Proceedings of the 1989 Winter Simulation Conference
E.A. MacNair, K.J. Musselman, P. Heidelberger (eds.)

ESTIMATING SYSTEM AVAILABILITY AND RELIABILITY

Ambuj Goyal

IBM Research Division
Thomas J. Watson Research Center
Yorktown Heights, New York 10598

ABSTRACT

This paper deals with methods for constructing and solving large Markov chain models of computer system availability and reliability. A set of powerful high level modeling constructs is discussed that can be used to represent the failure and repair behavior of the components that comprise a system, including important component interactions. If time independent failure and repair rates are assumed then a time homogeneous continuous time Markov chain can be constructed automatically from the modeling constructs used to describe the system. Since, the size of Markov chains grows exponentially with the number of components modeled, simulation appears to be a practical way for solving models of large systems. However, the standard simulation takes very long simulation runs to estimate availability and reliability measures because the system failure event is a rare event. Therefore, variance reduction techniques which can aid in computing rare-event probabilities quickly are of interest. Specifically, the Importance Sampling technique has been found to be most useful. The modeling language and the simulation methods discussed in this paper have been implemented in a program package called the System Availability Estimator (SAVE).

1. INTRODUCTION

Fault-tolerant computing has been applied to two fundamentally different classes of applications. One deals with mission oriented systems with high reliability requirements such as space computers, avionics systems and ballistic missile defense computers [5, 8]. For the mission to succeed the system must not fail during the mission time. Hence the probability that the system does not fail during the mission time, i.e. the system reliability, is a measure of interest. The other class of applications deals with continuously operating systems with high availability requirements such as telephone switching systems, general purpose computer systems, transaction processing systems (e.g. airline reservation systems) and communication network computers. For such systems, system failures can be tolerated if they occur infrequently and they result in short system down times. For such systems, the expected fraction of time the system is operational, i.e. the system availability, is a measure of interest. As in [17], we use the term dependability to refer to both reliability and availability, in general.

In this paper we will focus on continuous time Markov chain models of fault-tolerant computers, which are perhaps the most commonly used. From the modeling point of view a system consists of a collection of hardware and software components, each of which may be subject to failure, recovery and repair. Software components in operation can be modeled with constant failure rates, as described in [17]. Component interactions often have a substantial effect on system availability and must therefore be considered in addition to the individual component behaviors. The state space size of such models grows (often exponentially) with the number of components being modeled. Therefore, we provide a high level modeling language containing constructs which aid in representing the failure, recovery and repair behavior of components in the system as well as important component interactions. The challenge in developing such a language lies in making it comprehensive enough so that important system details can be modeled, but simple enough so that the Markov chain can be automatically constructed from the modeling language description. In Section 2 we describe such a modeling language. Appendix 1 contains a list of all the constructs. The language has been incorporated in a system dependability modeling program package, called the System Availability Estimator (SAVE) [9, 11]. SAVE automatically generates the Markov chain from the modeling language description.

Typically, numerical methods are used to solve these Markov chains. Although, many modeling packages have been built, e.g., [5] and [11], which incorporate numerical methods capable of computing steady-state as well as transient state probabilities of Markov chains

with thousands of states, the size of system modeled is typically small because the number of states in the system increases exponentially with the number of components. Techniques like state lumping and unlumping [10,23] and state aggregation and bounding [1, 22] can reduce the size of the state space substantially. However, large systems with a large number of redundant components are still out of the range of the solution capabilities of current numerical methods, primarily due to storage or computational limitations. On the other hand, simulation algorithms tend to be relatively insensitive to the size of the state space of the simulated Markovian model, both in terms of storage and computational requirements. However, standard simulation is inefficient in our setting because the principle focus of interest; namely, system failures, occur so infrequently in highly dependable systems. As a consequence, few system failures, if any, would be observed if standard simulation methods were to be used in our problem context. In Section 3, we focus on the variance reduction techniques used to improve the efficiency of the simulation methods implemented in SAVE. In Section 4, we illustrate the effectiveness of the variance reduction techniques using a simple example.

2. DEPENDABILITY MODELING LANGUAGE

The uses of system dependability models include comparing various fault tolerant design alternatives and for a particular design identifying any dependability bottlenecks that may require subsequent design improvements. A system model is an abstraction of the system where we ignore certain design details to reduce the size of the model, but we include other details to be able to study the various design tradeoffs. For example, when modeling a computer system each processor could be considered a component or at a lower level each of the logic modules that comprise a processor could be considered a component. In the latter case there will be many more components and the model will have a much larger state space. The level of detail chosen should depend on the questions to be addressed using the model and/or on the obtainable failure and repair data as well as on the resulting model size. Thus, if one wanted to determine the effect of a standby processor on system availability it might be better to consider processors to be components rather than the logic modules that comprise them.

In the remainder of this section we summarize the main constructs of the dependability modeling language which has been implemented in SAVE. (For completeness the entire syntax of the language is given in Appendix 1.) The most important construct in the modeling language is the COMPONENT construct which is used to model the failure and repair behavior of a component including its interactions with other components. In the modeling language we will use the term "failure" to refer to either an error or a failure and we will use the term "repair" to refer to any action that renders a "failed" component operational. Many types of repair are possible including automatic recovery, operator restart of a subsystem or the entire system and physical repair or replacement. In the modeling language a component is assumed to fail in possibly multiple modes and a different type of repair can be associated with each mode. The overall state of a single component is shown in Figure 1. Each type of repair can be performed by a different class of "repairman" and have a different repair rate. The repair strategies followed by a repairman in a class are discussed later in this section. The transition rates from operational to failed and failed to operational states are affected by component interactions and the repair strategies followed by the repairmen. In general, the COMPONENT construct describes not just a single component instance but a single component type, where components of the same type are identical as far as their failure and repair behaviors are concerned.

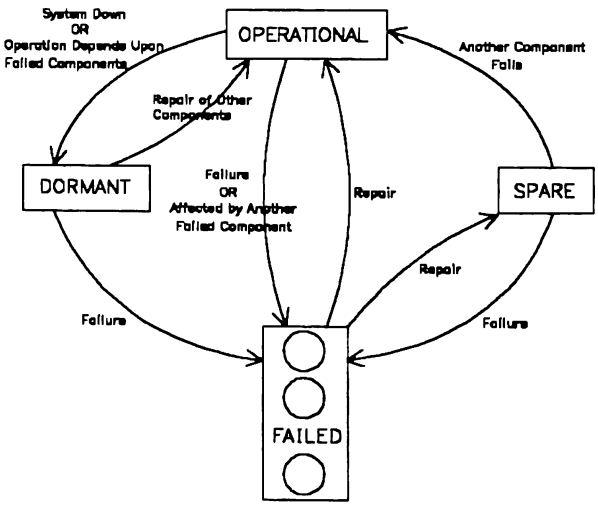

Figure 1. States of a single component type.

In real systems the failure behavior of a component can be more complicated than we have described so far. Operational and repair dependencies, failure/error propagation, and differences between dormant, spare and operational states of a component affect the failure

process. To capture such complex failure behavior of a component, we add two more states, namely spare and dormant, in our component model as shown in Figure 1. Now, a component can be in one of four states: operational, failed (could be in one of many modes), spare and dormant. In the latter three states the component is considered unoperational. A component is said to be dormant when it is not operating because its operation depends upon some other components which are unoperational or the whole system is unoperational. For example a processor is dormant when its power supply has failed. The component may fail when it is either operational, spare, or dormant and its failure rate may be different in the three states. A component may also fail if it is affected by the failure of other components in the system (failure propagation).

We next discuss the REPAIRMAN CLASS construct that is used to provide information about the actions taken by the repairmen in each repairman class. As was mentioned earlier in this section different types of repair, e.g. automatic recovery, operator restart and physical repair or replacement, can be associated with the different failure modes of a component. Each type of repair can be performed by a different repairman class. A repairman class could represent operators who do restart, field engineers who do physical repair or replacement of the parts of the system, and either hardware or software, that does automatic recovery. A repairman class is specified by assigning it a name, giving the number of repairmen in the class, the repair strategy used and, if the repair strategy is based on priorities, assigning a priority to each component type that can be repaired by the repairman class. We use only a preemptive resume type repair strategy so that together with the assumption of time independent failure and repair rates, the state of the Markov chain can be represented by a concatenation of the states (operational, failed, spare or dormant) of each component.

Next we discuss EVALUATION CRITERIA construct which is used to specify the conditions under which a system is considered operational or available to perform service. A system is considered to be available if specified subsets of its components are operational. This can be expressed using a reliability block diagram or equivalently a Boolean expression over the component names involving "and" and "or". Alternatively it may be easier to specify conditions under which the system is unavailable by specifying subsets of its components that are unoperational. This can be done using a fault tree or equivalent Boolean expression. Clearly connectivity or lack of connectivity between components can affect the above Boolean specification and hence the system dependability measures.

3. SIMULATION OF DEPENDABILITY MODELS

Different measures are used to evaluate the modeled systems depending upon whether they are mission oriented systems or continuously operating systems. Some of the dependability measures of interest are steady-state availability, reliability, mean time to failure, expected interval availability and the complementary distribution of interval availability (i.e., the probability that a system would achieve a higher interval availability than a specified value between 0 and 1.) Similar measures have also been constructed for degradable systems, e.g., steady-state performance and distribution of performance over a time interval [21]. Detailed surveys of these modeling techniques and the dependability measures calculated appear in [8, 20].

As mentioned in Section 1, Monte Carlo simulation is the most appropriate technique for solving dependability models of large systems. Simulation is especially useful for those models for which the transition rate matrices exceed the available storage. By nature, this approach has the immediate advantage of having relatively small storage requirements. On the other hand, since the failure events are rare events, it is apparent that the analysis by simulation of large models with a high degree of redundancy will require many regenerative cycles or many long independent replications in order to attain reasonable confidence intervals [8, 19]. Our goal is to obtain variance reduction methods that are applicable to a broad class of models. Specifically, we are interested in models defined by the reliability and availability modeling language described in Section 2, so that the techniques can be implemented in a software package and made available to designers in an automatic and transparent fashion. For highly reliable and highly available systems, it is usual for the repair/recovery rates of components to be orders of magnitude larger than the failure rates, and in these circumstances the use of importance sampling variance reduction techniques [14, 16] can be very effective in reducing the simulation run length significantly.

Importance sampling for rare event simulation has been used successfully in [3], [18], [24], [27] and [26]. Proper selection of the importance sampling distribution makes the rare events more likely to occur; this results in a variance reduction. The key, of course, is

to choose a good importance sampling distribution. The theory of large deviations was used in [3], [26] and [24, 27] to select an effective distribution for problems arising in Markov chains with "small increments", random walks, and queueing networks, respectively. Effective heuristics were used in [18] to select importance sampling distributions for reliability estimation in large models of machine repairman type. In the SAVE package, we have used the importance sampling techniques described in [2], [12], [13] and [25]. These techniques are capable of estimating both steady-state and transient measures simultaneously from the simulation, orders of magnitude faster than ordinary simulation. These techniques are very robust in the sense that they are applicable to a broad class of dependability models.

For measures such as the steady-state availability and the mean time to failure (MTTF), the estimators are based on combining regenerative simulation [4] with importance sampling. As recommended in [15], we first transform the continuous time chain into an appropriate discrete time Markov chain. Simple "failure biasing" techniques are then used to select importance sampling distributions as described in [2] and [18]. For regenerative systems, steady state performance measures can be expressed as a ratio. In [2], a single importance sampling distribution was used to estimate both the numerator and the denominator of this ratio. The distribution used in [2] is dynamic in the sense that it does not correspond directly to a time homogeneous Markov chain. This technique was called Dynamic Importance Sampling (DIS). In [12], different dynamic importance sampling distributions were used to estimate the numerator and denominator independently which resulted in additional variance reduction (actually, importance sampling was not used at all to estimate the denominator). This technique was called Measure Specific DIS (MSDIS).

Direct application of these techniques does not yield a significant variance reductions for estimating the MTTF. An intuitive reason for this is as follows. When we make failure events occur more often by choosing an appropriate importance sampling distribution, the value of the estimator ends up smaller than the actual value and, in addition, the likelihood ratio is less than one. This actually ends up producing variance rather than reducing it. However, when the MTTF is formulated as a ratio of two expectations (both are estimated using a regenerative simulation), then significant variance reductions can be achieved using the importance sampling techniques. Further details of this method are given in [13, 25]. The theoretical basis for selecting specific importance sampling distributions, bias expansions, run-length allocations for MSDIS and the implementation details are discussed in [13].

For transient measures, such as reliability, interval availability and distribution of interval availability, the estimators are obtained by independently replicating observations based on combining "conditioning" (e.g., [6, 7]) or "forcing" (e.g., [18]) methods with importance sampling. While no analytic result exists for comparing conditioning with forcing, the conditioning approach has several advantages over the forcing approach. First, with forcing, different holding times must be generated for each value of t, the observation period, for which the transient measure is to be estimated. Because of sampling errors, the estimates of the transient measure may not be monotonic in t. Using the conditional approach, simultaneous, monotonic estimates of the transient measures are obtained. Second, with forcing, different conditional holding time distributions are used and different likelihood ratios must be maintained for each value of t for which the transient measure is to be estimated. This is not necessary in the conditional approach. Thus, it has computational time advantage when the transient measure is computed for multiple values of t simultaneously. Once again, the estimators, the stopping rules and the implementation details for the transient measures are given in [13].

4. AN ILLUSTRATIVE EXAMPLE

The dependability modeling language of Section 2 and the variance reduction techniques discussed in Section 3 have been implemented in the SAVE package [9, 11] so that large availability models can be simulated. We use a simple example to illustrate the effectiveness of the variance reduction techniques. The block diagram for the example is shown in Figure 2 and the SAVE language description in Appendix 2. The following heuristics were used to select the importance sampling distributions for the embedded Markov chain as suggested in [2, 12, 18]. We assigned a higher combined probability *bias*1 to the failure transitions in all the states where both failure and repair transitions are feasible, and once a system failure state was entered, we returned to the original probability distributions. This method was called the *Bias1/Ratio* method, or simply the *Bias1* method. We also used another method called the *Bias1/Bias2*, where besides *bias*1, we assigned a higher combined probability *bias*2 to those failure tran-

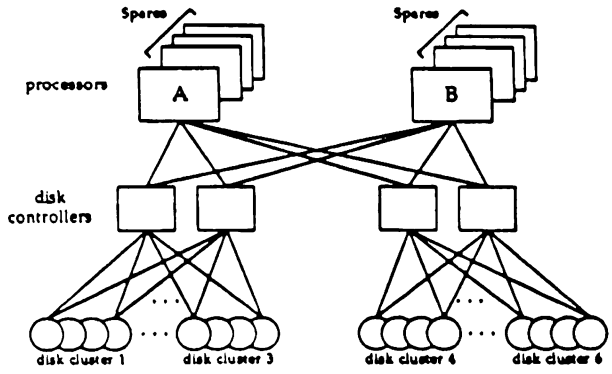

Figure 2. A block diagram of the computing system modeled.

sitions which correspond to component types which have at least one component of their type already failed. This exhausts the redundancy quickly and has much better chance of selecting the most likely path to system failure. Based on empirical results obtained in [2], [12] and [25], the values for *bias*1 and *bias*2 were selected as follows: for DIS, .5 and .5, and for MSDIS, .9 and .9. Also for MSDIS, we assigned 10% of the total events to estimate the denominator for unavailability as suggested in [13].

The example of Figure 2 (Appendix 2) was simulated using the SAVE package. We performed the following 4 experiments for estimating unavailability. Each experiment was run for approximately 100,000 events. The percentage relative half-width of confidence intervals, defined to be 100% times the confidence interval half-width divided by the point estimate, for the direct simulation, the *Bias1* method with DIS, the *Bias1/Bias2* method with DIS and the *Bias1/Bias2* method with MSDIS were 27.1%, 7.6%, 2.7% and 1%, respectively. Coverage experiments done in [13] show that these orders of magnitude variance reductions are realizable with very short runs - as small as a few thousand events. Similar results have been obtained for MTTF and all the other transient measures described in Section 3. These results are summarized in [13].

5. SUMMARY

System dependability is becoming an increasingly important factor in evaluating the behavior of commercial computer systems. This is due to the increased dependence of enterprises on continuously operating computer systems and to the emphasis on fault tolerant designs. Models of such system will continue to get more complex as the size and the complexity of these systems increase and as more system details are included in the model. Although, the theoreticians will continue to create state-of-the-art modeling techniques, the practitioners will stay away from using them unless a method is discovered to bridge the gap between the two. The SAVE modeling package is an attempt to do so.

SAVE incorporates a high level modeling language with simple and powerful modeling constructs. These constructs are able to describe the failure and repair behavior of components as well as their interdependencies very easily. We have built very large models (hundreds of components) reasonably quickly using the SAVE language.

Since, SAVE is specifically designed for practitioners, it is important to incorporate most general and robust solution methods which apply to a broad class of models that can be generated using the language. Special cases are rare, and therefore, the simulation methods which work for a restricted class of models are of a lesser interest from the point of view of a package like SAVE. The importance sampling distributions we have selected are very simple and are applicable to all dependability measures mentioned in Section 3. This results in efficient computation of all the measures simultaneously from a single simulation run. SAVE methods have been used to construct and solve thousands of models of computer and communication systems.

REFERENCES

[1] Bobbio, A. and Trivedi, K. S. (1986). An Aggregation Technique for the Transient Analysis of Stiff Markov Chains. *IEEE Transactions on Computers* **C-35**, 803-814.

[2] Conway, A.E. and Goyal, A. (1987). Monte Carlo Simulation of Computer System Availability/Reliability Models. *Proceedings of the Seventeenth Symposium on Fault-Tolerant Computing*. Pittsburgh, Pennsylvania, 230-235.

[3] Cottrell, M., Fort, J.C. and Malgouyres, G. (1983). Large Deviations and Rare Events in the

Study of Stochastic Algorithms. *IEEE Transactions on Automatic Control* **AC-28**, 907-920.

[4] Crane, M.A. and Iglehart, D.L. (1975). Simulating Stable Stochastic Systems, III: Regenerative Processes and Discrete Event Simulations. *Operations Research* **23**, 33-45.

[5] Dugan, J.B., Trivedi, K.S., Smotherman, M.K. and Geist, R.M. (1986). The Hybrid Automated Reliability Predictor. *Journal of Guidance, Control, and Dynamics* **9, 3**, 319-331.

[6] Fox, B.L. and Glynn, P.W. (1986). Discrete-Time Conversion for Simulating Semi-Markov Processes. *Operations Research Letters* **5**, 191-196.

[7] Fox, B.L. and Glynn, P.W. (1988). Discrete-Time Conversion for Finite-Horizon Markov Processes. Technical Report No. 790, School of Operations Research and Industrial Engineering, Cornell University, Itahaca, New York.

[8] Geist, R.M. and Trivedi, K.S. (1983). Ultra-High Reliability Prediction for Fault-Tolerant Computer Systems. *IEEE Transactions on Computers* **C-32**, 1118-1127.

[9] Goyal, A., Carter, W.C., de Souza e Silva, E., Lavenberg, S.S., and Trivedi, K.S. (1986). - The System Availability Estimator. *Proceedings of the Sixteenth Symposium on Fault-Tolerant Computing*. Vienna, Austria, 84-89.

[10] Goyal, A., Lavenberg, S.S., and Trivedi, K.S. (1987). Probabilistic Modeling of Computer System Availability. *Annals of Operations Research* **8**, 285-306.

[11] Goyal, A. and Lavenberg, S.S. (1987). Modeling and Analysis of Computer System Availability. *IBM Journal of Research and Development*, **31 6**, 651-664.

[12] Goyal, A., Heidelberger, P. and Shahabuddin, P. (1987). Measure Specific Dynamic Importance Sampling for Availability Simulations. *1987 Winter Simulation Conference Proceedings*. A. Thesen, H. Grant and W.D. Kelton (eds.). IEEE Press, 351-357.

[13] Goyal, A., Shahabuddin, P., Heidelberger, P., Nicola, V.F. and Glynn, P. (1989). A Unified Framework for Simulating Markovian Models of Highly Dependable Systems. IBM Research Report RC 14772, Yorktown Heights, New York.

[14] Hammersley, J.M. and Handscomb, D.C. (1964). *Monte Carlo Methods*. Methuen, London.

[15] Hordijk, A., Iglehart, D.L. and Schassberger, R. (1976). Discrete Time Methods for Simulating Continuous Time Markov Chains. *Adv. Appl. Prob.* **8**, 772-788.

[16] Kahn, H. and Marshall, A.W. (1953). Methods of Reducing Sample Size in Monte Carlo Computations. *Journal of Operations Research Society of America* **1, 5**, 263-278.

[17] Laprie, J.C. (1984). Dependability Evaluation of Software Systems in Operation. *IEEE Transactions Software Engineering* **SE-10**, 701-713.

[18] Lewis, E.E. and Böhm, F. (1984). Monte Carlo Simulation of Markov Unreliability Models. *Nuclear Engineering and Design* **77**, 49-62.

[19] Liceaga, C.A. and Siewiorek, D.P. (1986). Towards Automatic Markov Reliability Modeling of Computer Architectures. *NASA Technical Memorandum* **89009**.

[20] Johnson, Jr. A. M. and Malek, M. (1988). Survey of Software Tools for Evaluating Reliability, Availability, and Serviceability. *ACM Computing Surveys* **20, 4**, 227-269.

[21] Meyer, J.F. (1980). On Evaluating the Performability of Degradable Computing Systems. *IEEE Transactions on Computers* **C-39, 8**, 720-731.

[22] Muntz, R.R., de Souza e Silva, E. and Goyal, A. (1989). Bounding Availability of Repairable Computer Systems. *Proceedings of the ACM Sigmetrics and Performance'89*. Berkeley, California. Also, to appear in the December 1989 issue of *IEEE Transactions on Computers*.

[23] Nicola, V.F. (1989). Lumping in Markov Reward Processes. IBM Research Report RC 14719, Yorktown Heights, New York.

[24] Parekh, S. and Walrand, J. (1989). A Quick Simulation Method for Excessive Backlogs in Networks of Queues. *IEEE Transaction on Automatic Control* **34**, 1, 54-66.

[25] Shahabuddin, P., Nicola, V.F., Heidelberger, P., Goyal A. and Glynn P.W. (1988). Variance Reduction in Mean Time to Failure Simulations. *1988 Winter Simulation Conference Proceedings*. M.A. Abrams, P.L. Haigh and J.C. Comfort (eds.). IEEE Press, 491-499.

[26] Siegmund, D. (1976). Importance Sampling in the Monte Carlo Study of Sequential Tests. *The Annals of Statistics* **4**, 673-684.

[27] Walrand, J. (1987). Quick Simulation of Rare Events in Queueing Networks. *Proceedings of the Second International Workshop on Applied Mathematics and Performance/Reliability Models of Computer/Communication Systems*. G. Iazeolla, P.J. Courtois and O.J. Boxma (eds). North Holland Publishing Company, Amsterdam, 275-286.

APPENDIX 1: SAVE LANGUAGE

```
MODEL:<modelname>
   METHOD:<NUMERICAL|MARKOV|COMBINATORIAL|SIMULATION>
*
PARAMETERS: <parameter-name>, <parameter-name>,..
*
CONSTANTS: <constant-name>, <constant-name>, ...
   CONSTANT-NAME: <constant-value|expression>
   CONSTANT-NAME: <constant-value|expression>
   .
*
LISTS: <list-name>, <list-name>, ...
   LIST-NAME: <comp-name><(no.-of-comps)>, ...
   LIST-NAME: <comp-name><(no.-of-comps)>, ...

*
COMPONENT:                    <comp-name><(no.-of-comps)>
  SPARES:                     <no.-of-spares>
  SPARES FAILURE RATE:        <expression>
  OPERATION DEPENDS UPON:     <comp-name><(no.)>, ..
  REPAIR DEPENDS UPON:        <comp-name><(no.)>, ...
  DORMANT WHEN SYSTEM DOWN:   <YES|NO>
  DORMANT FAILURE RATE:       <expression>
  FAILURE RATE:               <expression>, <expression>, ...
  FAILURE MODE PROBABILITIES:<prob-value>, ...
  REPAIR RATE:                <expression>, ...
  REPAIRMAN CLASS USED:       <class-name>, ...
  COMPONENTS AFFECTED:        <NONE|list-name|comp-name(<no.>)>, ...
      <LIST-NAME|COMP-NAME>: <affect-prob-val>, ...
      <LIST-NAME|COMP-NAME>: <affect-prob-val>, ...
COMPONENT:

*
EVALUATION CRITERIA: <ASSERTIONS|BLOCKDIAGRAM|FAULTTREE|PERFORMANCE>

*
REPAIRMAN CLASS:    <class-name>(<number>|UNLIMITED)
  REPAIR STRATEGY: <PRIORITY|ROS>
```

```
   COMPONENT-NAME: <priority-level>
   COMPONENT-NAME: <priority-level>
REPAIRMAN CLASS:

*
END
```

APPENDIX 2: EXAMPLE SYSTEM

```
 MODEL: Example
  METHOD: simulation
  CONSTANTS: procfr, cfr, dfr, rr1, rr2, c
     procfr: 1/2000
     cfr   : 1/2000
     dfr   : 1/6000
     rr1   : 1
     rr2   : 1/2
     c     : 0.99
*
  COMPONENT                       :  ProcA(2)
      DORMANT WHEN SYSTEM DOWN    :  NO
      FAILURE RATE                :  procfr
      FAILURE MODE PROBABILITIES:  0.5, 0.5
      REPAIR RATE                 :  rr1, rr2
      COMPONENTS AFFECTED         :  ProcB, ProcB
            ProcB: 1-c
            ProcB: 1-c
*
  COMPONENT                       :  ProcB(2)
      DORMANT WHEN SYSTEM DOWN    :  NO
      FAILURE RATE                :  procfr
      FAILURE MODE PROBABILITIES:  0.5, 0.5
      REPAIR RATE                 :  rr1, rr2
      COMPONENTS AFFECTED         :  ProcA, ProcA
            ProcA: 1-c
            ProcA: 1-c
*
  COMPONENT                       :  Cont1(2)
      DORMANT WHEN SYSTEM DOWN    :  NO
      FAILURE RATE                :  cfr
      FAILURE MODE PROBABILITIES:  0.5, 0.5
      REPAIR RATE                 :  rr1, rr2
*
  COMPONENT                       :  Cont2(2)
      DORMANT WHEN SYSTEM DOWN    :  NO
      FAILURE RATE                :  cfr
      FAILURE MODE PROBABILITIES:  0.5, 0.5
      REPAIR RATE                 :  rr1, rr2
*
  COMPONENT                       :  Diskc1(4)
      DORMANT WHEN SYSTEM DOWN    :  NO
      FAILURE RATE                :  dfr
```

```
      FAILURE MODE PROBABILITIES:  0.5, 0.5
      REPAIR RATE                :  rr1, rr2
*
  COMPONENT                      :  Diskc2(4)
      DORMANT WHEN SYSTEM DOWN   :  NO
      FAILURE RATE               :  dfr
      FAILURE MODE PROBABILITIES:  0.5, 0.5
      REPAIR RATE                :  rr1, rr2
*
  COMPONENT                      :  Diskc3(4)
      DORMANT WHEN SYSTEM DOWN   :  NO
      FAILURE RATE               :  dfr
      FAILURE MODE PROBABILITIES:  0.5, 0.5
      REPAIR RATE                :  rr1, rr2
*
  COMPONENT                      :  Diskc4(4)
      DORMANT WHEN SYSTEM DOWN   :  NO
      FAILURE RATE               :  dfr
      FAILURE MODE PROBABILITIES:  0.5, 0.5
      REPAIR RATE                :  rr1, rr2
*
  COMPONENT                      :  Diskc5(4)
      DORMANT WHEN SYSTEM DOWN   :  NO
      FAILURE RATE               :  dfr
      FAILURE MODE PROBABILITIES:  0.5, 0.5
      REPAIR RATE                :  rr1, rr2
*
  COMPONENT                      :  Diskc6(4)
      DORMANT WHEN SYSTEM DOWN   :  NO
      FAILURE RATE               :  dfr
      FAILURE MODE PROBABILITIES:  0.5, 0.5
      REPAIR RATE                :  rr1, rr2
*
  EVALUATION CRITERIA: blockdiagram
      Cluster1: Diskc1(3) and Diskc2(3) and Diskc3(3)
      Cluster2: Diskc4(3) and Diskc5(3) and Diskc6(3)
      ProcA and ProcB and Cont1 and Cont2 and Cluster1 and Cluster2
*
  REPAIRMEN: 1
  REPAIR STRTATEGY: ROS
*           (Random Order Service)
  END
```

AUTHOR'S BIOGRAPHY

AMBUJ GOYAL received the B.Tech. degree from the Indian Institute of Technology, Kanpur, and the M.S. and Ph.D. degrees from The University of Texas at Austin. At present, he is a Research Staff Member and manages Fault-Tolerant Systems Theory Group at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York. His research interest include fault-tolerant computing, computer architecture, communication networks and performance evaluation. Dr. Goyal is a member of the IEEE Computer Society.

Ambuj Goyal
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, New York 10598
(914) 789-7504