THE UNIVERSITY OF MICHIGAN

COMPUTING RESEARCH LABORATORY

PERFORMABILITY MODELING OF DISTRIBUTED REAL-TIME SYSTEMS

J. F. Meyer

CRL-TR-28-83

AUGUST 1983

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

Invited Paper: International Workshop on Applied Mathematics and
Performance/ Reliability Models of Computer/ Communication Systems
Pisa, Italy
Sept. 1983

ABSTRACT

This survey/position paper concerns modeling concepts and techniques for the performability (performance-reliability) evaluation of distributed real-time systems. Due to the nature of application requirements, such systems typically exhibit properties of concurrency, timeliness, fault tolerance, and degradable performance. Relevant prior research is surveyed and classified according to these properties and, based on the position that performability models should accommodate all four properties, directions for future research are suggested.

I. INTRODUCTION

The inherent complexity of distributed real-time systems, coupled with rapidly increasing demands for their application, have caused their investigation to become a subject of recognized importance. In this paper, we consider problems of modeling such systems for the purpose of evaluating their ability to perform, i.e., their performability (a unified performance-reliability concept which includes, as special cases, the usual notions of performance and reliability). More specifically, we are concerned with the development of mathematical models and model-based methodologies for evaluating the performability of distributed systems designed for real-time applications, e.g., computer-communication networks, vehicle control systems, industrial process-control systems, automated manufacturing systems, etc.

The intent of this paper is to (i) identify properties of such systems that need to be accounted for in the modeling process, (ii) survey and classify prior research according to the identified properties, and (iii) suggest directions for future research based on existing model deficiencies.

II. SYSTEM PROPERTIES

Performance-reliability requirements for distributed real-time systems, although differing in detail from application to application, have some typical properties which, in turn, impose special conditions on system structure and behavior. In particular, such requirements typically call for high performance when the system is fault-free. This, coupled with the fact that resources are distributed, results in systems which exploit concurrent (parallel) processing. Secondly, real-time aspects of such requirements often impose conditions on the "timeliness" of processing activities, e.g., requirements for maximum allowable (input-output) response times which impose deadlines on the times that certain processes can be initiated or completed. Thirdly, real-time applications are typified by requirements for high reliability as well as high performance, resulting in fault-

tolerant systems which employ added resources and processes for the purpose of tolerating specified types of faults. Finally, due to a balance of demands for both performance and reliability, such requirements typically call for systems which exhibit degradable performance in the presence of faults, i.e., between the extremes of nondegraded performance (as would be exhibited if the system were fault-free throughout utilization) and fully degraded performance (system failure), the system is able to perform at intermediate levels which provide some benefit to the user.

The properties identified above, namely

1) Concurrency
2) Timeliness
3) Fault tolerance
4) Degradable performance

distinguish a class of systems which deserves increased attention with respect to the development of effective modeling methods. By their definitions, each property affects a system's ability to perform and, hence, each property needs to be appropriately represented in models for performability evaluation, However, a survey of prior work on unified performance-reliability models (Section III) indicates that this work has been primarily concerned with properties of fault tolerance and degradable performance. Concurrency and timeliness (to a lesser extent) have been modeled in a (strict) performance context but, until very recently, these properties have not been accounted for in the construction of performability models.

In suggesting directions for future research (Section IV), we contend that all four of these properties should be addressed in the modeling process. The types of models considered are mainly formal, probabilistic models, meaning (for our purposes) that the model's state behavior is a stochastic process. Such a process, in turn, can serve as a base model for evaluating performability with respect to a designated performance

variable.

In addition to a discussion of models, per se, we suggest that their nature and utility be explored in several directions. These include the investigation of basic model properties and the development of techniques for both constructing and solving such models.

III. SURVEY OF PRIOR RESEARCH

Traditionally, evaluations of computing system performance and reliability have been distinguished by regarding "performance" as "how well the system performs, provided it is correct" (see [1] - [4], for example) and regarding "reliability" as "the probability of performing successfully" (see [5] - [8], for example). In these terms, concurrency and timeliness (properties 1) and 2); see above) are performance related properties and, as such, have received considerable attention in the context of performance evaluation. Fault tolerance (property 3)) is a reliability related property and, likewise, has been dealt with extensively in reliability evaluation studies. On the other hand, degradable performance (property 4)) is a combined performance-reliability property which, as argued in [9], [10], cannot be accommodated by traditional views of either performance or reliability.

Instead, property 4) calls for the type of unified performance-reliability measures embodied by the general concept of performability, introduced in [9] and subsequently refined and applied in a number of studies [10] - [17]. Here, the *performance of a system* S over a specified time period T is represented by a random variable Y_S taking values in a set A. Elements of A are the *accomplishment levels* (performance outcomes) to be distinguished in the evaluation process. The *performability of* S is the probability measure p_S induced by Y_S where, for any measurable set B of accomplishment levels ($B \subseteq A$), $p_S(B)$ is the probability that S performs at a level in B. Solution of performability is based on an underlying stochastic process X_S, called the *base model of* S, which represents the dynamics of the system's structure, internal state, and environment during utilization.

The need for unified measures in the evaluation of degradable systems (systems which exhibit degradable performance) has likewise been recognized by others who, with various approaches, have contributed to the basic literature on this relatively new topic. This includes studies by Beaudry ([18], [19]; performance-related reliability measures), Losq ([20]; degradable systems composed of degradable resources), Troy ([21]; efficiency evaluation), Gay and Ketelsen ([22]; performance evaluation of degradable systems), Mine and Hatayama ([23]; job-related reliability), De Souza ([24]; benefit analysis of fault tolerance), Castillo and Siewiorek ([25], [26]; performance-reliability models for computing systems), Chou and Abraham ([27]; performance-availability models of shared resource multiprocessors). Osaki and Nishio ([28]; reliability of information), Beyaert, Florin, Lonc, and Natkin ([29]; dependability evaluation using stochastic Petri nets), Huslende ([30]; combined performance-reliability evaluation for degradable systems), and Krishna and Shin ([31]; performance measures for multiprocessor controllers).

If we adopt the term "performability" as the generic name for unified performance-reliability measures,† performability evaluation studies (i.e., those cited in the previous two paragraphs) have been concerned, for the most part, with fault-tolerant, degradable systems. Accordingly, associated modeling efforts (in the case of model-based evaluations) have focused on two of the four properties we seek to accommodate, i.e., fault tolerance and degradable performance (properties 3) and 4)). Models that account for concurrency (property 1)) and/or timeliness (property 2)) have received considerable attention during the past 20 years, but, with a few recent exceptions discussed below, they have been developed for the purpose of (strict) performance evaluation or behavior analysis with respect to properties such as boundedness, liveness, determinancy, etc. Such models presume a fixed system structure and thus are neither intended nor suited (without extension) to deal with properties 3) and 4). However, by virtue of their concern for pro-

†This is done for convenience in the discussion that follows; although this use is consistent with the general definition of performability, there is no intent here to legislate terminology.

perties 1) and 2), they embody concepts and constructs that are relevant to future model development.

Included here are Petri nets and their equivalents (see [32] for a recent and thorough coverage of Petri net theory), aimed primarily at behavior analysis of concurrent systems in a nonprobabilistic setting. At a higher level of abstraction, these network models can be represented by "named transition systems" of the type introduced by Keller [33], [34]. Also relevant are efforts to extend Petri-type nets, e.g., via the introduction of timing, to obtain models that are better suited to performance evaluation [35] - [41]. Regarding probabilistic models that deal with concurrency, it is legitimate to include queueing network models of the type that prevail in applications to performance evaluation (as surveyed, for example, in [42]). Typically, such models treat concurrency at much higher (less detailed) levels than do the models referred to above. Only recently have queueing network models dealt with concurrency at somewhat lower levels, e.g., the work of Heidelberger and Trivedi [43], [44]. Of greater relevance are probabilistic network models that capture concurrency at levels, similar to that of Petri nets. The latter include GERT Networks [45] and General Activity Networks [46] that date back to 1964, along with more recent models which are direct probabilistic extensions of Petri nets [29], [47], [48].

As indicated earlier, most of the models referred to in the previous paragraph presume a fixed system structure, thus excluding considerations of fault tolerance and degradable performance. A notable exception is the work of Beyaert, Florin, Lonc, and Natkin [29] wherein the occurrence of a fault is represented by the firing of a Petri net transition. To the extent that queueing models are able to represent concurrency, other exceptions are the work of Gay and Ketelsen [22], Huslende [30] and ourselves [15], [17]. Here faults are accounted for by incorporating a representation of system structure as part of the queueing model's state, thereby allowing structural change due to faults to be reflected in the model's state behavior.

IV. DIRECTIONS FOR FUTURE RESEARCH

The principal theme of this paper is that all four of the properties identified in Section II, that is,

1) Concurrency
2) Timeliness
3) Fault tolerance
4) Degradable performance

should be accounted for, collectively, in modeling the performability of distributed real-time systems. The survey presented in the previous section reveals that existing models are suited to systems exhibiting some of these properties (typically one or two) but not all. Although individual contributions may well have been overlooked, we believe that the survey is representative of the current state of the art in this area. If this is indeed the case, there is a need for innovative extensions of existing modeling techniques in order to deal effectively with all four of these properties.

In the discussion that follows, we suggest directions for future research in this regard. Specific areas addressed are the development of new model types, the investigation of their basic properties, and the development of techniques for constructing and solving such models.

Models

To accommodate the properties in question, we recommend that features of both queueing networks and stochastic Petri nets be extended and incorporated in new model types. Moreover, when defining such models, we believe it is helpful to first define a class of models which are nondeterministic (in their state behavior) but are not probabilistic. Via a specified interpretation of how states change with time, each nondeterministic model, when augmented by a designated set of probability distributions, yields a

corresponding probabilistic model. Although model-based evaluations of performability call for probabilistic models, the formulation of such models can thus employ nonprobabilistic models of the type described above. Moreover, these nonprobabilistic "skeletons" might also be used directly to verify certain model properties. One advantage of this two-step definition method is that it decomposes a relatively complex concept into simpler, more understandable parts. A much more important advantage, however, is that it facilitates the establishment of explicit connections between the structure/behavior of nondeterministic models and that of their corresponding probabilistic extensions.

Some preliminary work in this direction has been undertaken at The University of Michigan and, as a consequence, more extensive research is now in the proposal phase. Although it is too early to supply technical details, an informal description of our approach may help to clarify the types of models we think are necessary.

As the first step in a two-step definition (see above) and at the network level of abstraction, we have defined a class of nondeterministic models called *activity networks*. These networks are more general than Petri nets, where the need to generalize is due to the following important observation. Petri nets (and most other nonprobabilistic models of concurrent systems) exhibit nondeterministic behavior as the consequence of *temporal uncertainty*, i.e., among a set of concurrent activities, there is uncertainty as to which activity will be completed first. In other words (in Petri net terms), among a set of enabled transitions, there is uncertainty as to which transition will fire. Moreover, this is the only source of nondeterminism since enabled transitions are uniquely determined by the state (marking) of a Petri net and the next state is uniquely determined by the present state and the transition that fires.

However, when modeling the structure and behavior of complex systems, one wants to represent *spatial uncertainty* as well as temporal uncertainty, e.g., uncertainty about which activities are initiated in a given state or, on the completion of an activity, uncer-

tainty about the next state of the system. Such uncertainty is often related to faults, e.g., if the activity in question is a fault recovery process, completion of this activity may result in one of several states, including a state that represents system failure. Spatial uncertainty also occurs in representations of fault-free behavior. In a queueing network, for example, when service of a customer is completed (in some node of the network) the customer may leave the system or proceed to one of several nodes specified by the interconnections of the network. In a queueing network model, this uncertainty is quantified by an assignment of branching probabilities to the output connections of each node.

In our formal definition of an activity network, the structural primitives are *places* (as in Petri nets), *activities* (which play the role of Petri net transitions), and *cases* (to be discussed momentarily). The definition incorporates two nondeterministic constructs to represent each of two types of spatial uncertainty. Uncertainty about which activities are enabled in a given state is represented by the introduction of cases which indicate alternative sets of potential enabled activities. A place which has at least one case associated with it is a *decision place*. If a decision place has at least one token, a specific case and, accordingly, a specific set of enabled transitions is then selected nondeterministically. The second type of spatial uncertainty is uncertainty about the next state (according to place markings) once an activity is completed. This is represented by associating a set of sets of output places with each activity, rather than a single set as in Petri nets. When an activity completes, the set of output places to receive tokens is selected nondeterministically from this designated set of sets.

To obtain a corresponding class of probabilistic models (the second step of the two-step definition process), we extend the definition of an activity network to that of a *stochastic activity network*. This is done by quantifying spatial nondeterminism with probabilities and by quantifying temporal nondeterminism with probability distribution functions. More specifically, to a given activity network N we adjoin functions f, g, and F,

where f specifies the probabilities of output place selection, g specifies the probabilities of case selection, and F specifies the probability distribution functions of the activation periods of enabled activities. In each case, these probabilities and distributions generally depend on the marking of the network as well as its structure, affording us a great deal of flexibility in the model construction process.

To assist the process of describing activity network and stochastic activity network behavior, two additional model classes are distinguished at the state-transition level: *activity systems* and *stochastic activity systems*. Activity systems are not new (see the "named transition systems" of [33], [34], for example) and are abstract enough to represent a large variety of nonprobabilistic systems. In particular, they provide a natural, higher level representation of activity networks. The most detailed description of activity system behavior are its *state-activity* sequences, i.e., for a given initial state, the possible sequences of alternating states and activities that can result from a finite number of applications of the transition relation. Thus, when an activity system represents an activity network, the source of spatial uncertainty (completion vs. enabling) is no longer distinguished.

Stochastic activity systems are probabilistic extensions of activity systems where the extension is similar to that made at the network level. Moreover, in the manner that activity systems represent activity networks, stochastic activity systems are higher level representations of stochastic activity networks. The state behavior of a stochastic activity system is a stochastic process which can serve as the base model of a performability model. More precisely, suppose S is a system modeled by a stochastic activity network N. If M is the stochastic activity system corresponding to N then the state behavior of M is a base model X_S of S (see Section III). X_S together with a designated performance variable Y_S comprise a performability model of S.

Model Properties

Once a new class of models is established, there is a need to investigate basic model properties for two purposes. One purpose is to compare the general capabilities and limitations of models with those of other models used for performance-reliability evaluation (e.g., queueing networks and stochastic Petri nets). The second purpose is to obtain a clear understanding of structure-behavior relations for particular model types, providing knowledge that can facilitate the construction and solution of specific performability models.

With regard to both of these purposes, an important first step is to distinguish appropriate notions of model behavior which can serve as the basis for investigating model properties. These notions can differ among model types and, within a model type can be defined at different levels of abstraction. Thus, for a given model type, there are likely to be several notions of model behavior, each one serving a different specific purpose.

Relative to activity networks, for example, the lowest level descriptions of behavior are marking diagrams similar to those employed in the analysis of Petri nets. When an activity network is represented at the state-transition level by an activity system, the state-activity behavior of the latter (see above) is the (stable) marking behavior of the former. Accordingly, in discussing the behavior of activity networks, we need only refer to their corresponding activity systems. Indeed, this is the principal reason for associating activity systems with activity networks.

With respect to activity systems, the most detailed description of a model's behavior is its state-activity behavior. Such descriptions, however, are typically infinite, calling for simpler representations, e.g., something analogous to the "reachability trees" of Petri nets (see [32], Chapter 4, for example). In addition, behavior descriptions at higher (more abstract) levels are also needed. What we seek here are representations of behavior that will enable us to compare the modeling power of activity networks (as represented by

activity systems) with that of Petri nets and other similar network models. One possibility is a language of "activity sequences" defined in a manner analogous to that of a Petri net language (see [32], Chapter 6). However, because activity networks generally exhibit spatial nondeterminism (while Petri nets do not), the details of such definitions are yet to be determined.

Regarding stochastic models such as stochastic activity networks and stochastic activity systems, behavioral notions are complicated by the fact that time is an explicit dimension of model behavior. On the other hand, due to the probabilistic representation of uncertainty, behavior descriptions are more tractable and need not rely on the elaboration of individual state-activity sequences. Moreover, for the purpose of performability evaluation, knowledge of a model's state behavior (as opposed to state-activity behavior) will suffice. Since this is a stochastic process, model behavior can be described in more familiar terms.

Given that appropriate levels of behavior are distinguished, one can then establish structure-behavior properties that serve the purposes cited at the outset of this subsection. Of particular interest, in the context of performability modeling, are conditions on a model's structure that insure a certain type of stochastic behavior. For example, if the model is a stochastic activity network, we would like to know conditions under which its associated stochastic process has some specified property, e.g., is a finite-state process, a Markov process, a semi-Markov process, etc.

Model Construction and Solution

Directions suggested here concern the process of constructing performability models of systems and subsequently solving the models to obtain solutions of system performability. What we seek are techniques having relatively general applicability which, when appropriately selected and integrated, provide methodologies for performability evaluation.

Regarding model construction, we are proponents of hierarchical model building methods which, relative to a system S, facilitate the determination of a stochastic process representation of S. This process must be able to support evaluations of performability relative to the performance variable(s) of interest. Hence, the best way to proceed with model construction is to begin with a designation of the performance variable(s). In discussing the steps that follow, we suppose that the model types at our disposal are those described earlier, i.e., activity networks, stochastic activity networks, activity systems, and stochastic activity systems. Similar comments would apply to other types of models defined at the network and state-transition levels of abstraction.

The first step is to construct an activity network model of S which is refined enough to support an eventual solution of the system's performability (with respect to the designated performance variable(s)). Also, this network should be sufficiently detailed to permit the specification of the probability functions that convert it to a stochastic activity network. Since a model at this level may be too complex to admit a tractable solution, there is a need for methods of bottom-up hierarchical elaboration to obtain simpler, higher level network models. At some appropriate level, one determines the network's corresponding stochastic activity system and attempts to characterize its state behavior. The latter may call for additional simplification methods applicable to activity systems or their corresponding stochastic processes (state behaviors). The construction procedure terminates with a stochastic process that can serve as a tractable base model for the subsequent solution phase.

This procedure is obviously complex and can benefit considerably from computer programs which assist the model builder in making complicated decisions and constructions. Existing programs that support performance-reliability modeling are mainly concerned with model solution as opposed to model construction. Notable exceptions are the SURF program [49] and programs currently being developed for stochastic Petri nets [29]. How-

ever, a great deal more work needs to be done in this regard, particularly in the area of hierarchical elaboration.

Given the performance variable(s) and base model obtained in the construction phase, performability is determined by solving the probability distribution function(s) of the performance variable(s) in terms of the base model stochastic process. Although, conceptually, this is a classical problem, solutions are typically difficult to obtain for the types of processes and variables associated with realistic performability models. This is due, in part, to the fact that activities bearing on both performance and reliability are being represented in the same base model. Typically, performance related activities (e.g., job processing) are completed at much higher rates than are reliability related activities (e.g., fault arrivals). Hence, in the case of a Markovian base model, the ratio between the highest and lowest state transition rate can be many orders of magnitude. In other words, the differential equations describing the model's state behavior (i.e., those determined by the generator matrix of the Markov model) are "stiff" (See [50], for example) and, when solved numerically, require special treatment to avoid excessive errors.

Various approaches can be used to deal with this stiffness property and all deserve more extensive investigation. One approach is to exploit numerical solution methods developed specifically for stiff equations, e.g., Gear's methods [50]. A second approach is to find alternative formulations of the state occupancy probabilities such that solutions are less subject to numerical errors. An example of this approach are formulas obtained via "selective randomization" [51] where the Markov model is "randomized" with respect to states having high transition rates.

A third approach, and one that we are actively pursuing (see [15], [17]), is to lump states with high transition rates such that the lumped model is no longer stiff. Performance rates in the lumped states are determined via steady-state solution techniques, under the assumption that the consequences of high rate activities approach equilibrium

conditions between the completions of low rate activities. (As a result of this assumption, the solution obtained is approximate; it appears to be a good approximation, however, particularly in the case of very stiff base models). Once these performance rates are determined, performability is solved in terms of the lumped model. The latter step is far from trivial (see [17], Section III) but is capable of producing closed-form solutions as well as numerical solutions. Work is progressing on the development of tractable solution algorithms [52] and further research in this direction is recommended.

Additional difficulties are encountered when the base model is not a Markov process, e.g., the model is semi-Markovian or perhaps even more general than a semi-Markov process. Such processes also arise in (strict) performance and reliability evaluation but, when they do, the measures considered are typically expected values under long-run or steady-state conditions. Performability evaluation, on the other hand, requires solution of the probability distribution function of the performance variable. Moreover, the performance variable is usually defined with respect to a bounded utilization period, requiring (in most cases) the use of transient solution methods. These factors complicate the solution procedure, particularly when the base model is non-Markovian.

Although Markovian performability models (those based on a Markov process) constitute an important class, increased emphasis needs to be placed on methods for solving non-Markovian performability models. Some progress is being made in this regard, e.g., the solution algorithm presented in [52] applies even when the base model is not a semi-Markov process. The base model and performance variable are subject to other constraints, however, and hence the algorithm is not generally applicable. Much more work is needed in this direction if the capabilities of new models discussed earlier are to be exploited.

V. REFERENCES

[1] Kleinrock, L., *Queueing Systems, Volume II: Computer Applications* (Wiley, New York, 1976).

[2] Ferrari, D., *Computer Systems Performance Evaluation* (Prentice-Hall, Englewood Cliffs, NJ, 1978).

[3] Kobayashi, H., *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology* (Addison-Wesley, Reading, MA, 1978).

[4] Sauer, C. and Chandy, K., *Computer Systems Performance Modeling*, (Prentice-Hall, Englewood Cliffs, NJ, 1981).

[5] Bouricius, W. G., Carter, W. C. and Schneider, P. R., Reliability modeling techniques for self-repairing computer systems, *Proc. ACM 1969 Annu. Conf.*, (Aug. 1969) 295-309.

[6] Laprie, J. C., Reliability and availability of repairable structures, *Proc. 1975 Int. Symp. on Fault-Tolerant Comput.*, Paris, France, (June 1975) 87-92.

[7] Ng, Y.-W. and Avizienis, A., A reliability model for gracefully degrading and repairable fault-tolerant systems, *Proc. 1977 Int. Symp. on Fault-Tolerant Comput.*, Los Angeles, CA, (June 1977) 22-28.

[8] Costes, A., Landrault, C. and Laprie, J. C., Reliability and availability models for maintained systems featuring hardware failures and design faults, *IEEE Trans. Comput.*, C-27, (June 1978) 548-560.

[9] Meyer, J. F., On evaluating the performability of degradable computing systems, *Proc. 1978 Int. Symp. Fault-Tolerant Computing*, Toulouse, France, (June 1978) 44-49.

[10] Meyer, J. F., On evaluating the performability of degradable computing systems, *IEEE Tran. Computers*, C-29, (Aug 1980) 720-731.

[11] Meyer, J. F. and Furchtgott, D. G., Performability evaluation of fault-tolerant multiprocessors, *Digest of 1978 Government Microcircuit Applications Conf.*, Monterey, CA, (Nov. 1978) 362-365.

[12] Meyer, J. F. and Wu, L. T., Phased models for evaluating the performability of computing systems, *Proc. 1979, Conference on Information Sciences and Systems*, The John Hopkins Univ., Baltimore, MD, (March 1979).

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

[14] Meyer, J.F., Furchtgott, D.G., and Wu, L.T., Performability evaluation of the SIFT computer, *IEEE Trans. Computers*, C-29, (1980), 501-509.

[15] Meyer, J. F., Closed-form solutions of performability, *Proc. 1981 Int. Symp. Fault-Tolerant Computing*, Portland, ME, (June 1981) 66-71.

[16] Meyer, J. F. and Wu, L. T., Evaluation of computing systems using functionals of a Markov process, *Proc. 14th Hawaii Int. Conf. on System Sciences*, Honolulu, HI, (Jan. 1981) 74-83.

[17] Meyer, J. F., Closed-form solutions of performability, *IEEE Tran. Computers*, C-31, (July 1982) 648-657.

[18] Beaudry, M. D., Performance related reliability measures for computing systems, *Proc. 1977 Int. Symp. Fault-Tolerant Computing*, Los Angeles, CA, (June 1977) 16-21.

[19] Beaudry, M. D., Performance-related reliability measures for computing systems, *IEEE Trans. Computers*, C-27, (June 1978) 540-547.

[20] Losq, J., Effects of failures on gracefully degradable systems, *Proc. 1977 Int. Symp. Fault-Tolerant Computing*, Los Angeles, CA, (June 1977) 29-34.

[21] Troy, R., Dynamic reconfiguration: An algorithm and its efficiency evaluation, *Proc. 1977 Int. Symp. Fault-Tolerant Computing*, Los Angeles, CA, (June 1977) 44-49.

[22] Gay, F. A. and Ketelsen, M. L., 9th Performance evaluation of gracefully degrading systems, *Proc. 1979 Int. Symp. Fault-Tolerant Computing*, Madison, WI, (June 1979) 51-58.

[23] Mine, H. and Hatayama, K., Performance related reliability measures for computing systems, *Proc. 1979 Int. Symp. Fault-Tolerant Computing*, Madison, WI, (June 1979) 59-62.

[24] De Souza, J. M., A unified method for the benefit analysis of fault-tolerance, *Proc. 1980 Int. Symp. Fault-Tolerant Computing*, Kyoto, Japan, (Oct. 1980) 201-203.

[25] Castillo, X. and Siewiorek, D. P., A performance reliability model for computing systems, *Proc. 1980 Int. Symp. Fault-Tolerant Computing*, Kyoto, Japan, (Oct. 1980) 187-192.

[26] Castillo, X. and Siewiorek, D. P., Workload, performance, and reliability of digital computing systems, *Proc. 1981 Int. Symp. Fault-Tolerant Computing*, Portland, ME, (June 1981) 84-89.

[27] Chou, T. C. K. and Abraham, J. A., Performance/availability model of shared resource multiprocessors, *IEEE Trans. Reliability*, R-29, No. 1, (April 1980) 70-76.

[28] Osaki, S. and Nishio, T., *Reliability evaluation of some fault-tolerant Computer Architectures*, Lecture Notes in Computer Science. (Berlin, Germany, Springer-Verlag, 1980).

[29] Beyaert, B., Florin, G., Lonc, P. and Natkin, S., Evaluation of computer system dependability using stochastic petri nets, *Proc. 1981 11th Int. Symp Fault-Tolerant Computing*, Portland, ME, (June 1981) 66-71.

[30] Huslende, E., A combined evaluation of performance and reliability for degradable systems, *Proc. ACM/SIGMETRICS Conf. on Meas. and Modeling of Computing Syst.*, Las Vegas, Nevada, (Sept. 1981) 157-164.

[31] Krishna, C. M. and Shin, K. G. Performance Measures for Multiprocessor Controllers, in: Agrawala, A. K. and Tripathi, S. K. (eds.), *Performance '83* (North-Holland, Amsterdam, 1983).

[32] Peterson, J. L., *Petri Net Theory and the Modeling of Systems*, (Prentice-Hall, Englewood Cliffs, NJ, 1981).

[33] Keller, R., Vector replacement systems: A formalism for modeling asynchronous systems, Technical Report 117, Computer Science Laboratory, Princeton Univ., (December, 1972).

[34] Keller, R. M., Formal verification of parallel programs, *CACM*, 19, (July 1976) 371-384.

[35] Nutt, G., The formulation and application of evaluation nets, Ph.D. Thesis, Computer Science Group, Univ. of Washington, (July 1972).

[36] Ramchandani, C., Analysis of asynchroneous concurrent systems by Petri nets, Ph.D. Thesis, Dept. of Electrical Engineering, MIT, (July 1973).

[37] Baer, J. L. and Jensen, J., Simulation of Large Parallel Systems Modeling of Tasks, in: Bellner, H. and Gelenbe, E. (eds.), *Measuring, Modeling, and Evaluating Computer Systems*, (North-Holland, Amsterdam, 1977).

[38] Han, Y.W., Performance evaluation of a digital system using a petri net-like approach, *Proc. of the National Electronic Conference*, 32, (1978) 155-160.

[39] Noe, J. D., Nets in modeling and simulation, *Net Theory and Application*, Lecture Notes in Computer Science, 84, (Springer-Verlag, Berlin, Germany 1980).

[40] Ramchandani, C., Analysis of asynchroneous concurrent systems by Petri nets, *Ph.D. Thesis*, MIT, Dept. of Electrical Engineering, (July 1973).

[41] Sifakis, J., Performance evaluation of systems using nets, *Net Theory and Applications*, Lecture Notes in Computer Science, 84, (Springer-Verlag, Berlin, Germany 1980).

[42] *Computing Surveys*, 10, No.3, (Sept. 1978).

[43] Heidelberger, P. and Trivedi, K. S., Analytic queueing models for programs with internal concurrency, IBM Res. Rep. RC 9194, (1982).

[44] Heidelberger, P. and Trivedi, K. S., Queueing network models for parallel processing with asynchronous tasks, *IEEE Trans. Computers*, (November 1982) 1099-1109.

[45] Pritsker, A. A. B. and Happ, W. W., GERT: Graphical evaluation and review technique - Part I. Fundamentals, *Journal of Industrial Engineering*, 17, No. 5, (May 1966) 267-74.

[46] Elmaghraby, S. E., An algebra for the analysis of generalized activity networks, *Management Science*, 10, (Apr. 1964) 621-31.

[47] Shapiro, S., A stochastic Petri net with application to modeling occupancy times for concurrent task systems, *Networks*, 9, (Winter 1979) 375-379.

[48] Molloy, M., On the integration of delay and throughput measures in distributed processing models, *Ph.D. Thesis*, UCLA Computer Science Dept., (June 1981).

[49] Costes, A., Doucet, J. E., Landrault, C. and Laprie, J. C., SURF-A program for dependability evaluation of complex fault-tolerant computing systems, *Proc. 1981 Int. Symp. Fault-Tolerant Computing*, Portland, ME, (June 1981) 72-78.

[50] Gear, C. W., *Numerical Initial Value Problems in Ordinary Differential Equations*, (Prentice-Hall, Englewood Cliffs, NJ, 1971).

[51] Miller, D. R., Reliability Calculation Using Randomization for Markovian Fault-Tolerant Computing Systems, *Proc. 1983 Int. Symp. Fault-Tolerant Computing*, Milano, Italy, (June 1983) 284-289.

[52] Furchtgott, D. G. and Meyer, J. F., Performability Evaluation of Computing Systems Using Reward Models, Technical Report, CRL-TR-27-83, Dept. of Electrical and Computer Engineering, Univ. of Michigan (August 1983).