ELSEVIER

Performance Evaluation 46 (2001) 45–62

PERFORMANCE EVALUATION
An International Journal

www.elsevier.com/locate/peva

Fine grained software degradation models for optimal rejuvenation policies

Andrea Bobbio [a], Matteo Sereno [b,*], Cosimo Anglano [a]

[a] *Dipartimento di Scienze e Tecnologie Avanzate (DISTA), Università del Piemonte Orientale, Corso Borsalino 54, 15100 Alessandria, Italy*

[b] *Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10149 Torino, Italy*

Received 19 May 1999; received in revised form 6 March 2001

Abstract

In this paper, we address the problem of preventive maintenance of operational software systems, an approach recently proposed to counteract the phenomenon of software "aging". We consider the so-called "software rejuvenation" technique [Software rejuvenation: analysis, module and applications, in: Proceedings of the 25th International Symposium on Fault-Tolerance Computing (FTCS-25), Pasadena, CA, USA, June 1995], which consists in periodically stopping the software system and then restarting it in a "robust" state after a proper maintenance, and we propose a methodology for the quantitative analysis of rejuvenation strategies. The methodology is based on a *fine grained* model that assumes that it is possible to identify the current degradation level of the system by monitoring an observable quantity, so that the rejuvenation strategy can be tuned on the measured degradation. Based on this methodology, we present two different strategies that allow to decide whether and when to rejuvenate, and we exploit the theory of renewal processes with reward to estimate the steady-state unavailability of the software system, which is used to define an optimality criterion that allows us to evaluate the proper rejuvenation intervals. The methodology and the rejuvenation strategies are demonstrated by applying them to a real-world case study, arising in the area of database maintenance for data archiving, and to a hypothetical setting used to assess the sensitivity of the technique to various degradation processes.

Keywords: Preventive maintenance; Software aging; Software rejuvenation; Fine grained software degradation models; Quantitative analysis of rejuvenation strategies

1. Introduction

In recent years, considerable attention has been devoted to continuously running software systems whose performance characteristics are smoothly degrading in time until an intolerable reduction of capacity is eventually reached [3,11]. Smooth degradation can be related to a variety of causes, such as progressive filling up of finite storage media, memory leaking, unreleased file-locks, accumulation of round-off errors, etc. Many interesting examples and a bibliographic review can be found in [9].

* Corresponding author. Fax: +39-011-751603.

0166-5316/01/$ – see front matter
PII: S0166-5316(01)00037-2

Usual strategies to deal with systems that manifest smooth degradation are reactive in nature, i.e., they consist of actions taken after a failure [11]. Recently, preventive strategies have been proposed that consist in occasionally or periodically recovering the system and its operational environment to a clean state. These policies have been referred to as *rejuvenation*, where rejuvenation can be defined as a preventive maintenance action aimed at cleaning up the system to prevent more severe failures in the future. Therefore, rejuvenation has the same motivation and advantages/disadvantages as preventive maintenance policies in hardware systems [9].

Any rejuvenation typically involves an overhead, but it prevents more severe failures from occurring. Hence, an important issue in analysing rejuvenation policies is to determine their usefulness in terms of availability and cost and to provide an optimal criterion to decide when and how often to recover the system from the degraded state. Quantitative models have been, up to now, based on a "black box" approach. Huang et al. [11] consider the degradation as a two step process. From the clean state the system jumps into a "degraded" state from which two actions are possible: rejuvenation (with return to the clean state) or transition to the complete failure state. The authors model the three-state process as a continuous time Markov chain (CTMC) and they derive and discuss only the steady-state behaviour. In this way, the authors need to provide only the average values of the different transition times.

Garg et al. [7,8] introduce into the model the idea of a periodic and fixed rejuvenation interval so that the stochastic behaviour of the systems is no longer a CTMC. The authors resort to the use of Markov regenerative stochastic Petri nets. In subsequent works [9,10], the authors consider the effect of the workload by including the arrival and queuing of jobs in the system, and the time dependency of the load and of the service time on the age of the system. With this addition, the arrival and service rates are time-varying and the overall model becomes a three-state non-homogeneous Markov process.

All the mentioned papers are based on the idea that the degradation process can be captured by a three state model, so that it is possible to isolate a clean state, a degraded state and a crash state. As a consequence, to completely specify the model it is necessary to identify the distribution of the random times that characterise the transitions among the three states. This "coarse-grained" or "black box" type model has been criticised in [3,10], where the need for monitoring the system performance level is discussed and emphasised. If the system is monitored, the degradation process can be followed in time, so that the decision whether to rejuvenate or run the system with no corrective action can be taken based on the measured level of performance.

Following the above idea, this paper proposes a "fine grained" model to the smooth degradation process of a software system. The fine grained model is based on the assumption that it is possible to identify the current performance level of the system versus time by measuring an observable quantity, so that the future strategy can be tuned on the measured parameter. The observable parameter can be related to the filling level of a finite storage medium, the status of the hardware (like the number of available processors or communication channels [3]), the condition of the memory or of a buffer (like the number of free memory blocks or the loss probability in a finite buffer), or a load-dependent and time-dependent characteristic (like the response time). Due to the nature of the software degradation process, it is legitimate to assume that the damage accumulates in discrete amounts at isolated points in time. Hence, the degradation process is modelled as a sequence of successive faults or shocks [6] that increase the value of the observable parameter by a random quantity whose cumulative distribution function can be empirically estimated. The system is considered out of service as soon as the measured parameter reaches an assigned threshold level. In the following, we identify the crossing of the tolerable threshold by the observable parameter as the system *crash*.

Two criteria are studied to decide whether and when to rejuvenate: a risk based criterion in which the decision is taken so that the risk of incurring into a crash before rejuvenation is less than a preassigned level, and an alert threshold criterion, in which an alert threshold (below the crash threshold) is established and the system is rejuvenated as soon as the alert barrier is crossed. In order to formulate the evaluation of the rejuvenation interval as an optimal design problem, the costs (or down times) associated to a rejuvenation action or to a crash are included into the model. The long-term behaviour of the system is then represented as a renewal process with rewards, whose reward rates can be specified according to the assumed rejuvenation criterion. A complete analytical derivation is developed, assuming that the degradation is monotonically increasing. With both criteria an optimal rejuvenation interval is determined based on the minimisation of the system unavailability.

The paper is organised as follows. In Section 2, the degradation model is introduced and two variants are discussed: when the system is monitored at regular intervals, and when the statistics of the interarrival times between the faults can be inferred from the physical observation of the actual degradation mechanism. The investigated rejuvenation policies are dealt with in Section 3. The long-term behaviour of the system is modelled as a renewal process with rewards in Section 4; the computation of the accumulated reward can be used to evaluate the unavailability under the two considered rejuvenation policies. Section 5 presents two examples: the first one is a case study arising in the context of database maintenance, while in the second one a set of numerical experiments are performed to investigate the sensitivity of the model to various features of the underlying degradation process. Finally, Section 6 concludes the paper.

2. Software degradation models

We assume that the degradation condition of a software system can be determined by monitoring an appropriate parameter. In the following, the observed parameter is referred to as the *degradation index*. Let $s(t)$ be the value of the degradation index measured at time t. Hence, $s(t)$ $(t \geq 0)$ is the stochastic process modelling the system degradation versus time.

The degradation process, as observed by means of the degradation index, consists of a sequence of additive random shocks. Each shock increases the degradation index by a random variable X $(X \geq 0)$ of cdf $F_X(x)$. $F_X(x)$ must be inferred from the physical degradation process, or estimated through the collection of experimental measures.

We assume that the observed system is alive when $s(t)$ ranges in the interval $[s_{\min}, s_{\max})$, where $s_{\min}$ is the value of $s(t)$ measured when the system is fully available and $s_{\max}$ is the threshold representing the maximum tolerated value of the degradation index. Let T be the time at which $s(t)$ reaches the value $s_{\max}$ for the first time:

$$T = \min\{t : s(t) \geq s_{\max}\}, \tag{1}$$

hence, T is the time for the system to reach the crash state. T is a random variable with cdf $D_T(t, s_{\max}) = \Pr\{T \leq t\}$ and survival function $\bar{D}_T(t, s_{\max}) = 1 - D_T(t, s_{\max})$. In the following, we use the notations $D_T(t)$ and $\bar{D}_T(t)$ as shorthand notation for $D_T(t, s_{\max})$ and $\bar{D}_T(t, s_{\max})$. We will use the extended notation only if the considered threshold is different from $s_{\max}$.

Two different hypotheses are considered for the accumulation of the degradation, depending whether the system is monitored at equispaced intervals, or the statistics of the shocks interarrival times can be known from the physical degradation mechanism.

2.1. Equispaced observations

The degradation index $s(t)$ is monitored at equispaced intervals Δt, so that the sequence of values $s(k\,\Delta t)$ $(k = 0, 1, 2, \ldots)$ is known. Let X_k be the increment in the degradation index measured at the kth observation. We introduce the random variable h_k representing the cumulative degradation up to the kth interval, hence

$$s(k\,\Delta t) = h_k = s_{\min} + \sum_{i=1}^{k} X_i. \tag{2}$$

Since the degradation process is assumed to be monotonically increasing, because the increment of the degradation index is constrained to be a positive random variable, the survival function can be expressed as follows:

$$\bar{D}_T(k\,\Delta t) = \Pr\{h_k < s_{\max}\}. \tag{3}$$

If the increments are independent of the same distribution, then

$$\bar{D}_T(k\,\Delta t) = \int_0^{s_{\max}-s_{\min}} f_X^{*k}(z)\,\mathrm{d}z, \tag{4}$$

where $f_X^{*k}(z)$ is the k-fold convolution integral of the density $f_X(x)$ of the cdf $F_X(x)$ (see [15] for details).

2.2. Degradation through random shocks

The degradation of the system is assumed to be caused by faults or shocks occurring randomly in time according to a point process $S(t)$. $S(t)$ represents the number of shocks experienced by the system in the interval $[0, t]$, and must be inferred from the statistics of the actual degradation mechanism.

The increment of the degradation index at each shock is a random variable X of cdf $F_X(x)$. The degradation process is completely specified if the probability $P_k(t)$ of having k shocks at time t is known

$$P_k(t) = \Pr\{S(t) = k\}. \tag{5}$$

In this case, the survival function becomes

$$\bar{D}_T(t) = \sum_{k=0}^{\infty} \bar{D}_T(t|k) \cdot \Pr\{S(t) = k\}, \tag{6}$$

where $\bar{D}_T(t|k)$ is the survival probability at time t conditioned on the occurrence of k shocks in the time interval $[0, t]$. If we assume that the increment of the degradation index at each shock is a positive random variable X of cdf $F_X(x)$ (and density $f_X(x)$), and we further assume that the shocks are independent and the degradation process is additive, $\bar{D}_T(t|k)$ can be expressed as a k-fold convolution integral, by means of (4).

In degradation processes, it is customary to assume that the degradation rate is dependent on the age, and an age dependent degradation process can be modelled by assuming that the interarrival times between

random shocks are time-dependent, so that the point process $S(t)$ is a non-homogeneous Poisson process (NHPP). We assume a power-law dependence for the time-dependent rate $\lambda(t)$ of the NHPP

$$\lambda(t) = c\beta t^{\beta-1}, \tag{7}$$

where $\beta > 1$ means that the interarrival time between shocks is decreasing with the age of the system (shocks occur more frequently as the system deteriorates), and the opposite behaviour holds for $\beta < 1$. The special case $\beta = 1$ is the Poisson process.

Adopting the time-dependent rate defined by Eq. (7), the probability $P_k(t)$ and the survival probability $\bar{D}_T(t)$ can expressed as, respectively

$$P_k(t) = \mathrm{e}^{-ct^\beta}\frac{(ct^\beta)^k}{k!}, \qquad \bar{D}_T(t) = \sum_{k=0}^{\infty}\bar{D}_T(t|k)\cdot \mathrm{e}^{-ct^\beta}\frac{(ct^\beta)^k}{k!}. \tag{8}$$

The random shock model described in this section has been extensively studied in [6] under far more general assumptions than the additivity of identical independent shocks assumed in Eq. (4). A very general and interesting result derived from [6], in the case of Poisson distributed random shocks, is that under only the hypothesis that X is positive, the survival probability $\bar{D}_T(t)$ is increasing hazard rate in average (IHRA). Hence, independently on the nature of the cdf of a single shock, the overall process represents a true degradation on the average [4].

3. Rejuvenation policies

Rejuvenation is defined as a preventive maintenance action that prevents the system from reaching the crash condition. Hence, based on the fine grained degradation model introduced in the previous section, a rejuvenation policy is a rule (or a set of rules) to clean up the system before the threshold level $s_{\max}$ is reached. According to the defined model, cleaning up the system means restoring the system to a fully operational condition, so that after rejuvenation the observable parameter is restored to a value $s(t) = s_{\min}$. Two different rejuvenation policies are introduced and investigated: a policy based on a preassigned risk level, and a policy based on an alert threshold.

3.1. Risk-level rejuvenation policy

Let T, as defined in (1), be the first passage time of the degradation index $s(t)$ across the threshold level $s_{\max}$, and let $D_T(t)$ be its cdf. Given a confidence level α (e.g., $\alpha = 0.95$) the decision of performing the rejuvenation can be taken with a risk $(1-\alpha)$ at a time θ defined as

$$D_T(\theta) = 1 - \alpha. \tag{9}$$

The value θ is the $(1-\alpha)$th percentile of the distribution $D_T(t)$, and hence the event of incurring a crossing of the tolerance threshold $s_{\max}$ before θ occurs with a risk equal to $(1-\alpha)$.

A possible realisation of a stochastic process modelling the system rejuvenation under the risk-level policy is shown in Fig. 1. As the degradation proceeds, the decision whether to rejuvenate is taken at a time corresponding to the $(1-\alpha)$th percentile of the first passage time cdf of the degradation index across the barrier level $s_{\max}$. Therefore, under this policy the rejuvenation interval is fixed and equal to θ, and the crash is expected to occur before θ with probability $(1-\alpha)$.

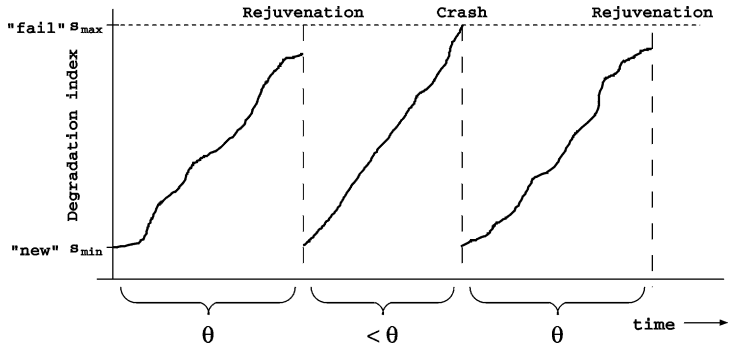

Fig. 1. A realisation of the stochastic process modelling the system rejuvenation.

3.2. Alert threshold rejuvenation policy

This policy is more user-oriented or control-oriented. A criterion is provided to decide whether to rejuvenate based on the current value of the measured degradation index so that the system can automatically perform the rejuvenation action when the proper conditions occur.

In this case, an alert threshold $s_{al} \leq s_{max}$ is established and the rejuvenation is performed as soon as $s(t)$ reaches the value s_{al} for the first time. A possible realisation of this rejuvenation policy is shown in Fig. 2.

In this second policy, the decision of performing a rejuvenation action is driven by the value reached by the degradation index. Under the equispaced monitoring, the rejuvenation is performed at the kth interval if $s((k-1)\Delta t) \leq s_{al}$ and $s(k\,\Delta t) > s_{al}$ (i.e., the system is below the alert threshold at the $(k-1)$th measure and jumps over the alert threshold at the kth measure). However, at the kth measure two events are still possible: if $s_{al} < s(k\,\Delta t) \leq s_{max}$ the system is below the failure threshold and a proper rejuvenation is performed, on the other hand if $s(k\,\Delta t) > s_{max}$ the system crashes before rejuvenation can take place. Of course the distance between s_{al} and s_{max} determines the probability that the system crashes before rejuvenation. Hence, a design problem is the determination of the level of the alert threshold s_{al}, or, expressed in relative terms, of the ratio $\sigma_{al} = s_{al}/s_{max}$.

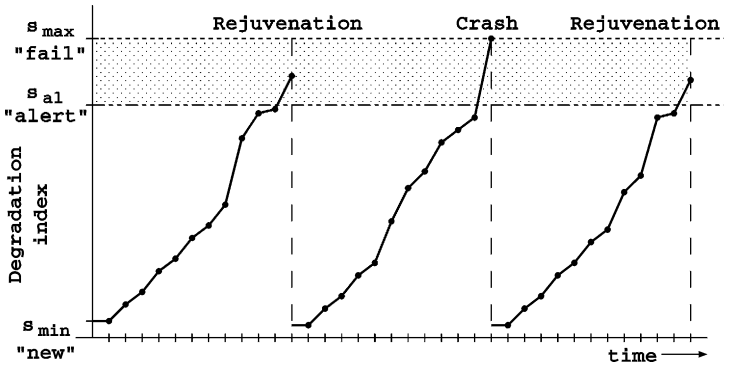

Fig. 2. A realisation of the stochastic process modelling the system rejuvenation based on an alert threshold.

Let T denote the first passage time of the degradation index $s(t)$ across a barrier of height s at the kth interval, and let $D_T(k\,\Delta t, s)$ denote its cdf and $d_T(k\,\Delta t, s)$ its density, i.e., $d_T(k\,\Delta t, s)\,\mathrm{d}s$ is the probability that the degradation index achieves a value $[s, s + \mathrm{d}s]$ at the kth interval.

The probability that the rejuvenation is performed at the kth interval, given that the alert threshold is s_{al}, can be evaluated according to the following expression:

$$\Pr\{\text{rej. at } k\text{th step}, s_{\mathrm{al}}\} = \int_{s_{\min}}^{s_{\mathrm{al}}} d_T((k-1)\,\Delta t, z)(F_{X_k}(s_{\max} - z) - F_{X_k}(s_{\mathrm{al}} - z))\,\mathrm{d}z, \tag{10}$$

and the probability that system crashes at the kth interval is

$$\Pr\{\text{crash at } k\text{th step}, s_{\mathrm{al}}\} = \int_{s_{\min}}^{s_{\mathrm{al}}} d_T((k-1)\,\Delta t, z)(1 - F_{X_k}(s_{\max} - z))\,\mathrm{d}z, \tag{11}$$

where $F_{X_k}(x)$ is the cdf of the kth increment.

If the increments of the degradation index are assumed to be independent identical random variables with cdf $F_X(x)$, we have that $F_{X_k}(x) = F_X(x)$ and the density $d_T(k\,\Delta t, s)$ in Eqs. (10) and (11) becomes $d_T(k\,\Delta t, s) = f_X^{*k}(s)$.

4. Long-term reward measures

The long-term behaviour of a system under rejuvenation can be depicted as in Fig. 1 or 2. The system starts in the "new" state and experiences gradual degradation until either a rejuvenation or a crash occurs, whichever comes first. In both figures, two rejuvenations and one crash have been represented. Subsequent to a rejuvenation or a crash, a restoration action is performed that brings the system in the "new" state again: the observable effect of the restoration is that the degradation index is reset to a value $s(t) = s_{\min}$. The long-term evolution of the system degradation in time can be seen as a *renewal process* [5], where the interrenewal distribution is the distribution of the minimum between rejuvenation and crash. Let $\{N(t), t \geq 0\}$ be the renewal process representing the degradation and successive restoration of the system. $N(t)$ represents the number of renewals up to time t.

In general, since rejuvenations are planned events, the overall downtime associated to a rejuvenation is less than the overall downtime associated to a crash [3,11]. Similarly, the cost associated to a scheduled downtime subsequent to a rejuvenation is less than the cost associated to an unscheduled downtime due to a crash. The proper rejuvenation interval can be estimated by an optimality criterion that balances these two cost factors.

The cost factors can be included in our model as rewards. At the end of each cycle a reward equal to the rejuvenation cost is accumulated if the cycle terminated with a rejuvenation, or a reward equal to the crash cost is accumulated if the cycle terminated with a crash. By using the results from the theory of *renewal processes with cost/rewards* [13], we can evaluate various measures related to the economical convenience of performing rejuvenation versus crash. The long-term reward measures are computed under the two rejuvenation policies considered in the previous section.

4.1. Reward measures for the risk policy

Given a confidence level α, the time θ at which rejuvenation is performed is given by Eq. (9). To point out the dependence of the rejuvenation interval on the risk level α, we use the notation θ_α.

Let Y_n denote the duration of the nth renewal cycle. If T_n is the crash time at cycle n, defined in Eq. (1), the nth interrenewal time is given by

$$Y_n = \min(T_n, \theta_\alpha).$$

At the end of the nth cycle a reward R_n is earned (or paid). We can assume that the rejuvenation cost is c_1 (independent of n). On the other hand, the cost of a crash, subsequent to a failure before rejuvenation is $c_1 + c_2$. Hence, c_2 is the extra-cost incurred by the system if a crash occurs before rejuvenation. At each renewal the reward is given by

$$R_n = \begin{cases} c_1 & \text{if } Y_n = \theta_\alpha, \\ c_1 + c_2 & \text{if } Y_n = T_n < \theta_\alpha. \end{cases}$$

Let $\mathcal{C}(t)$ be the total cost (total accumulated reward) incurred at time t. It follows that

$$\mathcal{C}(t) = \begin{cases} 0 & \text{if } N(t) = 0, \\ \sum_{n=1}^{N(t)} R_n & \text{if } N(t) > 0. \end{cases}$$

The process defined by $\{\mathcal{C}(t), t \geq 0\}$ is a renewal reward process [13]. By denoting with $r = E(R_n)$ the expected value of the reward R_n gained at the end of each cycle and by $\tau = E(Y_n)$ the expected value of the interarrival time Y_n, we can use the following result from the theory of renewal processes with cost/rewards (provided $r < \infty$ and $\tau < \infty$):

$$\lim_{t\to\infty} \frac{\mathcal{C}(t)}{t} = \frac{r}{\tau} \quad \text{with prob. 1.} \tag{12}$$

In particular, we have that

$$\tau = E(Y_n) = E(\min(T_n, \theta_\alpha)) = \int_0^{\theta_\alpha} (1 - D_T(t))\,\mathrm{d}t = \int_0^{\theta_\alpha} \bar{D}_T(t)\,\mathrm{d}t, \tag{13}$$

$$r = E(R_n) = c_1 + c_2 D_T(\theta_\alpha) = c_1 + c_2(1 - \alpha). \tag{14}$$

By substituting (13) and (14) into (12), the long-run cost rate is given by

$$\lim_{t\to\infty} \frac{\mathcal{C}(t)}{t} = \frac{c_1 + c_2(1-\alpha)}{\int_0^{\theta_\alpha} \bar{D}_T(t)\,\mathrm{d}t}. \tag{15}$$

According to the above results, the rejuvenation design problem can be formulated as follows. In order to design a system with a low risk of crash, α should be chosen close to 1. In this case, the rejuvenation interval gets smaller and we have more frequent rejuvenations. The rejuvenation cost increases and the crash cost decreases. The opposite behaviour is obtained as α decreases. In this case, the probability of incurring into a crash rises and so its related cost, but, on the other hand, the rejuvenation cost decreases.

Clearly, the variation of the risk α determines an opposite behaviour in the two cost factors that form the total accumulated reward. The rejuvenation cost is increasing with α, while the crash cost is decreasing.

Hence, it is legitimate to find an optimal value of α and, correspondingly of the rejuvenation interval θ_α, that minimises the long-run cost. The optimal value of θ_α depends on $D_T(\cdot)$ and α and on the reward factors c_1 and c_2.

4.2. Reward measures for the alert policy

Also in the case of the rejuvenation policy with the alert threshold, the evolution of the system in time can be seen as a renewal process where the renewals occur when the degradation index crosses the alert threshold (Fig. 2). The interrenewal time Y_n is given, in this case, by the first passage time of the degradation index $s(t)$ across the alert threshold s_{al}. Indeed, as the alert threshold is passed, the system is renewed independently of the fact that the restoration is due to a rejuvenation or a repair after a crash.

The long-term accumulated reward can be computed following the same pattern developed in the previous section. The expected value of the interarrival time Y_n is given by

$$\begin{aligned}\tau = E(Y_n) &= \sum_{k=1}^{\infty} k\,\Delta t \cdot \Pr\{\text{crossing } s_{\mathrm{al}} \text{ at } k\text{th step}\} \\ &= \sum_{k=1}^{\infty} k\,\Delta t \int_{z=0}^{s_{\mathrm{al}}} d_T((k-1)\,\Delta t, z)(1 - F_{X_k}(s_{\mathrm{al}} - z))\,\mathrm{d}z. \end{aligned} \tag{16}$$

Since

$$\Pr\{\text{crossing } s_{\mathrm{al}} \text{ at } k\text{th step}\} = \bar{D}_T((k-1)\,\Delta t, s_{\mathrm{al}}) - \bar{D}_T(k\,\Delta t, s_{\mathrm{al}}),$$

Eq. (16) becomes

$$\tau = \sum_{k=1}^{\infty} k\,\Delta t(\bar{D}_T((k-1)\,\Delta t, s_{\mathrm{al}}) - \bar{D}_T(k\,\Delta t, s_{\mathrm{al}})) = \Delta t \sum_{k=0}^{\infty} \bar{D}_T(k\,\Delta t, s_{\mathrm{al}}). \tag{17}$$

The average cost/reward earned (or paid) at each cycle can be computed by the following expression:

$$r = E(R_n) = c_1 \cdot \Pr\{\text{rej.}, s_{\mathrm{al}}\} + (c_1 + c_2) \cdot \Pr\{\text{crash}, s_{\mathrm{al}}\}, \tag{18}$$

where

$$\Pr\{\text{rej.}, s_{\mathrm{al}}\} = \sum_{k=1}^{\infty} \Pr\{\text{rej. at } k\text{th step}, s_{\mathrm{al}}\}, \tag{19}$$

$$\Pr\{\text{crash}, s_{\mathrm{al}}\} = \sum_{k=1}^{\infty} \Pr\{\text{crash at } k\text{th step}, s_{\mathrm{al}}\} \tag{20}$$

with $\Pr\{\text{rej.}, s_{\mathrm{al}}\} + \Pr\{\text{crash}, s_{\mathrm{al}}\} = 1$. The probabilities $\Pr\{\text{rej. at } k\text{th step}, s_{\mathrm{al}}\}$ and $\Pr\{\text{crash at } k\text{th step}, s_{\mathrm{al}}\}$ are those defined by Eqs. (10) and (11), respectively. The long-term cost rate, under the alert policy, can then be computed by substituting Eqs. (16) and (18) into (12).

The design parameter, in this case, is the relative value of the alert threshold σ_{al} expressed as the ratio $\sigma_{\mathrm{al}} = s_{\mathrm{al}}/s_{\max}$. If $\sigma_{\mathrm{al}} = 0$ the system is continuously rejuvenated if it does not crash before $k = 1$.

As σ_{al} increases the cost associated to the rejuvenation decreases (rejuvenation is less frequent) and the cost associated to the crash increases (crash becomes more likely). Hence, the two cost factors have a competitive behaviour as a function of σ_{al}, and an optimal value can be obtained once the numerical values of the cost parameters are specified.

4.3. System unavailability as a reward measure

If the downtimes, either due to rejuvenation or crash, are small compared with the expected interrenewal time τ, the cost function $\mathcal{C}(t)$ defined in (12) can be utilised to estimate the steady-state system unavailability.

Indeed, let c_1 be the downtime associated to a rejuvenation action and (c_1+c_2) the down time associated to a crash, the term $r = E(R_n)$ is the expected downtime associated with a renewal, and the cost function $\mathcal{C}(t)$ in (12) is the steady-state unavailability.

In the numerical experiments, reported in the following sections, the unavailability is considered as the representative measure of the system performance, and the influence of the two rejuvenation policies on the minimisation of the long-term system unavailability is discussed.

5. Examples

In this section, we apply the developed methodology to the analysis of two examples. The first one is a real-world case study arising in the context of database maintenance, where the issue is the development of optimal strategies for the permanent archiving of the redo log files (used to restore the contents of a DBMS in case of data losses). Although the system described here can be considered relatively obsolete, and as such no longer representative of modern DBMS systems, nevertheless, it serves as a simple illustration of the proposed modelling technique.

The second example consists of a set of numerical experiments aimed at investigating various features of the proposed methodology, and in particular the sensitivity of the proposed rejuvenation policies to the characteristics of the underlying degradation model. In this second example, the emphasis is on the behavioural aspects of the obtainable results rather than on the specific numerical values.

5.1. Rejuvenation model for permanent archiving of redo log files: a case study

Backup and restoration schemes are a vital part of any DBMS, since data losses caused by the occurrence of unpredictable events (such as hardware and/or software failures) are in general unacceptable. Periodic backups are usually performed with the aim of storing a stable copy of the database contents, from which the data state at the moment of the loss can be restored. Since backups alone do not provide complete protection against data losses (the data modifications performed by any committed transaction executed after the last backup cannot be recovered by simply restoring the archived data), additional mechanisms are required. A widely used approach to data restoration is based on the *redo log*, which is a file used to keep track of all the committed transactions. In the case of a data loss, the contents of the redo log are used to "roll forward" the most recently archived version of the database.

In this section, we address the problem of devising an optimal archiving strategy for an Oracle© database, managed by the CSI Piemonte (an institution belonging to the Italian Public Administration) on behalf of several Italian public agencies. In the case study considered here, log archiving has to be carried out *manually* by the DBMS administrator [1,2,12], who first brings the DBMS offline, then starts the archiving procedure from a primary disk to a secondary disk, and finally brings the DBMS online again. The considered archiving procedure of log files is rather antiquated since many modern commercial DMBSs provide facilities for *continuous* archiving without suspending the normal database operation.

A careful planning of redo log archiving is required in order to minimise the time in which the DBMS is not operational (the DBMS unavailability). Three main factors contribute to the non-operational time, namely the time elapsing from the instant in which the DBMS becomes non-operational to the instant in which the archiving process starts (*reaction time* t_r), the time taken to bring the DBMS offline and back online (*switch time* t_s), and the time required to transfer the redo log from the primary to the secondary disk (*transfer time* t_t).

The reaction time t_r depends on whether the archival has been planned or not. In the first case, the administrator voluntarily brings the DBMS offline, so a careful selection of the moment in which this activity is carried out minimises the reaction time. In the second case, the unplanned archival is forced by the filling up of the allowed disk area, but log archiving may start only after the administrator realises that the DBMS is offline. If this event occurs when the administrator is not alert (e.g., in night hours), the reaction time may be very long. Moreover, unplanned archival may happen when the database is heavily loaded, increasing the out-of-service cost. The switch time t_s is usually fixed, while the transfer time t_t depends on the disk-to-disk transfer rate and on the size of the log. Frequent archivals reduce the transfer time, but increase the total amount of switch time.

The considered problem of planning archival events can be formulated as a rejuvenation problem and analysed in the framework of the proposed methodology. The degradation process corresponds to the progressive occupation of the allocated disk space because of the growth of the redo log file. The rejuvenation events and crashes correspond to planned and unplanned archivals.

The first step for utilising the methodology presented in the previous sections is to characterise the degradation process. The observable degradation index $s(t)$ is the total occupancy level of the log file at time t (measured, for instance, in MB), and $s_{\max}$ is the maximum allowed dimension for the log file (3 GB in our case study). The log file size is monitored once a day ($\Delta t = 1$ day), so that the random increment X of the degradation index is the daily growth of redo log file whose cdf $F_X(x)$ needs to be experimentally estimated.

In the present case study, we have observed the behaviour of the log file for 82 consecutive days. The empirical cdf, built on the 82 sample points, has been reported in Fig. 3. We have assumed that $F_X(x)$ belongs to the cdf Gamma family $\Gamma(z, \lambda)$, and we have estimated the parameters of the Gamma cdf by a least square fitting technique over the empirical cdf. The calculated optimal Gamma cdf is also reported on Fig. 3, and corresponds to a value of $z = 0.54$ and $\lambda = 0.003104$ (with this value of z the Gamma distribution is decreasing hazard rate DHR [15]). The goodness of the fit has been tested resorting to the Kolmogorov–Smirnov test at a risk level $r_\alpha = 0.1$. The upper and lower bounds of the Kolmogorov–Smirnov goodness-of-fit test are reported on the figure, showing that the experimental points do not invalidate the hypothesis of being distributed according to a Gamma cdf.

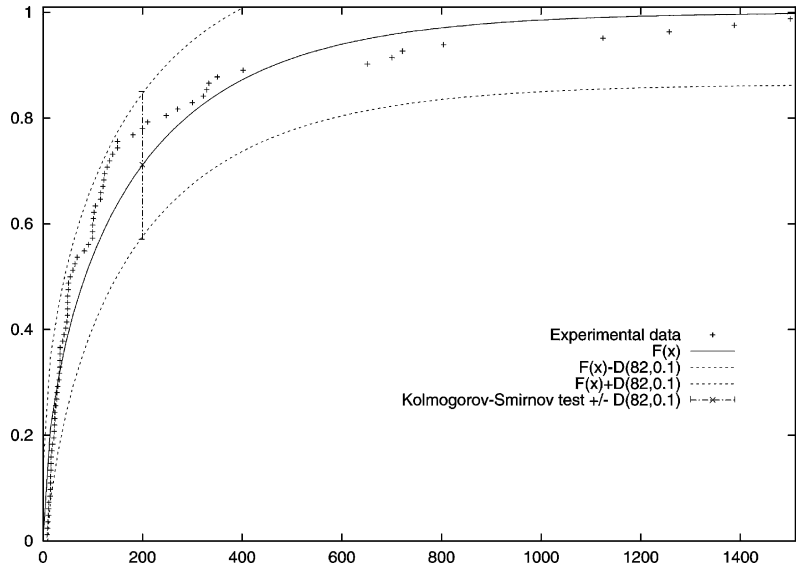

Fig. 3. Empirical cdf of log file's daily growth (crosses), the best fit $\Gamma(z, \lambda)$ cdf (solid line), and the upper and lower bounds of the Kolmogorov–Smirnov test at a risk level $r_\alpha = 0.1$ (dotted lines).

The Gamma family has many interesting properties that are explored in the next section; in particular, we make repeated use of the fact that the distributions of the Gamma family are closed under convolution [15]. [1]

The second step consists in evaluating the appropriate cost factors c_1 and c_2. The cost factor c_1 represents the out-of-service time in a rejuvenation action and is given by the sum of the switch time and the transfer time: $c_1 = t_s + t_t$.

The switch time t_s is roughly constant, being the result of several manual activities that have to be carried out in order to safely put offline the DBMS independently from the size of the redo log. In the present case study, the estimated value is $t_s = 4$ min. The transfer time t_t, conversely, depends on the amount of data that have to be moved from the primary to the secondary disk, and on the disk transfer rate. In this work we consider two possible scenarios: in the first one we used a disk transfer rate of 5 MB/s (typical of low-cost SCSI disks), while in the second one the above rate was set to 20 MB/s (achievable with a RAID system). The value of t_t in both cases has been estimated by assuming that the size of the redo log file was 3 GB, yielding to $t_t = 10$ and 2.5 min, respectively. By combining the above values of t_t with t_s, we obtain two possible rejuvenation cost factors, namely $c_1 = 14$ and 6.5 min.

The cost factor c_2 represents the extra contribution to the out-of-service subsequent to a crash and is given by $c_2 = t_r$. The estimation of the reaction time is more difficult since t_r depends on several factors, such as the time of the day in which the disk gets completely full, the immediate availability of the administrator, etc. It has been, however, observed that typical reaction times range from 10 to 20 min, so we have considered four different values for the crash cost, namely $c_1 + c_2 = 14 + 10 = 24$ min, $c_1 + c_2 = 14 + 20 = 34$ min, $c_1 + c_2 = 6.5 + 10 = 16.5$ min, and $c_1 + c_2 = 6.5 + 20 = 26.5$ min.

[1] If $W_1, W_2, \ldots, W_k$ is a sequence of mutually independent Gamma random variables (where W_i is $\Gamma(z_i, \lambda)$ for $i = 1, 2, \ldots, k$), then $S_k = \sum_{i=1}^{k} W_i$ has distribution $\Gamma(z, \lambda)$, where $z = z_1 + z_2 + \cdots + z_k$ (see [15] for details).

With the above numerical values, we can apply the two rejuvenation policies developed in the previous sections in order to provide optimal design parameters.

Rejuvenation under the alert policy. Under the alert policy, the design objective is the evaluation of an alert threshold s_{al}, expressed in percentage of the maximum size allocated for the redo log file, such that the rejuvenation (i.e., the transfer of the redo log file to the secondary disk), performed as soon as the actual occupancy of the log file reaches the value s_{al}, minimises the total DBMS unavailability.

By applying the methodology of Section 4.2, we have obtained the results reported in Fig. 4, where the contribution to the unavailability due to a rejuvenation (dotted line), the contribution due to a crash (dashed line) and the total unavailability (solid line) are plotted as a function of the ratio $\sigma_{al} = s_{al}/s_{max}$. In each graph, the minimum value of DBMS unavailability has been highlighted by using a '×' character placed on the Rej+Crash curves. As expected, the increase in the crash cost c_2 entails the detrimental effect of reducing the alert threshold.

Rejuvenation under the risk policy. Under the methodology proposed in Section 4.1, we can evaluate the DBMS unavailability as a function of the risk α, and, for each α, the corresponding value of the rejuvenation interval θ can be calculated. In Fig. 5, we have reported the system unavailability (in particular the contribution due to rejuvenation in dotted line, the contribution due to a crash in dashed line, and the total unavailability in solid line) as a function of the rejuvenation interval θ (in days).

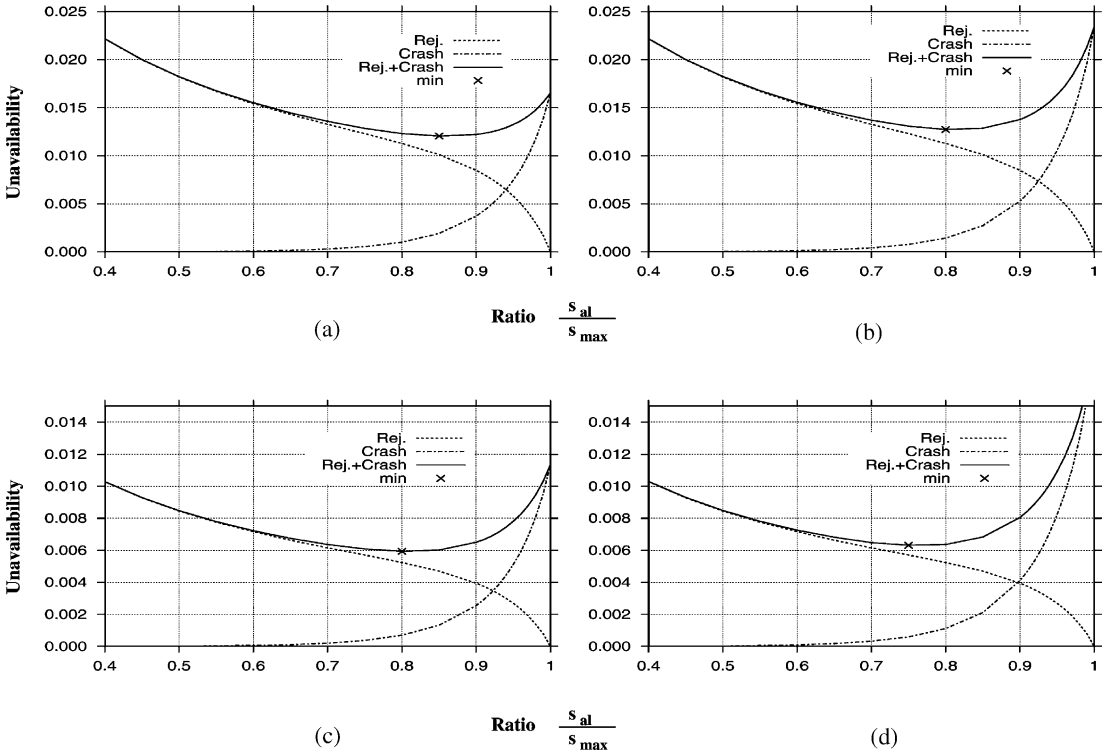

Fig. 4. Unavailability with different alert thresholds and $c_1 = 14$, $c_2 = 10$ min (a), $c_1 = 14$, $c_2 = 20$ min (b), $c_1 = 6.5$, $c_2 = 10$ min (c), $c_1 = 6.5$, $c_2 = 20$ min (d).

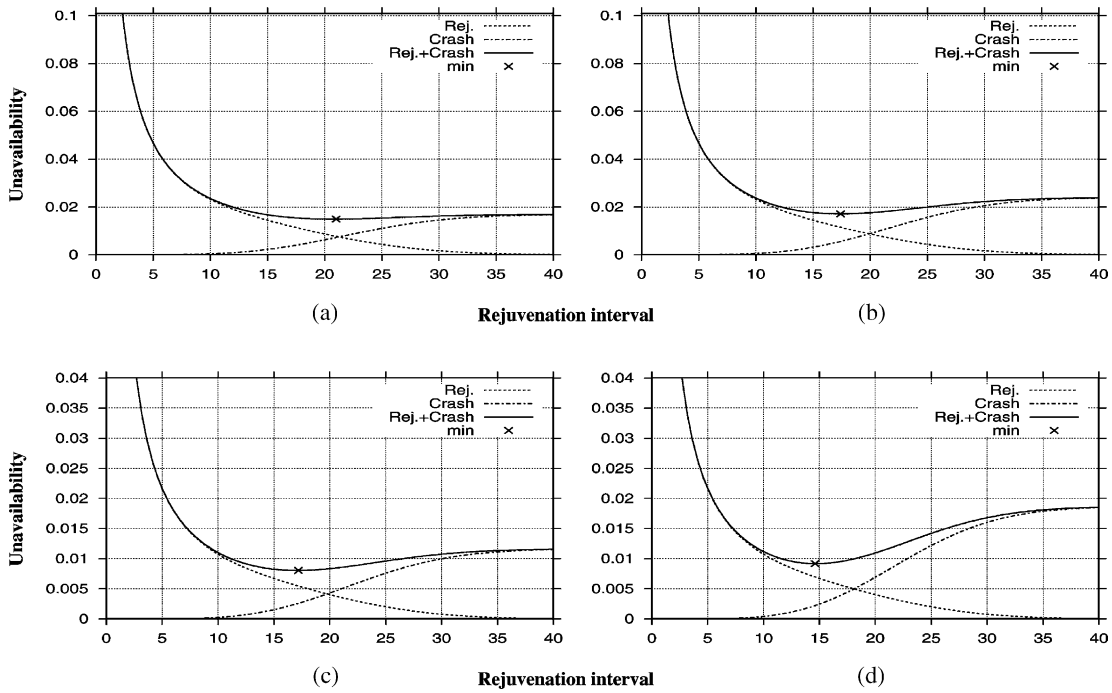

Fig. 5. Unavailability with different rejuvenation interval and $c_1 = 14$, $c_2 = 10$ min (a), $c_1 = 14$, $c_2 = 20$ min (b), $c_1 = 6.5$, $c_2 = 10$ min (c), $c_1 = 6.5$, $c_2 = 20$ min (d).

The optimal rejuvenation interval is the value of θ corresponding to the minimum unavailability which has been highlighted in Fig. 5 by means of a '$\times$' character placed on the Rej+Crash curves. Also in this case, increasing the cost factor c_2 implies more frequent rejuvenations with greater unavailability.

5.2. Sensitivity analysis of the rejuvenation models

In this example, we provide a set of numerical experiments to investigate the sensitivity of the methodology with respect to the drift distribution $F_X(x)$. To this end, we have experimented various cdfs having the same expected value but with different variances and different properties. For the sake of uniformity, we have assumed that the drift distribution is a gamma distribution $F_X(x) = \Gamma(z, \lambda)$, with appropriate parameters (z, λ). We have considered three cases:

1. $F_X(x) = \Gamma(0.2, 0.2)$. The cdf of the increment is decreasing hazard rate (DHR) with expected value equal to 1, variance equal to 5, and coefficient of variation equal to 2.24.
2. $F_X(x) = \Gamma(1, 1)$. The cdf of the increment is exponential (*constant hazard rate*) with expected value equal to 1, variance equal to 1, and coefficient of variation equal to 1.
3. $F_X(x) = \Gamma(5, 5)$. The cdf of the increment is an Erlang-5 (IHR-increasing hazard rate) with expected value equal to 1, variance equal to 0.2, and coefficient of variation equal to 0.447.

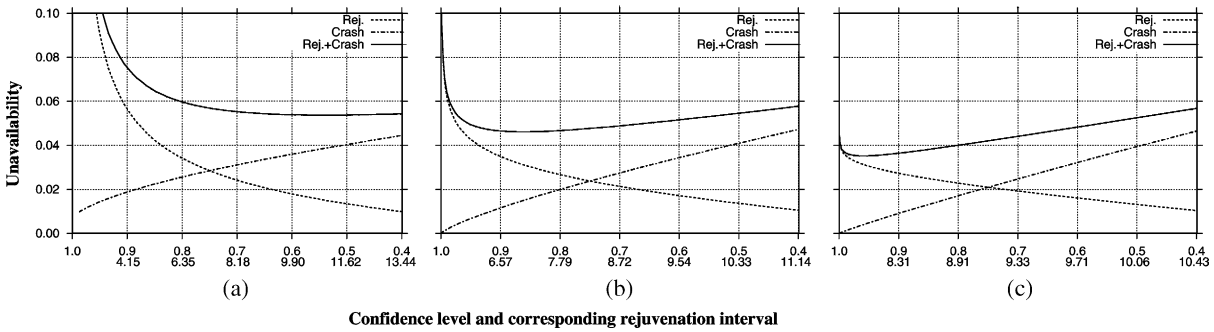

Fig. 6. Unavailability contributions with different confidence levels and $F_X(x) = \Gamma(0.2, 0.2)$ (a), $F_X(x) = \Gamma(1, 1)$ (b), and $F_X(x) = \Gamma(5, 5)$ (c).

All the successive numerical examples refer to a "failure profile" with the following characteristics:

- The feasible values of the degradation index are in the interval [0, 10), i.e., the minimum value of the degradation index is $s_{\min} = 0$ while the maximum is $s_{\max} = 10$.
- The down time to recover from a rejuvenation action is $c_1 = 15$ min; the down time to recover from a crash is $c_1 + c_2 = 45$ min.
- The degradation index is monitored once a day ($\Delta t = 1$ day).

In order to display the results with the different rejuvenation policies in a comparable form, all the plots report three curves for the steady-state unavailability: the contribution to the system unavailability of the rejuvenation (dotted line), the contribution to the system unavailability of the crash (dashed line) and the total system unavailability (solid line).

Rejuvenation under the risk policy. Fig. 6 plots the unavailability versus the confidence level α for the three considered cdfs $F_X(x)$. The plots are obtained for values of α ranging the interval [1, 0.4]. The second line under the x-axis reports the value of the rejuvenation interval θ (in days) corresponding to the value of α (Eq. (9)).

For instance, from Fig. 6(a), we derive that for $\alpha = 0.9$ the rejuvenation interval is $\theta = 4.15$ (days), the contribution to the unavailability due to the rejuvenation is 0.056, the contribution to the unavailability due to the crash is 0.019, and the total system unavailability is 0.075.

In all the numerical experiments the total system unavailability presents a minimum as a function of α. The corresponding value of θ can be utilised as a design parameter to find the optimal rejuvenation interval. With the reward values previously defined, the minimal unavailability is reached with a risk $(1-\alpha) = 0.46$ when $F_X(x) = \Gamma(0.2, 0.2)$ (Fig. 6(a)), with a risk $(1-\alpha) = 0.13$ when $F_X(x) = \Gamma(1, 1)$ (Fig. 6(b)), and with a risk $(1 - \alpha) = 0.04$ when $F_X(x) = \Gamma(5, 5)$ (Fig. 6(c)). The corresponding rejuvenation intervals are, in the three cases, $\theta = 11$, 7 and 7.4 days, respectively.

The numerical results are strongly dependent on the assumed values of the rewards c_1 and c_2. However, the minimum of the cost function can always provide an optimal design principle. For safety critical systems, in which the crash can have very catastrophic effects (see, e.g. [14]), the rejuvenation policy can be based on the risk only, without any consideration of economical factors. In any case, if we look at the three figures at the same level of the risk (say $\alpha = 0.9$), we can see that the rejuvenation interval becomes longer moving form (a) to (c) (decreasing the variance of the cdf $F_X(x)$), and, at the same time, the system unavailability decreases.

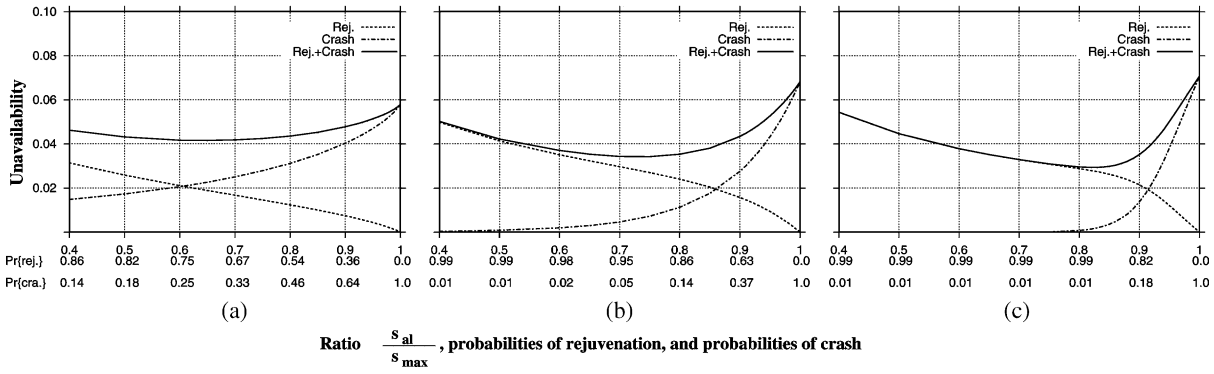

Fig. 7. Unavailability with different alert thresholds and $F_X(x) = \Gamma(0.2, 0.2)$ (a), $F_X(x) = \Gamma(1, 1)$ (b), and $F_X(x) = \Gamma(5, 5)$ (c).

Rejuvenation under the alert policy. Fig. 7 plots the unavailability contributions versus the ratio $\sigma_{al} = s_{al}/s_{max}$ for the three considered drift distributions $F_X(x)$. As in Fig. 4, the two successive lines of values under the x-axis report the corresponding rejuvenation probability (19) and the crash probability (20). Also with this policy, the unavailability attains a minimum that provides the optimal design parameter σ_{al}.

In particular, for $F_X(x) = \Gamma(0.2, 0.2)$, the minimal unavailability is achieved for a value $\sigma_{al} = 0.64$ (Fig. 7(a)); for $F_X(x) = \Gamma(1, 1)$, the minimal unavailability is achieved for $\sigma_{al} = 0.73$ (Fig. 7(b)); for $F_X(x) = \Gamma(5, 5)$, the minimal unavailability is achieved for $\sigma_{al} = 0.82$ (Fig. 7(c)).

Random shocks. Fig. 8 displays the unavailability of the system in three situations in which the software degradation is caused by faults or shocks occurring randomly in time according to a given point process $S(t)$. All the examples have been derived under the hypothesis that the drift distribution is exponential $F_X(x) = \Gamma(1, 1)$. Fig. 8(a) plots the unavailability obtained with equispaced observations, Fig. 8(b) plots the unavailability when the degradation is caused by a sequence of shocks occurring according a Poisson process with rate 1, while Fig. 8(c) shows the effect of shocks occurring according to an NHPP (see Eq. (8)) with parameters $c = 1$ and $\beta = 1.6$. All the pictures of Fig. 8 have been evaluated under the risk-level rejuvenation policy (Section 3.1 and are plotted versus the rejuvenation interval θ). The second line under the x-axis reports, for each value of the rejuvenation interval, the corresponding confidence level α (i.e., the probability of having the rejuvenation before the crash).

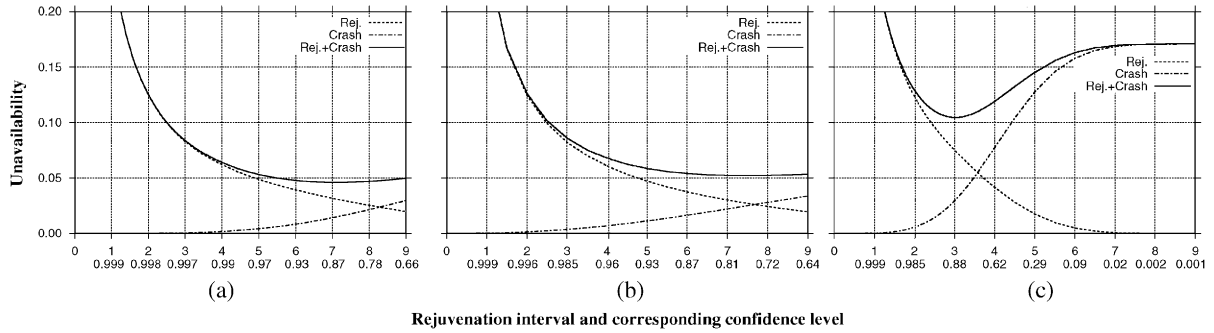

Fig. 8. Unavailability with $F_X(x) = \Gamma(1, 1)$: equispaced observations (a), sequence of shocks occurring according to a Poisson process with rate 1 (b), and sequence of shocks occurring according to an NHPP with $c = 1$ and $\beta = 1.6$ (c).

The acceleration in the occurrences of the successive degradation increments, observed when $S(t)$ is an NHPP with $\beta > 1$, provides a sharper minimum in the unavailability curve, so making the choice of the proper rejuvenation interval more critical. This result stresses the need of developing accurate and effective modelling techniques for software aging.

6. Conclusion

In this paper, we have discussed a new modelling technique for the quantitative analysis of smoothly degrading software systems. The aim of this work was to overcome the "black box" modelling approach previously proposed in the literature and to provide a more effective methodology to evaluate the convenience to perform preventive restoration actions known as rejuvenation. The proposed fine grained model is based on the assumption that it is possible to identify the current degradation of the system by monitoring an observable quantity referred to as the degradation index. A fully developed analytical treatment is formulated when the degradation index undergoes a monotonically increasing degradation process. Several practical cases can be identified, where the degradation index follows an increasing pattern. However, when this behaviour is not realistic (as in the case of a storage systems where the memory space can be requested or released), two modelling strategies can be followed:

1. only the "net" degradation is modelled, i.e. we sample the degradation index at successive peaks of increasing height (discarding the behaviour of the degradation index between two successive peaks);
2. the complete stochastic process of the degradation index is modelled, considering at each step the probability of the degradation index of moving up or down.

The first strategy can be accommodated in the present model if the statistics of the intervals between increasing peaks is collected. The second strategy requires a more complex formulation and solution of the related stochastic process and is matter for further research.

Two alternative policies have been investigated to determine whether and when to perform the rejuvenation. The evaluation of the optimal rejuvenation interval, under the two adopted policies, was performed by including a reward function into the model. The theory of renewal processes with rewards provided a means to estimate the system unavailability, and to derive the optimal policy as the one that minimises the unavailability.

The methodology has been applied to a case study aimed at evaluating an optimal strategy for copying a redo log file of a DBMS into a secondary disk in order to minimise the overall DBMS unavailability. Moreover, various features of the proposed rejuvenation strategies have been discussed as a function of the stochastic properties of the degradation process.

Acknowledgements

This work has been partially supported by E.U. under IST Programme IST-2000-5.1.4 "DEPAUDE", by the Italian MURST under Project "ISIDE", and by CNR under Grant No. 99.01716.CT01.

References

[1] Oracle7 Server for UNIX Administrator's Reference Guide, 1995 (Chapter 3: Required Operator Assistance).

[2] Informix Dynamic Server 7.3 Administrator's Guide, 1999, p. 18 (Chapter 18); available at the URL http://www.informix.com.
[3] A. Avritzer, E.J. Weyuker, Monitoring smoothly degrading systems for increased dependability, J. Emp. Software Eng. 2 (1) (1997).
[4] R.E. Barlow, F. Proschan, Statistical Theory of Reliability and Life Testing, Holt, Rinehart and Winston, New York, 1975.
[5] D.R. Cox, Renewal Theory, Chapman & Hall, New York, 1962.
[6] J.D. Esary, A.W. Marshall, F. Proschan, Shock models and wear processes, Ann. Probab. 1 (1973) 627–649.
[7] S. Garg, A. Pfening, A. Puliafito, M. Telek, K.S. Trivedi, Modelling and analysis of load and time-dependent software rejuvenation policies, in: Proceedings of the Third International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS3), Bloomingdale, IL, September 1996, pp. 35–39.
[8] S. Garg, A. Puliafito, M. Telek, K.S. Trivedi, Analysis of software rejuvenation using Markov regenerative stochastic Petri net, in: Proceedings of the Sixth International Symposium on Software Reliability Engineering (ISSRE95), Toulouse, France, October 1995.
[9] S. Garg, A. Puliafito, M. Telek, K.S. Trivedi, Analysis of preventive maintenance in transaction based software systems, IEEE Trans. Comput. 47 (1) (1998) 96–107 (special issue on dependability of computing systems).
[10] S. Garg, A. van Moorsel, Towards performability modelling of software rejuvenation, in: Proceedings of the Third International Workshop on Performability Modeling of Computer and Communication Systems (PMCCS3), Bloomingdale, IL, September 1996, pp. 40–44.
[11] Y. Huang, C. Kintala, N. Kolettis, N.D. Fulton, Software rejuvenation: analysis, module and applications, in: Proceedings of the 25th International Symposium on Fault-Tolerance Computing (FTCS-25), Pasadena, CA, USA, June 1995.
[12] P. Korth, A. Silberschatz, Database Systems Concepts, 2nd Edition, McGraw-Hill, New York, 1991 (Chapter 10).
[13] V.G. Kulkarni, Modelling and Analysis of Stochastic Systems, Chapman & Hall, New York, 1995.
[14] E. Marshall, Fatal error: how Patriot overlooked a Scud? Science 13 (1992) 1347 .
[15] K.S. Trivedi, Probability & Statistics with Reliability, Queueing and Computer Science Applications, Prentice-Hall, Englewood Cliffs, NJ, 1982.

Andrea Bobbio is presently Associate Professor of Computer Science at the "Universitá del Piemonte Orientale" in Alessandria, Italy. He graduated in nuclear engineering from Politecnico di Torino in 1969. From 1971 until 1992 he was at Istituto Elettrotecnico Nazionale Galileo Ferraris (Torino, Italy), where he was involved in researches on degradation mechanisms in electronic devices, system reliability, stochastic modelling, quantitative analysis, and optimisation. He is author of several reliability oriented papers, and he has been principal investigator in several grants with public and private Institutions.

Matteo Sereno was born in Nocera Inferiore, Italy. He received his Doctor degree in Computer Science from the University of Salerno, Italy, in 1987, and his Ph.D. degree in Computer Science from the University of Torino, Italy, in 1992. He is an Associate Professor at the Computer Science Department of the University of Torino. His current research interests are in the area of performance evaluation of computer systems, modelling of communication networks and parallel architectures, queueing network and stochastic Petri net models.

Cosimo Anglano is an Associate Professor of Computer Science at the Universitá del Piemonte Orientale "A. Avogadro", Alessandria, Italy. He received the Laurea degree and the Ph.D., both in Computer Science, from the University of Torino, Italy, in 1990 and 1994, respectively. His research focuses on scheduling algorithms for cluster computing, resource management algorithms for distributed systems, and performance evaluation of distributed systems. He is a member of the IEEE Computer Society.