FABIO **A.** SCHREIBER

Information Systems: **A** Challenge for Computers and Communications Reliability

IEEE JOURNAL ON. SELECTED AREAS IN COMMUNICATIONS,

VOL. SAC4 NO. ***6,*** **OCTOBER 1986,**

pp. 1077-1083

Information Systems: A Challenge for Computers and Communications Reliability

FABIO A. SCHREIBER, SENIOR MEMBER, IEEE

Abstract—**Modern information systems are the meeting point of the most advanced technologies both in the digital computers and in the telecommunications fields. Owing to their relevance on the everyday life, their availability became a major concern for system designers as well as for system managers.**

In this paper, we present an integration of the design and evaluation techniques as far as reliability and availability are concerned, in contrast with the duality existing between the computational functions and devices on one part and the telecommunications services and equipment on the other, which no longer makes sense. To this scope, algorithmic components must be considered as part of the system, as well as the more traditional hardware and software components.

A framework, including several research areas, is presented together with some results in the field of analytical models. The relations between the performance and the availability aspects in an Information System are also considered.

I. Introduction

FOR a long time, the important features of high availability and high performance have been obtained, in computing systems, through the use of specialized hardware/software architectures. The main developments in these fields have been achieved in those application areas where availability was a primary goal, such as aerospace systems and electronic telephone exchanges.

The introduction of distributed computing systems, based on local area or geographically spread computer networks, made it feasible the implementation of highly available and gracefully degradable systems. This was accomplished by means of "traditional" computers at "traditional" costs, opening them to wider classes of applications—information systems among them. Moreover, applications in the area of office automation strongly require that typical telecommunications equipment, such as EPABX, be an integral part of an information system.

If the meaning of information system is broadened to include every system which is used to process or transmit information, we can see that, from an application point of view, computer and telecommunications technologies are no longer separable. In managerial information systems, computers are seen as the main processing devices, while the telecommunication subsystems is just seen as a "service" for carrying information among the remote terminals and the computers belonging to the system itself. In digital telecommunications switching systems, transmission and switching are the main functions, while computers are considered as complex "components" which deliver a service in computing routings, accountings, etc. This duality results in an integration of design and evaluation techniques from the point of view of reliability and availability. Computerized information systems are pervading every kind of social service and their availability has a great impact on everyday life.

Manuscript received December 2, 1985; revised May 2, 1986. This work was supported in part by the Ministero della Pubblica Istruzione.

The author is with the Department of Mathematics, University of Parma, I 43100 Parma, Italy.

IEEE Log Number 8609804.

A great deal of research efforts have been spent, on the computers side of the medal, on the control of concurrent transactions and on recovery algorithms for distributed database systems, which constitute the "heart" of distributed information systems. Three basic mechanisms have been described in the literature for concurrency control: locking, optimistic, and time-stamping. For reliable commitment of updating transactions, from two-phase up to four-phase protocols have been proposed. Their performance has been compared in terms of the number of exchanged messages, transaction back-out probability, the number of actions to be redone, etc. [7].

In computer networks, the transportation media of distributed information systems, a lot of work has been done on topologies, capacity assignment, routing algorithms, communication protocols, in order to achieve both good performance and availability [13]. However, little has been done to provide the end user with a feeling of what he can expect from the information system both in terms of availability (continuous service) and performance (e.g., response time and throughput).

The bounds between availability and performance are very tight. Recovery techniques and algorithms require additional system resources in terms of both hardware components and data, and they put an additional workload on the system. This generally results in decreased performance during fault-free operation. The cost of designing and implementing fault-tolerant algorithms can be high, and the marginal gain obtained in the availability/cost rate by using very sophisticated techniques could be small for the end user goals. On the other hand, the same resources needed for fault-tolerant operation could be used, in fault-free system states, for enhancing its performance.

Therefore, quantitative methods must be developed for evaluating the performance, availability, and cost of the overall information system. While many results exist as to the hardware components and subassemblies, little has been done in this direction for the software components and for the algorithms which control the operation of the global system [17], [14]. Studies about system performance have progressed apart from the availability issues and vice versa. In the recent years, the two disciplines having been considered in a unifying view [8].

The purpose of this paper is to define a framework connecting the two research areas and relating their performance and availability. While some results will be mentioned, the main thrust of this paper is to propose the framework, in order to stimulate further research in this field. At the end a short guide to bibliographical references is given for further research into the subject.

II. The Research Framework

Fig. 1 shows the environment in which researches on information systems performance availability quantitative evaluation (ISPAQE) are to be defined. We can find in it three irregularly shaped boxes labeled:

- quantitative features of Hardware/Software system components;
- application programs and data;
- user service levels.

They represent the external environment of the ISPAQUE problem.

The two rectangular boxes, on the contrary, represent the system designer choices, generally called *strategies*, and their influence on the actual operation of the system.

In the following sections we are going to examine each box in detail.

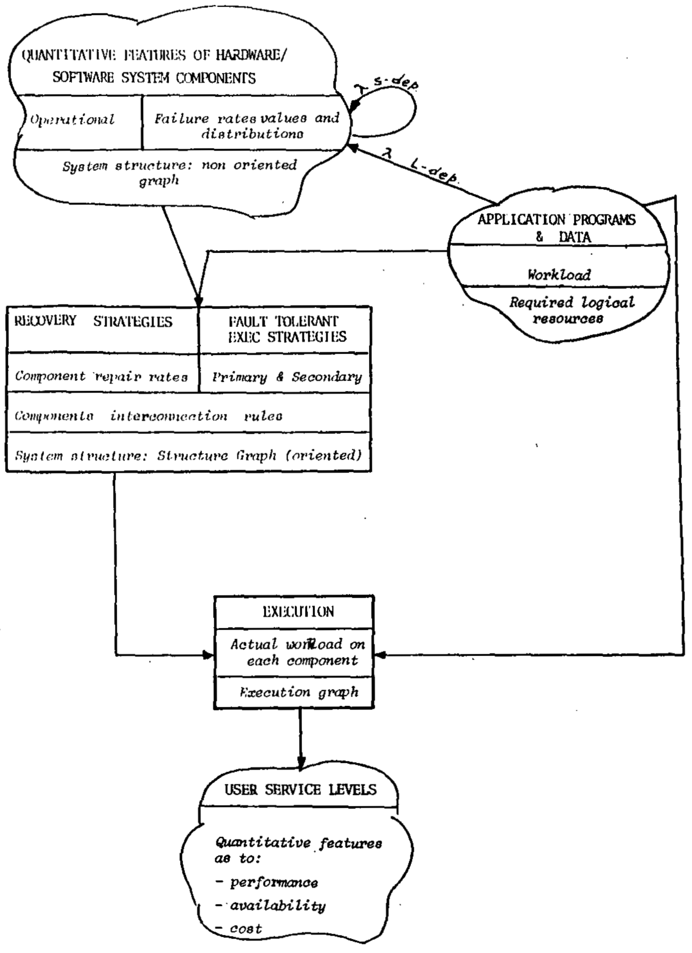
QUANTITATIVE FEATURES OF HARDWARE/ SOFTWARE SYSTEM COMPONENTS
Operational
Failure rates values and distributions
System structure: non oriented graph
λ s-dep
λ L-dep.
APPLICATION PROGRAMS & DATA
Workload
Required logical resources
RECOVERY STRATEGIES
FAULT TOLERANT EXEC STRATEGIES
Component repair rates
Primary & Secondary
Components interconnection rules
System structure: Structure Graph (oriented)
EXECUTION
Actual workload on each component
Execution graph
USER SERVICE LEVELS
Quantitative features as to:
- performance
- availability
- cost

Fig. 1. The research environment.

A. Hardware and Software Components

This box in Fig. 1 represents the research problems concerning the building blocks of the information system, considered either as single components or as subassemblies.

We want to stress that both hardware and software are considered in the same way; also we do not indicate whether the component belongs to the computing or to the communications equipment. The ultimate goal should be that of developing a "components algebra," where components are seen as abstract objects, the properties of which can be studied independently of their technological nature.

Quantitative features of components can be grouped into two classes of parameters: 1) operational parameters, and 2) reliability parameters. The former describe the functional features of the component (e.g., the capacitance value of a capacitor, the current gain of a transistor, the number of instructions/s of a central processor unit, the processing time of a program, etc.) and are mostly related to its performance. The latter describe the component's behavior with respect to a fault-free operation (e.g., mean time-to-first failure, steady-state failure rates, failure probability distributions, etc.).

However, while modeling and evaluation techniques for hardware components and subsystems have been known since many years [2], work must still be done to model the reliability behavior of software. This is especially true of control algorithms, which play an essential role in assuring the availability of an information system. Software, in fact, owes its reliability properties not only to the lack of bugs in the actual programs, but also to the correct design of the control algorithms and to the proper tuning of their control parameters [17].

Looking at transition rates of components, it can be seen that on the basis of their behavior, they can belong to one of the following classes:

- *independent* rates, i.e., their value does not depend on the workload applied to the component, nor on the state of any other component in the system;
- *workload-dependent* rates, i.e., their value depends on the workload applied to the component itself, or to some other component in the system;
- *state-dependent* rates, i.e., their value is related to the state of some other component in the system.

Therefore, an important research area in this field is the study of how the failure rate of a component, independently of its own physical nature, depends on the state of the whole system (S-dependence) and on the system workload (L-dependence). In fact, some studies of availability evaluation of information systems and of their control algorithms suggest that state and workload depen-

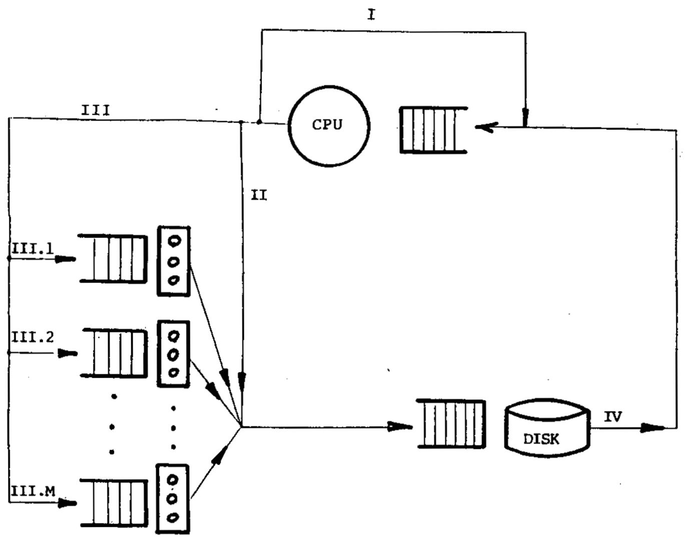

Fig. 2. Model of a concurrency control algorithm.

dence of transition rates can not be neglected, since they constitute a major issue.

The rationale for research in this direction lies in the consideration that, once software components and control mechanisms would be described in the same way as hardware ones (i.e., by means of the transition rates and their distributions and properties), information systems could be modeled using the same well-established techniques used in evaluating reliability and availability of complex hardware equipment.

To substantiate these statements, let us make some examples.

At first, let us consider a concurrency control mechanism for a database management system based on mutual exclusion semaphores. An efficient algorithm should allow as many concurrent accesses as possible to a data item, provided that some dangerous situations—such as reading it while it is being modified—are avoided by locking the data item while it is being read. If we look at this situation from the user's point of view, it looks as if a transient fault has occurred on the memory device where the data item is stored, making it temporarily unavailable. A deadlock situation can be seen as a permanent fault. Frequency and duration of locks can therefore be respectively interpreted as the failure rate and the time to repair for the components of the type semaphore [6], [7], [10], [16]. Fig. 2 shows a possible model for evaluating the transition rates of such a "device." A transaction first asks for access to the data and goes through path III to the relevant semaphores; once it has access, it loops through paths II and IV until processing is completed and eventually it leaves the system freeing the semaphores. Just by inspection it can be seen that the "down time" of a semaphore depends on the global workload presented to the CPU and the disk unit.

In [16], this model is solved in the simplest case of single granule transactions, and an example of the obtained results is shown in Fig. 3. The failure rate is represented as a function of the percentage A of updating transactions and of the multiprogramming degree N (workload).

A very simple example of a subsystem showing state-dependent failures is given by a set of disk drives under the same controller; failure of the controller results in the unavailability of all the drives. Its counterpart in telecommuncations could be represented by a set of twisted pairs housed in the same cable; damaging of the cable results in the unavailability of all the communication circuits housed in it [14]. Other examples showing similarities between computer and telecommunications systems emerge in real-time systems, where expiration of a time-out can depend on a peak workload [9], [12], and in distributed database systems, where fragments and copies of logical files depend on each other as to the failure rate figures [10]. For example, when multiple physical copies of a file exist in a distributed database, all the operations on a failed copy can be transferred to another copy (possibly stored in another computer) which is still available. The stress on the file is then increased both owing to the load increase on the disk unit and to an increased danger to data integrity.

The possibility exists to group and classify the components of an information system, not on the basis of their physical nature, but on the ground of their reliability behavior with each class being represented with the same model. Considering the examples given above, we can make a parallel between the copies of a same file in a distributed database, and the different combinations of transmission links which provide alternative paths between two users of a telecommunication network. Transactions can be routed to the least loaded copy as well as messages can be routed through the least loaded path. Overloading a resource—file copy or transmission line—results in both cases in a higher response time, which in a real-time system can, in turn, result in a failure (Fig. 4).

If the routing mechanism is not properly controlled and tuned, a complete black-out of the system can be induced, at high global workloads, by instability in the feedback loop of the routing mechanism itself! (Fig. 5). A very similar behavior is shown by a set of ac electric-power generators under a load-dispatching algorithm. These cases represent sets of equivalent interchangeable components managed by a load control algorithm; in a certain sense we can establish a kind of "morphism" among them. In any case the failure rate of a component is largely determined by the behavior of the algorithm as a consequence of the system workload [12].

Rather surprisingly, some investigations seem to show that hardware electronic components, like CPU's, have load-dependent failure rates too [17].

Owing to the large number of components in an information system and to the computational complexity of the algorithms for computing the overall availability, hierarchical structuring shall be used to model the behavior of subassemblies of growing complexity.

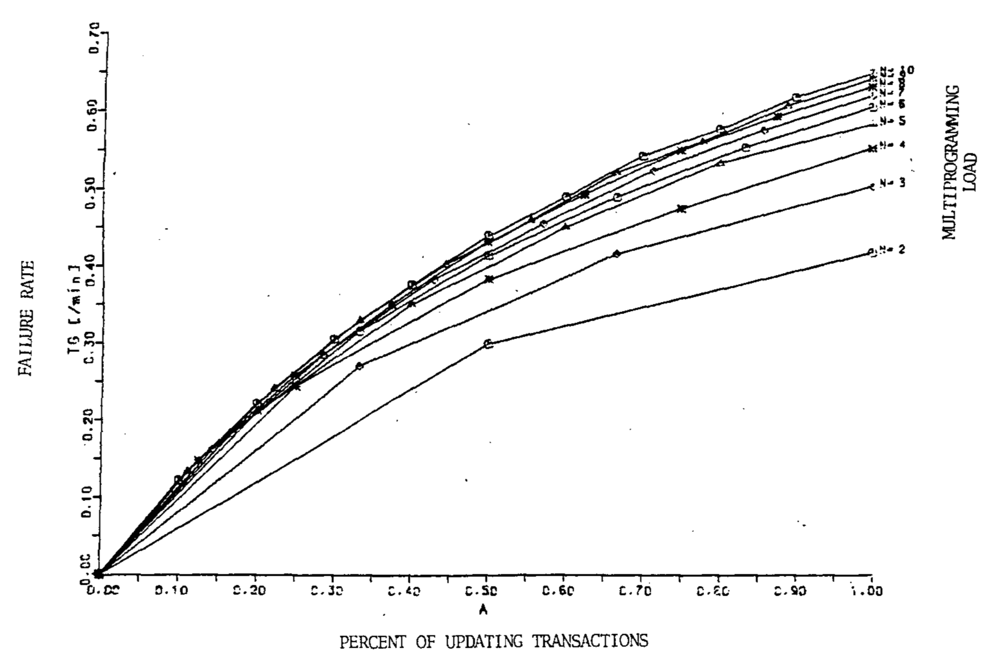

Fig. 3. Semaphore failure rate.

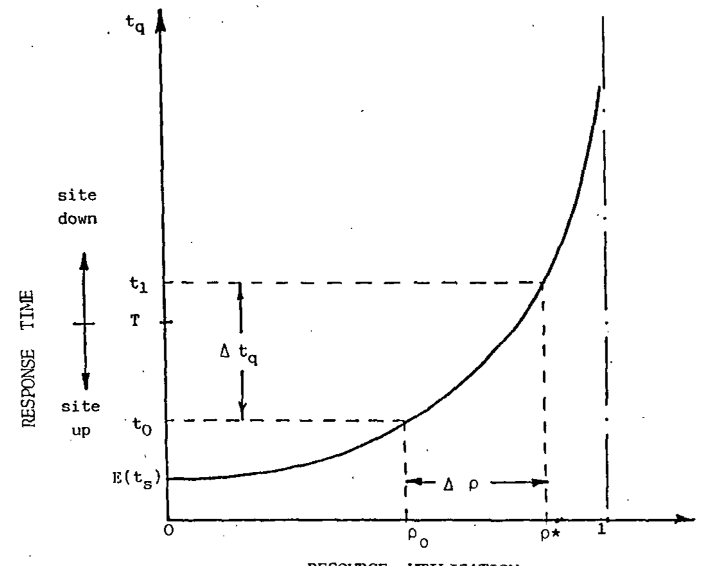

Fig. 4. Example of the response time versus load relation.

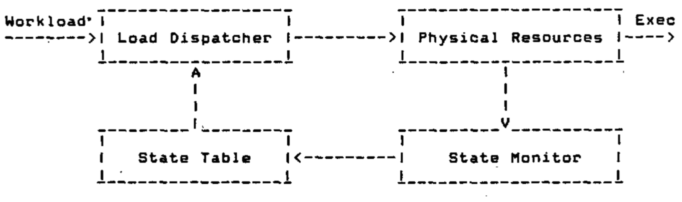

Fig. 5. The feedback loop.

LEVEL 5	Information System
LEVEL 4 system control	concurrency control, routing, transaction commitment, etc.
LEVEL 3 functional units	CPU, I/O subsystem, complex software packages, etc.
LEVEL 2 subassemblies	electronic cards, communication subsystems, simple programs, etc.
LEVEL 1 simple components	chips, transmission lines, subroutines, etc.

Fig. 6. A layered model.

Fig. 6 shows a possible layered description of an information system. At each abstraction level, the reliability parameters of its components are to be evaluated on the basis of the values computed at the lower level. The input to the lowest level is constituted by the manufacturer's data, while the output of the upper level is representative of the behavior of the whole information system.

B. Application Programs

This box in Fig. 1 represents the workload of the information system. Research in workload characterization is classical in Performance Evaluation studies. Models of the workload at some computing centers are built in order to study how to optimize the throughput and/or the response time by choosing, for instance, an optimum scheduling policy. Paging overhead in a virtual memory environment is deeply affected by the features of the applied workload.

In a reliability study, the workload on each resource of the information system is relevant for determining its failure rate, as we mentioned in Section II-A [9], [12]. However, in a complex system, the actual load on each component can not always be directly determined, since the application programs (transactions) are often written in terms of some logical resource set, which is mapped on physical resources only at compile time or, even later, at execution time.

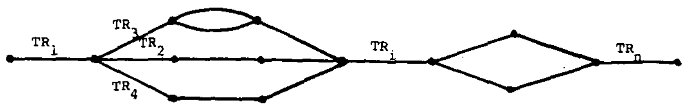

Fig. 7. Example of a structure graph.

These mapping rules are usually called *execution strategies*. In information systems, many possible execution strategies exist for performing a given transaction, and research is being carried on to find the optimal ones with respect to some performance goals. Moreover, mainly in distributed information systems, we can take advantage of the possible replication of some resources either for enhancing some performance indexes (e.g., response time) or for using them as back-up resources in case of failure of some component of the same type (e.g., rerouting of messages in computer networks or of transactions in multicopy distributed databases) [15]. In this way we can obtain gracefully degrading systems into which performance can be traded for availability or vice versa.

Therefore, as far as the operation of the system is concerned, we shall define a primary strategy (e.g., for optimizing performance). In addition, a set of fault-tolerant secondary strategies will be proposed with which the system could still operate, even if at reduced performance levels.

Physical resources for performing a transaction can be represented in a directed acyclic graph, called the *Structure Graph*. In this graph the unique (sets of) resources needed for the transaction processing are represented by serial edges, while (sets of) resources which can be alternative to each other are represented by parallel edges. The first node represents the transaction startpoint, the last one represents transaction commitment. Each edge is labeled with the transition rates TR_i of the involved resource (Fig. 7).

The set of all the execution strategies defines the rules for interconnecting the system resources and building the Structure Graph. *Each strategy can be described then as a path on the Structure Graph.* The choice of a strategy defines, together with the external workload, the actual load on each component, so allowing the evaluation of the L-dependent failure rates [12].

In addition to the execution strategies, other algorithms and rules must be defined for fault or error detection and for repairing them. This is one of the aspects where the behavior of hardware and software diverges the most. The possibility of using multiple redundancy techniques for software modules has hardly been explored [17], while replacement policies seem unapplicable for software failures prevention.

Many studies have been done on fault detection and recovery algorithms and techniques. They have been compared in terms of the classes of failures they can recover from, and of the overhead they use during normal operation, expressed for instance as number of messages exchanged among the nodes of a distributed system, or the number of read/write operations on log files and so on [3], [4]. Distributed computing systems and databases have been particularly studied as to the aspects of telecommunications induced failures—where the system can become partitioned into two (or more) noncommunicating parts and data consistency must be restored among them upon recovery—and as to how to agree on multiple partners decisions. Voting algorithms and Bizantine agreement protocols are but examples of such techniques.

However, while the time to repair a hardware component can often be considered fairly independent of the failure mechanism (the component is just substituted and possibly repaired off-line), this hypothesis does not hold for software and algorithmic components, whose repair time does depend on the kind of fault. In some cases, the repair rate μ of a component can be evaluated as a linear function of the repair rates μ_i for each type of fault the component can undergo, weighted with the probability γ_i of occurrence of that fault, e.g.,

$$\mu = \sum_i \gamma_i * \mu_i; \qquad \sum_i \gamma_i = 1$$

We are interested in recovery strategies for determining the "repair rates" which can be expected from them. When we shall be able to give some figures for the repair rates obtained with a given recovery strategy, which involves algorithms and both hardware and software components, we shall be able to cope wih them in an integrated way. In fact, the Information System Availability can be expressed as the probability that the Structure Graph be connected [2], [11].

C. Service Levels

To compare different systems, a unified metrics must be defined. The last box in Fig. 1 is concerned with the definition of some indexes which could provide a quantitative reference for the behavior of a gracefully degradable system. Several proposals have been made since 1978 [1], [5], [8], and all of them agree on establishing some thresholds—generally a couple—to separate the 100 percent performance and availability operation states, from the degraded ones (either from the performance or from the availability point of view), and eventually from the faulty ones.

For example, in an airport information system, one could expect that a passenger check-in operation would have a 2 s response time when fully operational. One could bear that, upon failure of some component, the response time is increased to 5 s (degraded operation). However it is clear that, would the response time rise to 2 min, the system would become useless, even if still in operation.

Fig. 8 shows the zones which result from choosing two thresholds (p_1, p_2) for performance and two (a_1, a_2) for availability. Let us suppose that each component has only two states (working or failed) and let us define a state of the system, as far as performance/availability is concerned, as the set of the states of its components. To each state a processing power can be associated; let us sort the

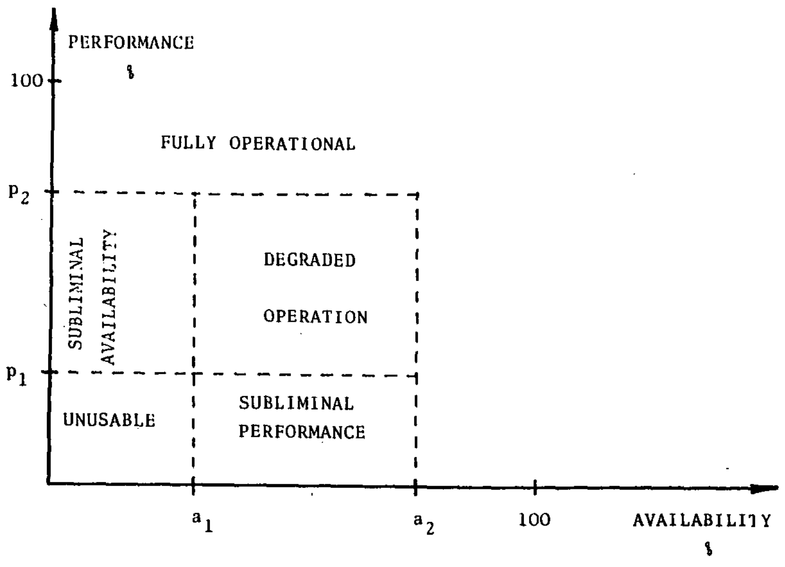

Fig. 8. Thresholds in performance and availability.

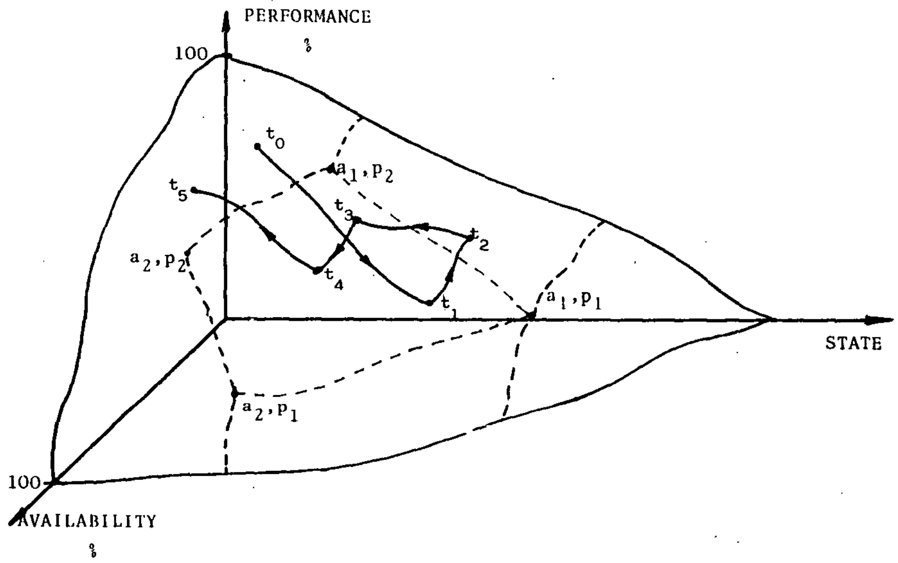

Fig. 9. The space of the service levels.

states in a nonincreasing processing power order. Then a surface (actually a set of discrete points) results, as in Fig. 9. As long as components fail and are repaired, the system's behavior is described by a line (trajectory) on this surface. Segments of trajectories which are orthogonal to the state axis represent situations in which, for the same system state, some power (resource) is removed from fault-tolerance functions in favor of enhancing performance, or vice versa.

There is no general agreement on which metrics should be used. In her proposal, Beaudry chooses indexes which reflect the system computational capacity, i.e., the capability of the system to perform a given amount of work [1]. In particular, she defines:

• the *computation reliability* R^* (t, T) as the probability that, at time t, the system is in an unfailed state and correctly executes a task of length T started at time t, given an initial system state;

• the *computation availability* a_c as the expected value of the computation capacity of the system at time t or in steady-state operation;

• the *computation capacity* $\alpha_i > 0$ as the amount of useful computation per unit time (e.g., mips) of the system in state i;

• the *capacity threshold* t_c as the time at which the computation availability reaches a specific value;

• the *available computation* in state i as $c_a = \alpha_i * t$.

Using these variables she transforms a time-domain representation of the system into a computation-domain representation, being able to evaluate such quantities as the MCBF (mean computation before failure), i.e., the expected amount of computation available on the system before its first failure, given an initial state.

Other authors choose user-measurable parameters; Huslende [5], for instance, defines a stochastic variable $H(t)$ = system performance at time t normalized with respect to performance in the fault-free state ($0 \leq H(t) \leq 1$).

Using $H(t)$, he derives quantities such as the *performance reliability* $R(a, t) = \{\text{prob}[H(\tau) \geq a], \forall \tau \geq t \mid H(0) = 1\}$, and the *performance availability* $A(a, t) = \text{prob}[H(t) > a]$. Referring to the thresholds in Fig. 8, quantities such as

$$\text{MTUO} = \text{medium time to unreliable operation} = \int_0^\infty R(a_2, t)\, dt$$

and

$$\text{MTNO} = \text{medium time to no operation} = \int_0^\infty R(a_1, t)\, dt$$

can be computed.

Some other interesting approaches such as the concept of *performability*, which tend to include both views, taking into account also some cost parameters, seem to be still too complex to be of practical relevance to the information system field [8].

Therefore it seems that also in this area there is *considerable room for research to obtain generally applicable significant indexes.*

III. Conclusions

The possibility has been shown to unify several different research areas which are more or less directly relevant to the evaluation of performance and availability of information systems.

An abstract description of the component's behavior could allow an integrated treatment of the computing and of the communication equipment independently of their technological nature (hardware, software, algorithmic). A structure graph, representing the interconnections of the components in the information system, together with their transition rates from working to failed states and vice versa are the tools for providing a quantitative evaluation of the information system availability. Owing to the complexity of the system global description, a layered approach should be used by evaluating subsystems of growing functional complexity.

Research efforts have started in several areas such as the evaluation of transition rates of control algorithms and of software components, and in providing the user with effective metrics for assessing and comparing performance, fault-tolerance, and cost of different systems.

References

[1] M. D. Beaudry, "Performance-related reliability measures for computing systems," *IEEE Trans. Comput.*, vol. C-27, no. 6, pp. 540–547, 1978.

[2] R. E. Barlow and F. Proschan, *Mathematical Theory of Reliability*. New York: Wiley, 1965.

[3] P. Dadam and G. Schlageter, "Recovery in distributed databases based on non-synchronized local checkpoints," in *Information Processing 80*, S. H. Lavington, Ed. Amsterdam, The Netherlands: North-Holland, IFIP, Tokyo, Japan, 1980, pp. 457–462.

[4] H. Garcia-Molina and F. M. Pittelli, "Implementing reliable distributed computing systems," Dep. Elec. Eng. Comput. Sci., Princeton Univ., Princeton, NJ, Tech. Rep., Jan. 1983.

[5] R. Huslende, "A combined evaluation of performance and reliability for degradable systems," in *ACM-SIGMETRICS Conf. on Measurement and Modeling of Comput. Syst.*, 1981, pp. 157–163.

[6] K. B. Irani and H. L. Lin, "Queuing network models for concurrent transaction processing in a database system," in *Proc. ACM-SIGMOD Conf.*, Boston, MA, 1979, pp. 134–142.

[7] W. Kiessling and G. Landherr, "A quantitative comparison of lock-protocols for centralized databases," in *Proc. 9th VLDB Conf.*, Florence, Italy, 1983, pp. 120–130.

[8] J. F. Meyer, "Closed-form solutions of performability," *IEEE Trans. Comput.*, vol. C-31, no. 7, pp. 648–657, 1982.

[9] J. A. Munarin, "Dynamic workload model for performance/reliability analysis of gracefully degrading systems," in *Proc. FTCS-13*, Milano, Italy, 1983, pp. 290–295.

[10] G. Martella, B. Pernici, and F. A. Schreiber, "An availability model for distributed transaction systems," *IEEE Trans. Software Eng.*, vol. SE-11, no. 5, pp. 483–491, 1985.

[11] G. Martella, B. Ronchetti, and F. A. Schreiber, "Availability evaluation in distributed database systems," *Performance Evaluation*, vol. 1, no. 3, pp. 210–211, 1981.

[12] F. A. Schreiber, "State dependency issues in evaluating distributed database availability," *Comput. Networks*, vol. 8, no. 3, pp. 187–197, 1984.

[13] J. D. Spragins, "A fast algorithm for computing availability in networks with dependent failures," in *Proc. IEEE INFOCOM'84*, San Francisco, CA, 1984.

[14] ——, "Limitations of current telecommunication network reliability models," in *Proc. IEEE Globecom'84 Conf.*, Atlanta, GA, 1984, pp. 836–840.

[15] G. M. Sacco and S. B. Yao, "Query optimization in distributed database systems," in *Advances in Computers*, vol. 21. New York: Academic, 1982.

[16] F. A. Schreiber, C. Calvi-Parisetti, and M. Visioli, "A model for evaluating the availability of (r, x) semaphores in database concurrency control," to be published.

[17] Special Issue on Software Reliability—Part 1, *IEEE Trans. Software Eng.*, vol. SE-11, no. 12, 1985.

Fabio A. Schreiber (SM'85) received the Dr.-Ing. degree in electronic engineering from the Politecnico di Milano, Italy, in 1969.

Since then, he has been with the Computer Science Laboratory at the Dipartimento di Elettronica of the Politecnico di Milano, as an Assistant and then as an Associate Professor of Applied Electronics. Since 1981, he has been a Full Professor of Computer Science at the Department of Mathematics, University of Parma. His main research interests are in the field of distributed informatics and information systems. His current researches include performance and reliability evaluation of distributed computing systems. He has been involved in the development of a distributed DBMS sponsored by the Italian National Science Council. He has authored more than fifty papers on these topics and has been invited as a speaker to many workshops and conferences. He has also consulted for several companies and university installations on advanced system design and is the Editor-in-Chief of *Rivista di Informatica*, the journal of the Italian Association for Automatic Computing.

Dr. Schreiber is a member of the Association for Computing Machinery and the Associazione Italiana per l'Informatica e il Calcolo Automatico.