Tuning of database audits to improve scheduled maintenance in communication systems

Stefano Porcarelli[1], Felicita Di Giandomenico[2], Amine Chohra[2*], Andrea Bondavalli[3]

[1] Univ. of Pisa, Computer Engineering Dep., Via Diotisalvi 2,I-56126, Pisa, Italy
stefano.porcarelli@guest.cnuce.cnr.it
[2] IEI/CNR, Via Moruzzi 1, I-56100, Pisa, Italy
{digiandomenico, chohra}@iei.pi.cnr.it
[3] Univ. of Firenze, Dip. Sistemi e Informatica, V. Lombroso 6/17, I-50134, Firenze, Italy
andrea.bondavalli@cnuce.cnr.it

Abstract. To ensure the consistency of database subsystems involved in communication systems (e.g., telephone systems), appropriate scheduled maintenance policies are necessary. Audit operations, consisting in periodic checks and recovery actions, are typically employed in databases to cope with run time faults which may affect the dependability and quality of service of the overall system. This paper aims at investigating on appropriate tuning of audit operations, so as to find optimal balances between contrasting requirements, namely satisfactory database availability and low overhead due to audits. For this purpose, a methodology to analyse the behaviour of the database under scheduled maintenance is here suggested. Analytical models, essentially based on Deterministic and Stochastic Petri Nets (DSPN), are defined and analysed, in terms of dependability indicators. A sensitivity analysis wrt to the most affecting internal and external parameters is also performed on a case study.

1 Introduction

The problem of protecting data used by applications during their execution, against run-time corruption, has long been recognised to be a critical aspect highly impacting on the reliability/availability of systems relying on such internal database. Communication systems, such as telephone systems, are today-typical systems suffering from this problem, especially when a wireless environment is involved, which makes the data more prone to corruption. Indeed, these systems need to keep trace of resource usage status and of users data for correctly setting up and managing user calls. For this purpose, a database is included, where data are organised in such a way to capture the relationships existing among them. Data corruption may result in the delivery of a wrong service or in the unavailability of the service, with (possibly heavy) consequences on the quality of service perceived by users. Effective mechanisms to detect and recover from data corruption are then necessary; typically, audit operations are used, to perform periodic maintenance actions. Audits check and make the appropri-

* On leave at IEI-CNR, supported by ERCIM (European Research Consortium for Informatics and Mathematics) under post-doctoral training program (Contract Nr: 99-04).

ate corrections according to the database status and the detection/correction capability of the audit itself. How to tune the frequency of such checks in order to optimise system performance becomes another important aspect of the problem. This paper aims to give a contribution exactly on this last point.

In order to provide an analysis and evaluation support to help the on-line monitoring of data structures, the goal of our work consists in the definition of a methodology to model and evaluate the relevant dependability attributes of scheduled audit strategies. Contrasting, but correlated, issues have to be coped with; namely: high reliability/availability calls for frequent, deep-checking audits, while good performance in terms of accomplished services suffers from the execution power devoted to audits. We follow an analytical approach, essentially based on Deterministic and Stochastic Petri Nets (DSPN) [1, 7]. Analytical models, which capture the behaviour of the database in presence of scheduled maintenance, are defined and evaluated, in terms of identified dependability and performance measures. A sensitivity analysis with respect to the most affecting internal and external parameters is also performed on a case study, which helps in devising appropriate settings for the order and frequencies of audits to optimise selected performance indicators.

The rest of the paper is organised as follows. Section 2 presents the main characteristics of the target system and of the available audit policies. Section 3 introduces our approach to audit tuning. Section 4 discusses the identified figures of merit the assumptions made, and the basic sub-model elements used to analyse the behaviour of the database and of the audits. In section 5, a case study is set up and quantitatively evaluated to illustrate the utility of our approach; then conclusions are in Section 6.

2 System Context

We target telephone communication systems, which include a database subsystem, storing system-related as well as clients-related information, and providing basic services to the application process, such as read, write and search operations. Data concerning the status, the access rights and features available to the users, and routing information for dispatch calls are all examples of data contained in the database. The database is subject to corruption determined by a variety of hardware and/or software faults, such as internal bugs and transient hardware faults. The occurrences of such faults have the potential of yielding to service unavailability. Because of the central role played by such database in assuring a correct service to clients, means to pursue the integrity/correctness of data have to be carried out.

With the term data audit it is commonly indicated a broad range of techniques to detect errors and recover from them. The kind of checks performed on the data to test its correctness highly depends on the specific application at hand, on the system components, and environmental conditions which determine the expected fault model. Both commercial off-the-shelf and proprietary database systems are generally equipped with utilities to perform data audits, such as in [3, 4, 8]. For the purpose of our study, we assume that a set of audit procedures to cope with data corruption are provided, each characterised by a cost (in execution time) and coverage (as a measure of its ability to detect and/or correct wrong data). From the point of view of coverage, we distinguish between *partial audits*, characterised by a coverage lower than 1, and *complete audit*, which performs complete checks and recovery such that, after its

execution, the system can be considered as good as a new one. The considered audits are activated at pre-determined time intervals, in accordance with a maintenance strategy performed by an *audit manager*. In fact, an *audit manager* selects the part of the database to check/recover, the detection/recovery scheme to apply, and the frequency with which each check/recovery operation has to be performed. The audit manager is therefore responsible for applying the maintenance strategy to cope with database corruption and therefore preventing system unavailability. To set up an appropriate maintenance strategy, the audit manager would need some support, which helps it in evaluating the efficacy of applying different combinations of the available audit operations. In this work, we focus on such evaluation component (*strategy evaluator*), by developing a methodology to proper tuning of audit operations. In Fig. 1, the logical structure of the database subsystem and of the involved components is shown.

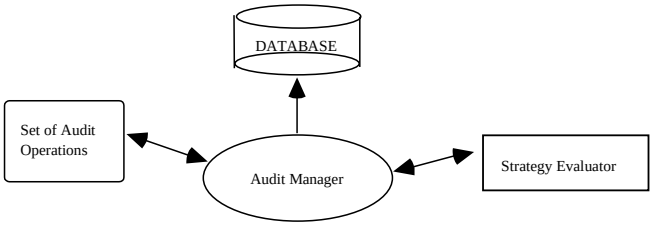

Fig. 1. Logical overview of the database subsystem

Records of the database tables also include fields that are used to reference records belonging to other tables. Such reference fields (pointers) have a dynamic content. Whenever a call is set up, a set of linked records is inserted in the database; these records store all the data relevant for the establishment and management of the ongoing calls. Records allocated to store the information on a specific call are released when a call ends. The specific set of relations that identify the linked structure of the database defines the *dependency scheme*.

A pointer may fail in two ways: *out of range*, i.e., its value incorrectly assumes the value of a memory location outside the database tables, or *in range*, when it wrongly points to a location memory inside the tables space. The latter kind of fault shows more dangerous in the general case, since a record belonging to another dependency is erroneously deallocated; we therefore say that an *in range fault* generates a *catastrophic failure*, while an *out of range* fault results into to a *benign failure*. In addition, although the single *out of range* fault is not catastrophic, its repeated occurrence (above a pre-fixed threshold) leads to a catastrophic failure. After a catastrophic failure, the system stops working.

In this work, we concentrate on maintenance policies for enhancing pointer correctness, which is undoubtless very critical for the application correctness; however, our approach is general methodology which can be easily adapted to take into account different specific database information.

3 A Methodology to Fine-Tuning of Audit Operations

Our goal is to identify a methodology to model and evaluate the relevant dependability attributes of scheduled audit strategies in order to derive optimal maintenance solutions. The main aspects of such a methodology are:

1. the representation of basic elements of the system and the ways to achieve composition of them;
2. the behaviour of the system components under fault conditions and under audit operations to restore a correct state;
3. the representation of failure conditions for the entire system;
4. the interleaving of audits with on-going applications and their relationships;
5. the effects of (combinations of) basic audit operations on relevant indicators for the system performance, in accordance with application requirements.

Our approach is based on Deterministic and Stochastic Petri Nets (DSPN). Specifically, in accordance with the points listed above, we defined general models which capture the behaviour of the database and of the maintenance policy checking it, to be easily adapted to specific implementations of databases and audit actions. The defined models allow investigating on the most relevant aspects in such system, related to both the integrity of the database and the overhead caused by the audit activities.

For the analysis purpose, the basic elements of the database are the pointer fields of the tables. In order to compact the basic information, one can represent in the same model structure the pointers belonging to database tables which: i) have the same failure rate; and ii) share the same audit operations, applied at the same frequency. We call the tables whose pointer fields share such characteristics as *homogeneous set*. Such compactness process has to be carefully performed in accordance with the set of maintenance policies to be analysed.

To represent the process of generation of pointers and of their next deletion at the end of the user call, one needs to model also the applications working on the database. This way, the events of system failure caused by erroneous pointers in dependencies at the moment of the end of a call are also captured.

Finally, the complete maintenance strategy has to be modelled, in the form of alternation of pure operational phases with others where applications and audits run concurrently.

The presentation of such general models, as well as the interactions among them, follows in the next section.

4 Modeling of Maintenance Policies

Before presenting the models, the relevant figures of merit defined for the analysis purpose and the assumptions made in our study are described.

In performing the system analysis and evaluation, we consider that the system works through missions of predefined duration [1]. To our purpose, two measures have been identified as the most sensible indicators, and the developed models have been tailored to them.

1. The reliability that should be placed on the database correctness, expressing the continuity of service delivered with respect to system specifications [5]. Actually,

to better appreciate the effect of maintenance, we will evaluate the unreliability, as a measure of the probability of not surviving a mission of a pre-fixed duration.

2. A performability measure [6], which shows appropriate to evaluate whether a certain maintenance strategy is "better" than another. Necessary to performability is the definition of a reward model; we use here, by way of example, a simple additive reward model that fits our mission-oriented systems. We assume that a gain G_1 is accumulated for each unit of time the system spends while performing operational phases, and a value G_2 is earned for each unit of time while audit operations are in execution, with $G_1 > G_2$. Finally, a penalty P is paid in the case of failure, again for each time unit from the failure occurrence to the end of the mission.

The models and analysis have been developed under the following assumptions:

1. pointers corrupt with an exponential rate λ_c. Pointer faults occur independently from each other, so the corruption rate for a dependence is the sum of the corruption rates of each pointer involved in that dependence;
2. audit operations and applications share the same processor(s); when audits are in execution, a reduced number of user calls can be satisfied. The entity of such reduction, being related to audit operations, may vary significantly;
3. audit operations are characterised by a coverage *c*, indicating the audit's probability of successful detection/correction. Intuitively, the higher is *c*, the more complex (and time consuming) is the corresponding audit;
4. according to the kinds of pointer failure (i.e., *in range* or *out of range*), *catastrophic* or *benign* failures are possible, as already discussed in Section 2;
5. each active user call involves an element (record) in each database table.

4.1 The Models

Exploiting the multiple-phased structure of our problem, we developed separate models to represent a) the behaviour of the system through the alternation of operational and audit phases, and b) the failure/recovery process of the system components.

Fig. 2 shows the model of a generic maintenance strategy. It represents the alternation of a (variable) number of operational phases (*Op1, …, Opn*) and audit phases (*Ma1, …, Man*), determining a *maintenance cycle*, which is then cyclically re-executed. Only one token circulates in the net. The deterministic transitions *TOp1, …, TOpn* model the duration of operational phases, while the deterministic transitions *TMa1, …, TMan* model the duration of the corresponding audit phase. The places *S1,…, Sn* and the instantaneous transitions *TS1, …,TSn* allow to complete the recovery action in the *homogeneous set*s (described later) before a new phase starts.

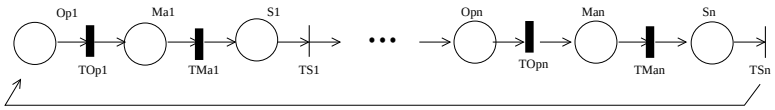

Fig. 2. Model of the maintenance strategy

The main elements of the application sub-net, shown in Fig. 3 (a), are:

- The place *Call_active* contains the number of the on-going calls.
- The place *Corrupted* contains the number of *out of range* corruptions of a dependence (*benign failures*); one token in the place *Failed* represents the *catastrophic failure* of the system.
- The instantaneous transition *T_active* allows updating the number of tokens in the *homogeneous set*: whenever a call is set-up, represented by token moving from *Call* to *Call_active,* a token is added in the place *Table* of each *homogenous set.*
- The exponential transition *T_idle* represents the duration of a call. When the system is in an operational phase, that transition fires with rate μ; during an audit phase the rate is $x*\mu$, where $0 \leq x \leq 1$ accounts for the percentage of the power processing lost during an audit phase with respect to an operational one.
- The instantaneous transitions *I_to_S*, *I_to_C*, and *I_to_F* model the behaviour of the database when a call ends. The choice of which of them fires depends on the marking of the places *actived* and *failed1* (out of range) or *failed2* (in range) in the representation of a homogeneous set sub-net (see Fig. 3(b)).

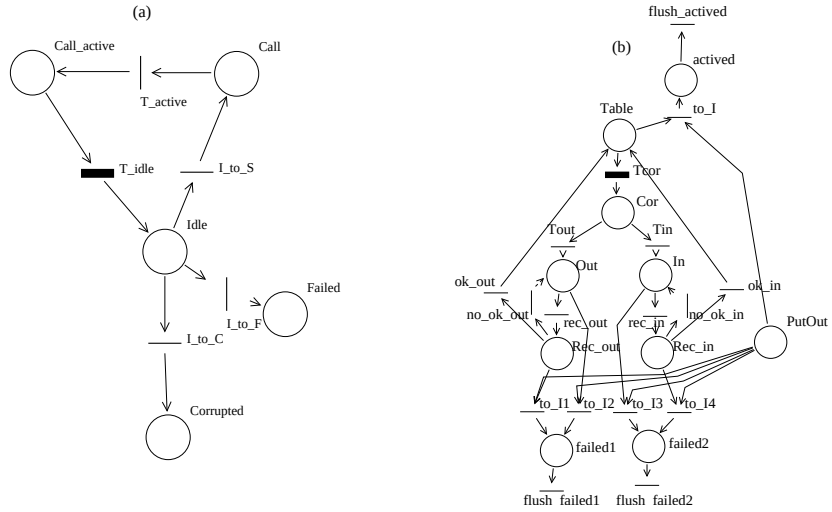

Fig. 3. The application model (a) and the model of a homogeneous set (b)

Fig. 3 (b) shows the model of a *homogeneous set*, i.e., of the pointers belonging to database tables having the same failure rate and subject to the same audits, with the same frequency. The sub-nets of the application and of the homogeneous set have to be connected together, since pointers are created and deleted by user calls. The meaning of the main elements in Fig. 3 (b) is:

- The firing of the exponential transitions *Tcor* models a pointer corruption. The instantaneous transitions *Tout* and *Tin* move a token in the places *Out* and *In* respectively to distinguish if a given pointer is corrupted *out of range* or *in range.*
- During a maintenance phase transitions *rec_out* and *rec_in* are enabled according to the audit specifications.
- The instantaneous transitions *no_ok_out, ok_out, no_ok_in, ok_in* model the recovery actions performed at the end of audit phases. They are enabled when there is a

token in the places *Sn* of the maintenance submodel. The success or failure of a recovery is determined by the coverage *c* of the applied audit.

- When a call ends, a token (a pointer) will leave the homogeneous set sub-net. In a probabilistic way and on the basis of the marking of the places *failed1*, *failed2*, and *actived* the decision is made on whether the dependence associated with a call is corrupted (out of range or in range) or not. The instantaneous transitions *I_to_S*, *I_to_C*, and *I_to_F* of the application sub-net (see Fig. 3 (a)) operate such choice.
- The instantaneous transitions *to_I, to_I1, to_I2, to_I3,* and *to_I4* are enabled when transition *T_Idle* of the application submodel fires and a token is moved in the place *PutOut.*
- The instantaneous transitions *flush_actived*, *flush_failed1*, and *flush_failed2* fire when there are no tokens in the place *Idle* and after the instantaneous transitions *I_to_S*, *I_to_C* and *I_to_F* of the application sub-net.

From the DSPN models, the measures we are interested in are derived as follows:

- The *Unreliability* is the probability of having one token in the place *Failed* (in the application model) or a given number of tokens in the place *Corrupted.*
- The *Performability* is evaluated with the following formula:
  G_1* {Operational time while the system works properly} + G_2*{Audit time while the system works properly} – P*{Time while the system is failed}.

5 A Case Study

To illustrate the usefulness of our approach and to give the reader an idea of the relevance of our analysis, a case study is set-up and evaluated.

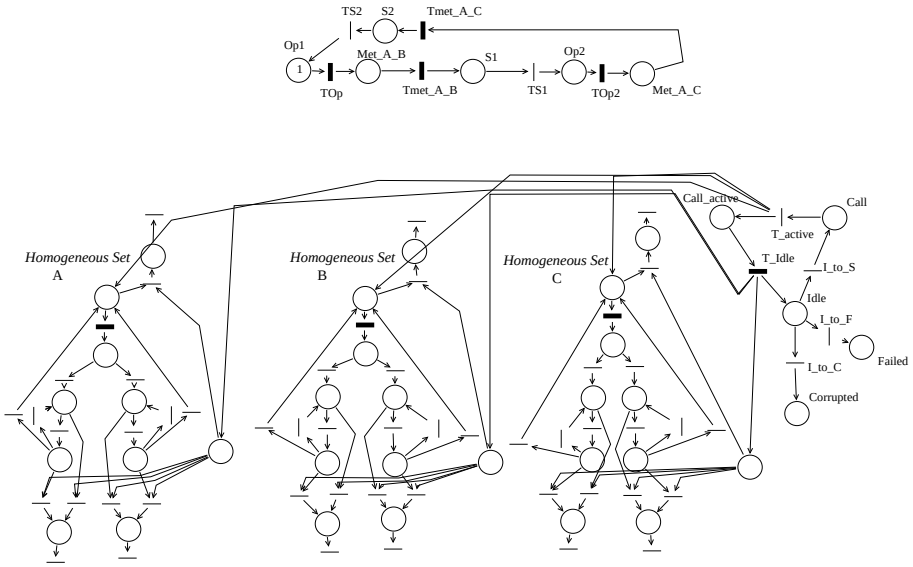

Fig. 4. Model of the case study

We consider a database supporting a hypothetical telephone system, to which both partial and total audits are periodically applied. The defined maintenance strategy consists in alternating partial checks on different sets of dynamic data (pointers) with operational phases for a certain number of times, after which a complete audit is executed which resets the database status to the initial conditions. We are interested in evaluating the unreliability and performability between two complete audits; it is then straightforward to make forecasts on the system for any chosen interval of time.

By applying our methodology and composing the model elements defined in the previous section, the model instance for our case study is derived, as sketched in Fig. 4. The upper part of the model represents the maintenance strategy, which encompasses two operational phases interleaved with two executions of the some partial audit on two non-disjoint sets of data. Therefore, three homogeneous sets (A, B and C) are defined in the lower part of the model. The relationships with the application model are shown in the right side of the Fig. 4.

5.1 Numerical Evaluation

The derived models are solved by the DEEM tool [2], which provides an analytical transient solver. DEEM (DEpendability Evaluation of Multiple phased systems) is a tool for dependability modelling and evaluation, specifically tailored for multiple phased systems and therefore very suitable to be used in our context.

The variable parameters in our numerical evaluations are: i) the pointer corruption rate λ_C, which varies from $5*10^{-7}$ to $5*10^{-8}$ per seconds; ii) the duration of an operational phase, Top, which ranges from 60 to 300 seconds; iii) the coverage factor of partial audits, from 0.8 to 0.999; iv) the parameter P (penalty) of the reward structure. The other involved parameters have been kept fixed; among them: the time interval between two complete audits has been set to 2 hours; the maximum number of user calls concurrently active is 100; the call termination rate is $3.33*10^{-3}$ per seconds; the number of benign failures necessary to determine a catastrophic system failure is 5; the parameters G_1 and G_2 of the reward structure.

Fig. 5(a) shows the performability as a function of the duration of the operational phase, for different values of the penalty associated to the failure condition of the system. For the chosen setting, it can be observed a noticeable influence of such penalty factor P on the resulting performability. When P is just a few times the value of G_1 (i.e. the gain in case the system is fully and correctly operating), increasing Top brings benefits to the performability. This means that in such a case, the main contribution to the performability is given by the reward accumulated over operational phases. However, for P from 200 to 300, an initial improvement can be observed, which is then lost, although the performability degradation is not dramatic. When P is two order of magnitudes higher than G_1, the cost of a failure is so big that lengthening Top (which implies a higher risk of failure) results in a performability detriment.

Fig. 5(b) shows the performability keeping fixed the reward structure and at varying values of the coverage of the audit procedure and the length of the operational phase. Two effects can be immediately noticed. First, as expected, the performability improves with growing values of the coverage. Second, it can be observed a "bell shape" of the curves: the performability grows at growing values of the duration of the operational phase till a maximum value of Top after which the trend inverts.

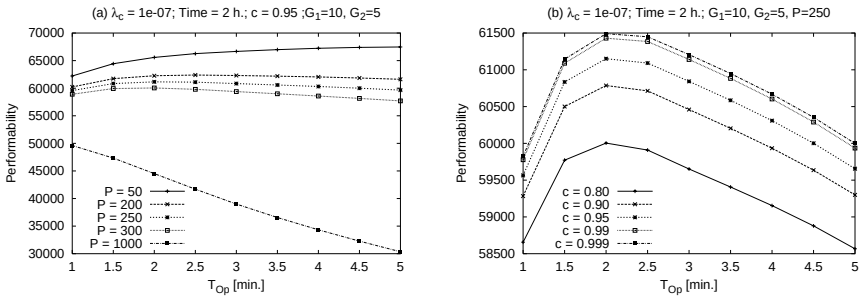

Fig. 5. Performability at varying of T_{op} and Penalty (a) and Coverage (b) respectively

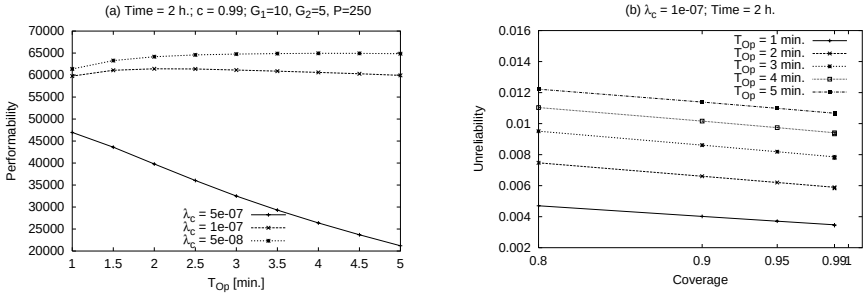

Fig. 6. Performability at varying T_{op} and λ_C (a) and Unreliability (b) at varying c and T_{op}

In fact, the higher reward obtained during a longer operational phase is at first the dominant factor in determining the performability, but lengthening Top also means exposing the system to a higher risk of failure, and the penalty P to be paid in such a case becomes the most influencing parameter in the second part of the Fig. 5(b).

Fig. 6(a) completes the analysis of the performability, at varying values of Top and for three different values of the pointer failure rate. The impact of λ_C on the performability is noteworthy, and behaviour similar to that in Fig. 5(a) is observed. Fig. 6(b) shows the behaviour of the unreliability at varying values of the coverage and for several values of Top. Of course, the unreliability improves at increasing both the audits frequency (i.e., small Top) and the coverage of the audits. It can be noted that same values of the unreliability can be obtained by adopting audits with a higher coverage or applying more frequently audits with a lower coverage.

The analyses just discussed give a useful indication about the tuning of the major parameters involved in the database system. The optimal trade-off between the frequency of the audits and the investment to improve the coverage of the audits can be found, to match the best performability and dependability constraints.

6 Conclusions

This paper has focused on maintenance of dynamic database data in a communication system. To achieve a good trade-off in terms of overhead and efficacy of the maintenance, it is necessary to properly choose which audit operations are to be applied and how frequently they should be used.

We proposed a modular methodology to model and evaluate the relevant dependability attributes of scheduled audit strategies. Our approach is based on Deterministic and Stochastic Petri Nets (DSPN) and on the DEEM tool. Despite our proposed approach needs further work for being assessed, nevertheless we have identified several relevant characteristics specific to this class of systems.

The major impact of this study is the definition of a general model for the evaluation of the effectiveness of the audit strategies. Paramount criteria for our work have been the extensibility and flexibility in composing the audit strategies. Of course, in order for our models to be really useful for the selection of proper order and frequencies of audit operations, input parameters such as cost and coverage of the checks and failure data are necessary and should be provided. Investigations to assess the merits of our approach towards the incremental structure of audit methods are planned as the next step. Also, extensions of the case study to include the comparison of the effectiveness/benefits derived from applying different combinations of audits (i.e., different maintenance strategies) constitute interesting evolution to this work.

References

1. A. Bondavalli, I. Mura, and K. S. Trivedi. Dependability Modelling and Sensitivity Analysis of Scheduled Maintenance Systems. In Proc. EDCC-3, September 1999, pp. 7-23.
2. A. Bondavalli, I. Mura, S. Chiaradonna, R. Filippini, S. Poli, F. Sandrini. DEEM: a Tool for the Dependability Modeling and Evaluation of Multiple Phased Systems. In Proc. Int. Conf. DSN2000, New York, June 26-28.
3. D. Costa, T. Rilho, H. Madeira. Joint Evaluation of Performance and Robusteness of a COTS DBMS through Fault-Injection. In Proc. Int. Conf. DSN 2000, pp. 256-260.
4. G. Haugk, F. M. Lax, R. D. Royer, J. R. Williams. The 5ESS Switching System. Maintenance Capabilities. AT&T Technical Journal, vol. 64, n.6, 1985, pp. 1385-1416.
5. J. C. Laprie. Dependability - Its Attributes, Impairments and Means. In Predictably Dependable Computing Systems, J.C. Laprie, B. Randell, H. Kopetz, B. Littlewood (Eds.), Springer Verlag, 1995, pp. 3-24.
6. J. F. Meyer. On Evaluating the Performability of Degradable Computing Systems. IEEE Transactions on Computers, vol. C-29, pp. 720-731, August 1980.
7. I. Mura, A. Bondavalli, X. Zhang and K. S. Trivedi. Dependability Modeling and Evaluation of Phased Mission Systems: a DSPN Approach. In Proc. of 7th IFIP Int. Conf. DCCA, San Jose, USA, IEEE Computer Society Press, 1999, pp. 299-318.
8. Oracle8 Server Utilities release 8.0. Chapter 10, http://pundit.bus.utexas.edu/oradocs/server803/A54652_01/toc.htm