprocessor systems, *Performance Evaluation* **6** (1986) 249–260.

[38] A. Goyal, W.C. Carter, E. de Souza e Silva, S.S. Lavenberg and K.S. Trivedi, The system availability estimator, in: *Proc. 16th International Symposium on Fault-Tolerant Computing* Vienna, Austria (IEEE Computer Soc. Press, Silver Spring, MD, 1986) 84–89.

[39] A. Goyal and A.N. Tantawi, Evaluation of performability for degradable computer systems, *IEEE Trans. Comput.* **36** (6) (1987) 738–744.

[40] V. Grassi, L. Donatiello and G. Iazeolla, Performability evaluation of multicomponent fault-tolerant systems, *IEEE Trans. Reliabil.* **37** (2) (1988) 216–222.

[41] D. Gross and D.R. Miller, The randomization technique as a modeling tool and solution procedure for transient Markov processes, *Oper. Res.* **32** (2) (1984) 343–361.

[42] B.R. Haverkort and I.G. Niemegeers, Proper, a performability modelling and analysis tool, in: D. Potier and B. Puigjaner, eds., *Modelling Techniques and Tools for Computer Performance Evaluation* (Plenum, New York, NY, 1989) 335–352.

[43] B.R. Haverkort, I.G. Niemegeers and P. Veldhuyzen van Zanten, DyQNtool: a performability tool based on the dynamic queueing network concept, in: G. Balbo, ed., *Modelling Techniques and Tools for Computer Performance Evaluation* (North-Holland, Amsterdam, 1991).

[44] A.L. Hopkins, T.B. Smith and J. Lala, FTMP—a highly reliable fault-tolerant multiprocessor for aircraft, *Proc. IEEE* **66** (10) (1978) 1221–1239.

[45] R.A. Howard, *Dynamic Probabilistic Systems, Vol. II: Semi- Markov and Decision Processes* (Wiley, New York, NY, 1971).

[46] M.C. Hsueh, R.K. Iyer and K.S. Trivedi, Performability modeling based on real data: a case study, *IEEE Trans. Comput.* **37** (4) (1988) 478–484.

[47] R. Huslende, A combined evaluation of performance and reliability for degradable systems, in: *ACM Performance Evaluation Rev.* **10** (1981) 157–164.

[48] International Telecommunication Union *CCITT Red Book, Fasc. III.1: General Characteristics of International Telephone Connections and Circuits*, Geneva, Switzerland (1985).

[49] B.M.R. Islam and H.H. Ammar, Performability of the hypercube, *IEEE Trans. Reliabil.* **38** (5) (1989) 518–526.

[50] B.R. Iyer, D.M. Dias and P.S. Yu, Performability analysis of operation modes of configurable duplex systems, in: *ACM–IEEE Fall Joint Computer Conference*, Dallas, TX (IEEE Computer Soc. Press, Silver Spring, MD, 1986) 785–796.

[51] B.R. Iyer, L. Donatiello, and P. Heidelberger, Analysis of performability for stochastic models of fault-tolerant systems, *IEEE Trans. Comput.* **35** (10) (1986) 902–907.

[52] D.R. Jones and H.A. Males, Communication systems performability: new horizon, in: *IEEE International Conference on Communications*, Boston, MA, USA (IEEE, 1989) 15–23.

[53] V.V. Karmarkar and J.G. Kuhl, Fail-softness evaluation in multiple-bus local computer networks, in: *Proc. 19th International Symposium on Fault-Tolerant Computing*, Chicago, IL (IEEE Computer Soc. Press, Silver Spring, MD, 1989) 538–544.

[54] P.J.B. King and I. Mitrani, The effect of breakdown on the performance of multiprocessor systems, in: F.J. Kylstra, ed., *Performance '81* (North-Holland, Amsterdam, 1981) 201–211.

[55] I. Koren and Z. Koren, On gracefully degrading multiprocessors with multistage interconnection networks, *IEEE Trans. Reliabil.* **38** (1) (1989) 82–89.

[56] C.M. Krishna and K.G. Shin, Performance measures for multiprocessor controllers, in: A.K. Agrawala and S.K. Tripathi, eds., *Performance '83* (North-Holland, Amsterdam, 1983) 229–250.

[57] V.G. Kulkarni, V.F. Nicloa, R.M. Smith and K.S. Trivedi, Numerical evaluation of performability and job completion time in repairable fault-tolerant systems, in: *Proc. 16th International Symposium on Fault-Tolerant Computing*, Vienna, Austria (IEEE Computer Soc. Press, Silver Spring, MD, 1986) 252–257.

[58] V.G. Kulkarni, V.F. Nicola and K.S. Trivedi, On modeling the performance and reliability of multimode computer systems, *J. Syst. Softw.* **6** (1 & 2) (1986) 175–182.

[59] J.-C. Laprie, Dependability: A unifying concept for reliable computing and fault tolerance, in: T. Anderson,

ed., *Dependability of Resilient Computers* (BSP Professional Books, 1989) 1–28.

[60] Y. Levy and P. Wirth, A unifying approach to performance and reliability objectives, in: M. Bonatti, ed., *Teletraffic Science* (North-Holland, Amsterdam, 1989) 1173–1179.

[61] V.O.K. Li and J.A. Silvester, Performance analysis of networks with unreliable components, *IEEE Trans. Commun.* **32** (10) (1984) 1105–1110.

[62] C. Lin and D.C. Marinescu, Stochastic high-level Petri nets and applications, *IEEE Trans. Comput.* **37** (7) (1988) 815–825.

[63] J. Losq, Effects of failures on gracefully degradable systems, in: *Proc. 7th International Symposium on Fault-Tolerant Computing*, Los Angeles, CA (IEEE Computer Soc. Press, Silver Spring, MD, 1977) 29–34.

[64] R.A. Marie, A.L. Reibman and K.S. Trivedi, Transient analysis of acyclic Markov chains, *Performance Evaluation* **7** (1987) 175–194.

[65] J. F. Meyer, A definition of system reliability, JPL Technical Summary 3341-66-1, Jet Propulsion Laboratory, February 1966.

[66] J.F. Meyer, Computation-based reliability analysis, in: *Proc. 5th International Symposium on Fault-Tolerant Computing*, Paris, France (IEEE Computer Soc. Press, Silver Spring, MD, 1975) 123.

[67] J.F. Meyer, An approach to evaluating the effectiveness of computing systems, in: *Proc. 1976 Johns Hopkins Conference on Information Sciences and Systems*, Baltimore, MD (EE Dept., Johns Hopkins Univ., Baltimore, MD, 1976) 376–383.

[68] J.F. Meyer, Computation-based reliability analysis, *IEEE Trans. Comput.* **25** (6) (1976) 578–584.

[69] J.F. Meyer, A model hierarchy for evaluating the effectiveness of computing systems, in: *Proc. 3rd National Reliability Symposium*, Perros-Guirec, France (CNET, France, 1976) 539–555.

[70] J.F. Meyer, On evaluating the performability of degradable computing systems, in: *Proc. 8th International Symposium on Fault-Tolerant Computing*, Toulouse, France (IEEE Computer Soc. Press, Silver Spring, MD, 1978) 44–49.

[71] J.F. Meyer, On evaluating the performability of degradable computing systems, *IEEE Trans. Comput.* **29** (8) (1980) 720–731.

[72] J.F. Meyer, Closed-form solutions of performability, *IEEE Trans. Comput.* **31** (7) (1982) 648–657.

[73] J.F. Meyer, Performability modeling of distributed real-time systems, in: G. Iazeolla, P.-J. Courtois and A. Hordijk, eds., *Mathematical Computer Performance and Reliability* (North-Holland, Amsterdam, 1988) 361–372.

[74] J.F. Meyer, Performability evaluation of telecommunication networks, in: M. Bonatti, ed., *Teletraffic Science* (North-Holland, Amsterdam, 1989) 1163–1172.

[75] J.F. Meyer, D.G. Furchtgott and L.T. Wu, Performability evaluation of the SIFT computer, in: *Proc. 9th International Symposium on Fault-Tolerant Computing* Madison, WI (IEEE Computer Soc. Press, Silver Spring, MD, 1979) 43–50.

[76] J.F. Meyer, D.G. Furchtgott and L.T. Wu, Performability evaluation of the SIFT computer, *IEEE Trans. Comput.* **29** (6) (1980) 501–509.

[77] J.F. Meyer, A. Movaghar and W.H. Sanders, Stochastic activity networks: structure, behavior, and application, in: *Proc. International Workshop on Timed Petri Nets*, Torino, Italy (IEEE Computer Soc. Press, Silver Spring, MD, 1985) 106–115.

[78] J.F. Meyer, K.H. Muralidhar and W.H. Sanders, Performability of a token bus network under transient fault conditions, in: *Proc. 19th International Symposium on Fault-Tolerant Computing*, Chicago, IL (IEEE Computer Soc. Press, Silver Spring, MD, 1989) 175–182.

[79] H. Mine and K. Hatayama, Performance related reliability measures for computing systems, in: *Proc. 9th International Symposium on Fault-Tolerant Computing*, Madison, WI (IEEE Computer Soc. Press, Silver Spring, MD, 1979) 52–59.

[80] I. Mitrani and P.B.J. King, Multiserver systems subject to breakdowns: an empirical study, *IEEE Trans. Comput.* **32** (1) (1983) 96–98.

[81] I.L. Mitrani and B. Avi-Itzhak, A many-server queue with service interruptions, *Oper. Res.* **16** (1968) 628–638.

[82] M.K. Molloy, Performance analysis using stochastic Petri nets, *IEEE Trans. Comput.* **31** (9) (1982) 913–917.

[83] A. Movaghar and J.F. Meyer, Performability modeling with stochastic activity networks, in: *Proc. Real-Time Systems Symposium*, Austin, TX (IEEE Computer Soc. Press, Silver Spring, MD, 1984) 215–224.

[84] B. Müller-Clostermann, A decomposition approach for the stationary analysis of fault tolerant queueing networks, *J. Syst. Softw.* **6** (1 & 2) (1986) 199–204.

[85] B. Müller-Clostermann, An approximate product form for a class of degradable queueing networks, *Performance Evaluation* **8** (1988) 165–172.

[86] J. Munarin, Dynamic workload model for performance/reliability analysis of gracefully degrading systems, in: *Proc. 13th International Symposium on Fault-Tolerant Computing*, Milano, Italy (IEEE Computer Soc. Press, Silver Spring, MD, 1983) 290–295.

[87] K.H. Muralidhar and J.R. Pimentel, Performability analysis of the token bus protocol, in: *Proc. IEEE INFOCOM'87*, San Fransisco, CA, March 1987 (IEEE Computer Soc. Press, Silver Spring, MD, 1987) 55–63.

[88] W. Najjar and J.-L. Gaudiot, Reliability and performance modeling of hypercubebased multiprocessors, in: G. Iazeolla, P.-J. Courtois and O.J. Boxma, eds., *Computer Performance and Reliability* (North-Holland, Amsterdam, 1988) 305–319.

[89] M.F. Neuts and D.M. Lucantoni, A Markovian queue with N servers subject to breakdowns and repairs, *Manage. Sci.* **25** (9) (1989) 849–861.

[90] V.F. Nicola, V.G. Kulkarni and K.S. Trivedi, Queueing analysis of fault-tolerant computer systems, *IEEE Trans. Softw. Eng.* **13** (3) (1987) 363–375.

[91] S. Osaki and T. Nishio, *Reliability Evaluation of Some Fault-Tolerant Computer Architectures* Lecture Notes in Computer Science, Vol. 97 (Springer, Berlin, 1980).

[92] K.R. Pattipati and S.A. Shah, On the computational aspects of performability models of fault-tolerant computer systems, *IEEE Trans. Comput.* **39** (6) (1990) 832–835.

[93] R.A. Sahner and K.S. Trivedi, Reliability modeling using SHARPE, *IEEE Trans. Reliabil.* **13** (2) (1987) 186–193.

[94] W.H. Sanders and J.F. Meyer, METASAN: a performability evaluation tool based on stochastic activity networks, in: *ACM - IEEE Fall Joint Computer Conference* Dallas, TX (IEEE Computer Soc. Press, Silver Spring, MD, 1986) 807-816.

[95] W.H. Sanders and J.F. Meyer, Performability evaluation of distributed systems using stochastic activity networks, in: *Proc. 2nd International Workshop on Petri Nets and Performance Models*, Madison, WI (IEEE Computer Soc. Press, Silver spring, MD, 1987) 111-120.

[96] W.H. Sanders and J.F. Meyer, Performance variable driven construction methods for stochastic activity networks, in: G. Iazeolla, P.-J. Courtois, and O.J. Boxma, eds., *Computer Performance and Reliability* (North-Holland, Amsterdam, 1988) 383-398.

[97] W.H. Sanders and J.F. Meyer, Reduced base model construction methods for stochastic activity networks, in: *Proc. 3rd International Workshop on Petri Nets and Performance Models* Kyoto, Japan (IEEE Computer Soc. Press, Silver Spring, MD, 1989) 74-84.

[98] W.H. Sanders and J.F. Meyer, A unified approach for specifying measures of performance, dependability, and performability, in: A. Avižienis and J.-C. Laprie, eds., *Dependable Computing for Critical Applications* (Springer, Berlin, 1990) 215-237.

[99] W.H. Sanders and J.F. Meyer, Reduced base model construction methods for stochastic activity networks, *IEEE J. Selected Areas in Comm.* **9** (1) (1991) 25-36.

[100] R.M. Smith, K.S. Trivedi and A.V. Ramesh, Performability analysis: measures, an algorithm, and a case study, *IEEE Trans. Comput.* **37** (4) (1988) 406-417.

[101] R. Troy, Dynamic reconfiguration: an algorithm and its efficiency evaluation, in: *Proc. 7th International Symposium on Fault-Tolerant Computing*, Los Angeles, CA (IEEE Computer Soc. Press, Silver Spring, MD, 1977).

[102] N.M. van Dijk, simple bounds for queueing systems with breakdowns, *Performance Evaluation* **8** (1988) 117-128.

[103] J.H. Wensley, L. Lamport, J. Goldberg, M.W. Green, K.N. Levitt, P.M. Melliar-Smith, R.E. Shostak and C.B. Weinstock, SIFT: design and analysis of a fault-tolerant computer for aircraft control, *Proc. IEEE* **66** (10) (1978) 1240-1255.

[104] L.T. Wu, Operational models for the evaluation of degradable computing systems, in: *ACM Performance Evaluation Rev.* **11** (1982) 179-185.

[105] L.T. Wu and J.F. Meyer, Phased models for evaluating the performability of computing systems, in: *Proc. 1979 Johns Hopkins Conference on Information Sciences and Systems* Baltimore, MD (EE Dept., Johns Hopkins Univ., Baltimore, MD, 1979) 426-431.