Availability Modeling of Modular Software

James Ledoux

Availability Modeling of Modular Software

James Ledoux
Institut National des Sciences Appliquées

Key words – Markov chain, Counting process, Reliability growth, Poisson approximation.

Summary & Conclusions – Dependability evaluation is a basic component in the assessment of the quality of repairable systems. We develop here a general model specifically designed for software systems that allows the evaluation of different dependability metrics, in particular, of availability measures. The model is of the structural type, based on Markov process theory. In particular, it can be viewed as a attempt to overcome some limitations of the well-known Littlewood's reliability model for modular software. We give both the mathematical results necessary to the transient analysis of this general model and the algorithms that allow to evaluate it efficiently. More specifically, from the parameters describing : the evolution of the execution process when there is no failure, the failure processes together with the way they affect the execution, and the recovery process, we obtain the distribution function of the number of failures on a fixed mission period. In fact, we obtain dependability metrics which are much more informative than the usual ones given in a white-box approach. We briefly discuss the estimation procedures of the parameters of the model. From simple examples, we illustrate the interest in such a structural view and we explain how to take into account reliability growth of part of the software with the transformation approach developed by Laprie and al. Finally, the complete transient analysis of our model allows to discuss in our context the Poissonian approximation reported by Littlewood for its model.

1 Introduction

Dependability is a central concept in modern systems employing software. By far the most extensive work on software dependability assessment concerns the modeling of the growth in software reliability which takes place as a result of fault removal. The nearly exclusively used approach adopts the so-called *black-box* view of the system, in which only the interactions with the environment are considered. It generally gives a good understanding of the product's behavior during its development phase. Models associated with such an approach provide the Cdf of the s-time up to the first failure (see [1]). This output can help, for instance, to decide to stop the testing phase of a product.

If the system is already in operation, the reliability growth phenomenon is not so crucial and the behavior of the whole system is highly dependent on how the different components are requested. In this case, the *white-box* (or structural) point of view is an alternative approach in which the structure of the system is explicitly taken into account. This is advocated for instance in [2], [3], [4]. In particular, the structural approach allows one to analyze the sensitivity of the dependability of the system with respect to the dependability of its components. Moreover, in case of a software system having some recovery capability, it is of interest to make dependability predictions considering the influence of these recovery actions on the execution dynamics and of the delays to recover a safe state. These features are better covered when the components of the system and their relationships are explicitly considered. It is worth noting that reliability growth is introduced in software dependability assessment with a structural model in [5]. The used technique can be adapted to our context as it will be illustrated by a simple example.

In general, the underlying execution model in the structural approach is a Markov chain considering typically transitions among the execution of different components. Our work is based on the well-known Littlewood's model [6]. Our aim is to propose a structural model, working in continuous time, taking into account a class of failures which affect the execution process. Specifically, we can include mechanisms to recover a safe state for the system. A basic goal of our paper is to deal with the delays in recovering an operational state, in order to derive availability measures. In such a context, our main result is the distribution of the number of failures in a given bounded interval. This is a very informative r.v. Its distribution allows, in particular, to analyze the delay up to the first failure, which is the usual main feature of black-box models. We also consider in some detail the computation of the mean number of failures in the considered interval, the failure intensity function and the point availability. The contribution of such a structural model to the software dependability evaluation, in particular during its operational phase, will be illustrated on few simple situations. A discrete time counterpart of our work may be developed from Cheung's model [2],[7].

The paper is organized as follows. In the next section, we develop a continuous time model which overcomes some limitations of Littlewood's one [6], [3]. In particular, we deal here with more general and informative measures. The main result proposes an analytic and algorithmic representation of the distribution of the cumulative number of failures over a given interval. We also give specific results on the expectation of this r.v. The basic mathematical tool is the uniformization technique [8]. At the end of Section 2, we briefly outline the problem of the estimation of the family of parameters associated with our model. The last section is devoted to simple illustrations of the practical application of our results and how the reliability growth of some component may be accounted for. Finally, we discuss in our context, on the Poissonian approximation given in [6] for the Littlewood's model.

2 Software dependability with a Markovian structural model

With respect to the black-box approach, just a few papers have been published on structural software reliability models. A representative sample is (in discrete time) [2],[7],[9] and (in continuous time) [6],[3],[4], [5],[10].

First at all, we recall the main features of the Littlewood's model [6] which are common to most of the previous cited works. That will introduce in a natural manner, our nominal model developed in the next subsection. In a first step, Littlewood defines an execution model of the software. The basic entity is the standard software engineering concept of *module* as for instance in [2]. The software structure is then represented by the *call graph* of the set $\mathcal{C}$ of the modules. These modules interact by execution control transfer and, a each instant, control lies in one and only one of the modules which is called the active one. ¿From such a view of the system, we build up a stochastic process $X = (X_t)$ which indicates the active module at each time t. This process is assumed to be a homogeneous Markov chain on the set $\mathcal{C}$.

In a second step, Littlewood describes the failure processes associated with execution actions that gives a view of the software in operation. Failure may happen during a control transfer between two modules or during an execution period of any module. During a sojourn of the execution process in the module i, failures are part of Poisson process having parameter μ_i. When control is transferred between the modules $i \in \mathcal{C}$ and $j \in \mathcal{C}$, a failure may happen with probability $\mu(i,j)$.

The architecture of the software can be combined with the failure behavior of the modules and that of interfaces into an *operational model* which can then be analyzed to predict dependability of the software. This method is referred as the "composite-method" according to the classification of Markov models in the white-box approach proposed in [11]. Basically, we are interested in the process $(F_t)_{t\geq 0}$ where F_t is the cumulative number of failures observed at time t. If the inter-failure period are much smaller than the switching rates modules, Littlewood states that $(F_t)_{t\geq 0}$ is a Poisson process with parameter

$$\lambda = \sum_{i\in\mathcal{C}} \pi(i)[\mu_i + \sum_{j\in\mathcal{C}} \mu(i,j)]$$

where π is the stationary distribution corresponding to the Markov process X. Analogous results are reported in [3] when X is assumed to be a semi-Markov chain. Such results give a "hierarchical-method" [11] for reliability prediction: we solve the architectural model and superimpose the failure behavior of the modules and that of the interfaces on to the solution to predict reliability. Note that:

- failures have no influence on the execution process. The software is instantaneously operational (with probability 1) even if delays are invoked in using some handling errors capability.
- User has a priori no information on the quality of an eventual Poissonian approximation.

The Subsection 2.3 proposes an algorithmic and analytic assessment of a richer operational model evolving in continuous time. That includes the transient analysis of the counting process associated with Littlewood's model. We quote that a recent work [12] proposes an extension of the Littlewood's model to take into account the effect of the workload on the performance of the software in a multiuser environment. However only steady-state measures are provided for 2-users, 1-module systems.

2.1 A nominal model

The first step in the structural approach to evaluate the dependability of a complex software system is to define an *nominal model* taking into account the knowledge of the structure, the potentially available data and the measures that will be calculated. Concerning the first point, the main assumption is that the system can be seen as a set of interacting *components*. This interaction is only made by execution control transfer and, at each instant, control lies in one and only one of the components, which will be called the *active* one. For a discussion on the concept of software component in the context of this paper the reader can see [10]. Component definition is a user-level task, depending on the system being analyzed, on the possibility of getting the required data, etc. As previously quoted, the first idea is to identify the components with the modules. The main problem with this approach is that the Markovian (or semi-Markovian) assumption for the call graph may be too much unrealistic. For instance, suppose that module M_i receives the execution control sometimes from module M_j, sometimes from module M_k. In many situations, the conditions under which the control is transferred and the characteristics of the tasks that must be performed by M_i may be completely different in the two cases. If M_i is a state of a Markov process, then the r.v. "nth sojourn time in M_i" is independent of the other sojourn times (in M_i and in the other states), does not depend on n and does not depend on the identity of the module from which the control is got. Thus, such a Markovian assumption may be too strong to be acceptable. Similar considerations can be done on the exponential distribution of the sojourn time in a module, on the switching transition probabilities between states, etc.

To avoid (at least, theoretically) the preceding drawbacks, we follow a different approach. Consider a partition of the whole code and call components its elements. Assume, for instance, that control starts in component C_i. After executing some subset of internal code, the services of component C_j are requested. Once C_j finishes this job, C_i ends in turn and control is transferred to component C_k which, in turn, request immediately the service of C_j. Instead of constructing three states C_i, C_j and C_k, we propose the scheme of Figure 1.

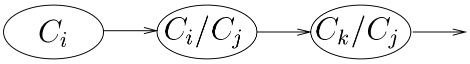


Figure 1: A component requested from two different places

The state labeled 'C_i/C_j' represents the execution control in C_j when it has been transferred from C_i and the same for 'C_k/C_j'. The main idea is to store in a state information on the *path* (of components) having transferred the control, if it is necessary to make the Markovian assumptions realistic. As in previous structural models, when a component C_i transfers the control to either C_j or C_k, we will model the situation by means of probabilistic jumps of the associated stochastic process. We will not discuss on the *flow process modeling* aspect further. The reader can find similar constructions to handle control structures (loops, tests, ...) for in Beizer's book [13]. Needless to say, it is clear that we have to identify only a few components leading to a moderate number of states ("labeled paths") because a model is not extremely useful without data and the more complex it is, the more difficult is the measurement task. This situation is also met in other software engineering activities as in testing, where the models used are very close to the previous one (see [13]). To end this discussion, it remains to underline that the goal here is to give a general description of the approach, not a complete specification of a new formalism. This must depend at least on the type of implementation language(s), the required detail level, the mechanisms that need to be included in the model, etc.

We will assume in the sequel, the nominal model is represented as the state graph of a homogeneous time-continuous Markovian process X. Its state space is denoted by $\mathcal{C} = \{1, 2, \ldots, M\}$. The process is characterized by its infinitesimal generator, denoted by $Q = (Q(i,j))\ i,j \in \mathcal{C}$, and by its initial distribution $\alpha = (\alpha_1, \ldots, \alpha_M)$. The entries $Q(i,j)$ are to be interpreted as the output rate from state i to state j, i.e. the parameter of the exponential distribution of any sojourn time in the state i weighted by the proportion of routing to state j, in absence of any failure phenomenon. The distribution α can be seen as the vector composed by the respective proportion of input nodes selection in the graph by the user, and thus it is called the *using profile*. We will assume X irreducible. This is because either the user does not distinguish any particular run end in its model and considers the continuity of the execution, or because there are states i representing final tasks and, in that case, we add transitions from i to every state j with transition probability α_j.

2.2 The operational model

In this section we develop a model which represents the system in operation. Formally, this consists in constructing a more complex stochastic process starting from X, by considering the fact that failures may happen and their possible effects on the system. We divide the failures in two types. The first one is composed by the *primary failures* and the occurrence of such a failure is a *primary event*. A primary failure leads to an execution break; the execution is restarted after some delay for instance from a checkpoint or perhaps from the beginning. The second class of failures is composed by the *secondary failures* (their occurrences are the *secondary events*). Such failures have no influence on the evolution of the model in that the execution process is assumed to be restarted instantaneously where the failure appears. They are those which are taken into account in [6].

Next, retrieving a safe state involves, in general, a random delay. Our model will allows us to assess the quality of the service with respect to the alternation continuity-suspension of the execution process, i.e. the availability of the system (see [8]). We suppose that the delay following an execution break is a random variable with a phase type (PH) distribution (see [14] for information on such a distribution). In that case, we denote the set of transient states associated with this PH distribution by $\mathcal{R}$ and these states will be referred as the *recovery states*. The sub-generator composed by the transition rates between elements of $\mathcal{R}$ is denoted by R. The phase type assumption allows us to represent times to recover an operational state which depend on the identity of the failed component. Indeed, the sequence of successive visited states in $\mathcal{R}$ can be related to the first state entered, which is the first recovery state selected by the failed component.

So, the nominal process is replaced by a process modeling the alternation operational-recovery periods. The main mathematical assumptions are the following.

- When the active component is i, a primary failure occurs with constant rate λ_i. The secondary events (secondary failures) are part of a Poisson process having rate μ_i. Always during a sojourn in state i, the two types of events are independent of each other.

- When control is transferred between two components, failures may also happen. If i and j are two states of the model such that transitions from i to j model control transfers, then a primary (resp. secondary) interface failure occurs with probability $\lambda(i,j)$ (resp. $\mu(i,j)$.) In order to simplify the technical evaluation of the model, we accept the occurrence simultaneously of the two types of failures (with probability $\lambda(i,j)\mu(i,j)$), with the result that the primary one will be taken into account only. In the model, we set $\lambda(i,j) = \mu(i,j) = 0$ if the transitions from state i to state j do not correspond to control transfers.

The introduction of distinct failure parameters for a transfer of control and for an "internal" execution is needed if the failure data are not gathered in a representative environment of the interactions between the components.

- Given a sequence of executed components, the failure processes associated with each state are independent. Also, the interface failure events are independent on each other and on the failures processes occurring when a component is active.

- If a primary event occurs during the execution of component i or when a control transfer takes place from component i, then $\alpha(i,j)$ is the probability that $j \in \mathcal{R}$ is the first visited recovery state.

- Let us define $X^* = (X^*_t)_{t\geq 0}$ as the process which gives either the active component or the recovery state reached at time t. Its state space is the set $\mathcal{E} = \mathcal{C} \cup \mathcal{R}$. Given a sequence of states of $\mathcal{E}$, the failure and recovery processes are assumed to be independent too. We assume that matrix $S = (S(i,j))_{(i,j)\in\mathcal{R}\times\mathcal{C}}$ of the transition rates between a recovery state to an operational state is given. We have $(R + S)\mathbf{1}^{\mathsf{T}} = \mathbf{0}$. The alternation operational-recovery periods is infinite by assuming the generator Q^* of the Markov chain X^* is irreducible:

$$i,j \in \mathcal{C} : Q^*(i,j) = Q(i,j)(1-\lambda(i,j)) \ \text{ if } i \neq j \quad \text{ and } \quad Q^*(i,i) = Q(i,i) - \lambda_i;$$

$$i \in \mathcal{R} : Q^*(i,j) = R(i,j) \ \text{ if } j \in \mathcal{R} \quad \text{ and } \quad Q^*(i,j) = S(i,j) \ \text{ if } j \in \mathcal{C};$$

$$(i,j) \in \mathcal{C} \times \mathcal{R} : Q^*(i,j) = [\ \lambda_i + \sum_{k \neq i, k \in \mathcal{C}} Q(i,k)\lambda(i,k)\ ]\alpha(i,j).$$

The proposed set of failure processes can be seen as the superposition of the failure processes considered in [6] (or [3]) and [4]. In papers like [6] the considered failures are those called secondary here. We see this fact as a limitation and our model attempts to avoid it by introducing the primary failures which affect the execution process.

It is worth noting that distributions $(S(i,j))_{j\in\mathcal{C}}$ ($i \in \mathcal{R}$) may represent a re-direction of the control flow which results from use of recovery actions. For instance, (primary) failures in a critical component that provoke a complete reset of the system is handled by setting, for any $j \in \mathcal{C}$, $S(i,j) = \alpha_j$ if i is the last state corresponding to the recovery phase; it means that the jump is performed according to the using profile distribution, independently of the state in which the failure occurs. Finally, let us drop the recovery data. If a failure happens "in" component i then the probability distribution $(\alpha(i,j))_{j\in\mathcal{C}}$ models the effect on the execution control.

2.3 Model analysis

The r.v. F_t is the cumulative number of primary or secondary failures over the interval $]0,t]$. At $t=0$, we assume that F_0 is 0, i.e. $\Pr(F_0 = 0) = 1$. We propose to deal with the following dependability indicators: the distribution function and the expectation of the r.v. F_t; the time to the first failure; the point availability and the failure intensity function.

The first active component is $i \in \mathcal{C}$ with probability α_i (i.e. $\Pr(X^*_0 = i) = \alpha_i$) and for all $i \in \mathcal{R}$,

$\Pr(X_0^* = i) = 0$. To analyze the process $F = (F_t)_{t \geq 0}$, let us consider the bi-dimensional time-continuous process $(F, X^*) = (F_t, X_t^*)_{t \geq 0}$. It follows from the independence assumptions that this is a homogeneous Markovian process over the state space $\mathbb{N} \times \mathcal{E}$. Let us denote

$$\begin{aligned} a(i,j) &= \text{transition rate from state } i \in \mathcal{C} \text{ to state } j \in \mathcal{C} \text{ without any failure,} \\ d^{(s)}(i,j) &= \text{transition rate from state } i \in \mathcal{C} \text{ to state } j \in \mathcal{C} \text{ with a secondary failure occurrence,} \\ d^{(p)}(i,j) &= \text{transition rate from state } i \in \mathcal{C} \text{ to recovery state } j \text{ with a primary failure occurrence.} \end{aligned}$$

We denote by $F_t^{(p)}$ (resp. $F_t^{(s)}$) the cumulative number of primary (resp. secondary) failures observed over $]0,t]$. Therefore, for all $t \geq 0$, we have by definition $F_t = F_t^{(p)} + F_t^{(s)}$.

The following expressions for these rates can be derived by listing the different failure or recovery events:

$$\begin{array}{rcll} a(i,j) &=& Q(i,j)(1-\lambda(i,j))(1-\mu(i,j)) & \text{if } i \neq j \text{ et } i,j \in \mathcal{C}, \\ a(i,i) &=& 0 & \text{if } i \in \mathcal{C}, \\ d^{(s)}(i,i) &=& \mu_i & \text{if } i \in \mathcal{C}, \\ d^{(s)}(i,j) &=& Q(i,j)[1-\lambda(i,j)]\mu(i,j) & \text{if } i \neq j \text{ et } i,j \in \mathcal{C}, \\ d^{(p)}(i,j) &=& [\,\lambda_i + \sum_{k \neq i, k \in \mathcal{C}} Q(i,k)\lambda(i,k)\,]\,\alpha(i,j) & \text{if } (i,j) \in \mathcal{C} \times \mathcal{R}. \end{array}$$

After checking that, for any i, we have $\sum_{j \in \mathcal{C}}[a(i,j) + d^{(s)}(i,j)] + \sum_{j \in \mathcal{R}} d^{(p)}(i,j) = -Q(i,i) + \lambda_i + \mu_i$ and denoting by δ_i this value, let us define the three matrices A, $D^{(p)}$ and $D^{(s)}$ by

$$A = (a(i,j))_{i,j \in \mathcal{C}} - \text{diag}(\delta_i)_{i \in \mathcal{C}} \qquad D^{(p)} = (d^{(p)}(i,j))_{(i,j) \in \mathcal{C} \times \mathcal{R}} \qquad D^{(s)} = (d^{(s)}(i,j))_{i,j \in \mathcal{C}}$$

(diag(δ_i) denoting the diagonal matrix with value δ_i in its (i,i)-entry). The generator of the homogeneous Markov chain X^* over state space $\mathcal{E}$ may be rewritten as

$$Q^* = \begin{pmatrix} A + D^{(s)} & D^{(p)} \\ S & R \end{pmatrix}.$$

Example 1 Let us briefly discuss now some particular cases of the present operational model Op. To replace these well-known models in their usual framework, we drop the recovery delays assumptions.

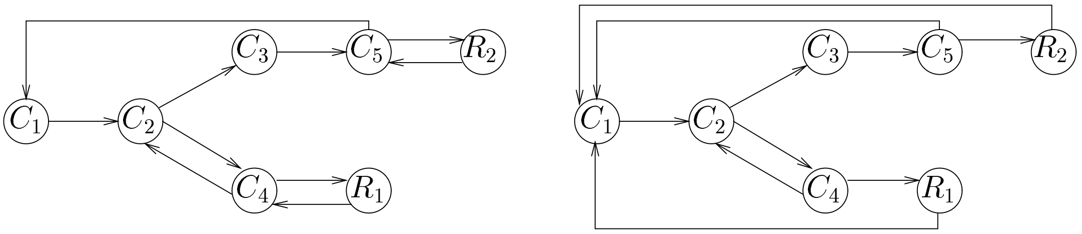
The first one is the Littlewood's model. There is no primary failure, that is $\lambda_i = 0$ and $\lambda(i, j) = 0$ for all $i, j \in \mathcal{C}$. Hence, we consider the point process of the only secondary events. Note that the generator $Q^* = A + D^{(s)} + D^{(p)}$ of the chain X^* given the active component in the operational model is also the generator Q of the execution process X. We retrieve the fact that the failures do not affect the evolution of the execution process.

Another interesting point process is the one obtained from Littlewood's model by assuming that the probability of a secondary failure during a control transfer is 0. Adding conditions $\mu(i, j) = 0$ for all $i, j \in \mathcal{C}$ in the previous context, we get $A = Q - \text{diag}(\mu_i)$ and $D^{(s)} = \text{diag}(\mu_i)$. This is a Markov Modulated Poisson Process (or MMPP) extensively used for instance to analyze the performance of data statistical multiplexers.

Example 2 Let us assume that our software system is constituted of five components denoted by $C_i,\ (i = 1, \ldots, 5)$. The parameters of their exponential execution times have been estimated to $-Q(1, 1) = 1$, $-Q(2, 2) = 0.5$, $-Q(3, 3) = 0.5$, $-Q(4, 4) = 1$, $-Q(5, 5) = 1$ where time is in hours. Figure 2, confined to the nodes associated with the five components, gives the transition graph of the nominal process X of the system.

Concerning the transition rates between components, we have $Q(1, 2) = 1$, $Q(2, 3) = 0.025$, $Q(2, 4) = 0.475$ and $Q(3, 5) = 0.5$. For the sake of simplicity, we consider only primary failures and the failure rate for each component is assumed to take into account the occurrence of internal and interface failures. Components C_4 and C_5 are the only ones for which a non-zero failure rate is available: $\lambda_4 = 0.03$ and $\lambda_5 = 0.01$. After a failure has occurred in C_4 (resp. C_5), retrieving a operational state involves a s-time distributed as an exponential distribution with parameter $-R(1, 1) = 5$ (resp. $-R(2, 2) = 10$). The two recovery states are denoted by R_1 and R_2.

We define two different operational models $(Op)_1$ and $(Op)_2$ from the distinct conditions to recover a safe state after a failure. We assume for $(Op)_1$ that execution restarts from the component which has failed as in [3]. The transition graph of $(Op)_1$ is given by in the l.h.s of Figure 2. For the second model $(Op)_2$, execution restarts always in component C_1. The transition graph is in the r.h.s of Figure 2.



Model $(Op)_1$ Model $(Op)_2$

Figure 2: The operational model with five components and two recovery states

2.3.1 The distribution function of F_t

We state the main result. This result is obtained in a similar way as in [15, Th3.1] with help of the well-known uniformization technique [8]. Therefore the proof is omitted.

Result 2.1 *Let us set* $u = \sup\{\ i \in \mathcal{C},\ \ j \in \mathcal{R}\ :\ |A(i,i)|, |R(j,j)|\ \}$. *For all* $t \geq 0$, *we have*

$$\Pr(F_t \leq k) \quad = \quad \text{poif}(k, ut) + \sum_{h=k+1}^{\infty} e^{-ut} \frac{(ut)^h}{h!} \alpha x_{\mathcal{C}}^{\mathsf{T}}(k,h) \tag{1}$$

where the column vector $x_{\mathcal{C}}^{\mathsf{T}}(k,h)$ *is deduced from the system of relations:*

$$\left.\begin{array}{rcll} x_{\mathcal{C}}^{\mathsf{T}}(k,h) & = & \widehat{A}\, x_{\mathcal{C}}^{\mathsf{T}}(k,h-1) + \widehat{D}^{(p)}\, x_{\mathcal{R}}^{\mathsf{T}}(k,h-1) + \widehat{D}^{(s)}\, x_{\mathcal{C}}^{\mathsf{T}}(k-1,h-1) & h,k \geq 1; \\ x_{\mathcal{C}}^{\mathsf{T}}(0,h) & = & \widehat{A}\, x_{\mathcal{C}}^{\mathsf{T}}(0,h-1) & h \geq 1; \\ x_{\mathcal{C}}^{\mathsf{T}}(k,0) & = & \mathbf{1}^{\mathsf{T}} & k \geq 0; \\ x_{\mathcal{R}}^{\mathsf{T}}(k,h) & = & \widehat{R}\, x_{\mathcal{R}}^{\mathsf{T}}(k,h-1) + \widehat{S}\, x_{\mathcal{C}}^{\mathsf{T}}(k-1,h-1) & h,k \geq 1; \\ x_{\mathcal{R}}^{\mathsf{T}}(k,0) & = & \mathbf{1}^{\mathsf{T}} & k \geq 1. \end{array}\right\} \tag{2}$$

where $\widehat{A} = I + A/u$, $\widehat{R} = I + R/u$, $\widehat{D}^{(s)} = D^{(s)}/u$ *and* $\widehat{D}^{(p)} = D^{(p)}/u$.

Result 2.1 leads to evaluate the probability $\Pr(F_t \leq k)$, for k and t fixed, by the series (1). We recall the inherent error due to the use of the uniformization technique: if we truncate the series (1) at level $H > k$ then

$$\sum_{h=H+1}^{\infty} e^{-ut}\frac{(ut)^h}{h!}\alpha x_C^{\mathsf{T}}(k,h) \leq \text{poisfc}(H, ut) \quad \text{(since } \alpha x_C^{\mathsf{T}}(k,h) \leq 1\text{)}.$$

Hence, the algorithm can be resumed as (a) choose the tolerance error ε; (b) compute H such that poisfc$(H, ut) < \varepsilon$; (c) compute the vectors $x_C^{\mathsf{T}}(k,h)$ for $h = 0, \ldots, H$ with system (2).

Example 3 (Example 2 continued) The two models are analyzed by means of the matrices

$$A = \begin{pmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & -0.5 & 0.025 & 0.475 & 0 \\ 0 & 0 & -0.5 & 0 & 0.5 \\ 0 & 1 & 0 & -1.03 & 0 \\ 1 & 0 & 0 & 0 & -1.01 \end{pmatrix} \quad D = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0.03 & 0 \\ 0.01 & 0 \end{pmatrix} \quad R = \begin{pmatrix} -5 & 0 \\ 0 & -10 \end{pmatrix},$$

$$S_1 = \begin{pmatrix} 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 10 \end{pmatrix} \text{ for } (Op)_1 \quad \text{and} \quad S_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 10 \end{pmatrix} \text{ for } (Op)_2.$$

Figure 3 gives the graph of the function $t \to \Pr(F_t \leq k)$ for $(Op)_1$ (with $k = 0, \ldots, 3$). Numerous dependability indicators follow from the knowledge of the distribution of F_t. For $k = 0$, we have the zero-failure feature of the system, that is its reliability function. It also allows us to answer a question as “up to what time t the probability to observe at most 2 failures is greater than 0.95 ?” We get $t \leq 88h$ using the algorithm.

Time to the first failure. It follows from the second and third renewal equations in (2) that the time to the first failure T for the operational model Op has a phase-type distribution:

$$\Pr(T > t) = \Pr(F_t = 0) = \alpha e^{At}\mathbf{1}^{\mathsf{T}}.$$

Then, the previous algorithm is merely the computation of the state probability for an absorbing Markov process by means of the uniformization technique.

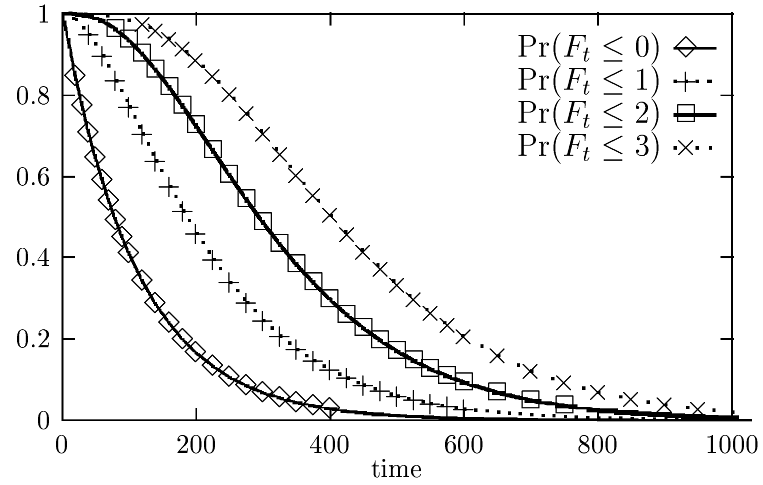


Figure 3: Function $(\Pr(F_t \leq k))_{t \geq 0}$ for $(Op)_1$ with $k \leq 3$

Point availability. The point availability function is defined at t by the probability $A(t)$ that the system is operational at t. This means that the identity process X^* indicate one of the components at time t:

$$A(t) = \sum_{i \in \mathcal{C}} \Pr(X_t^* = i) = \sum_{i \in \mathcal{C}} \left[(\alpha, \mathbf{0}) e^{Q^* t} \right] (i).$$

Since X^* is an irreducible chain, the row vector $(\alpha, \mathbf{0}) e^{Q^* t}$ tends componentwise to the stationary distribution π as t tends to infinity. We retrieve the usual expression $\sum_{i \in \mathcal{C}} \pi(i)$ of the asymptotic point availability $A(\infty)$.

2.3.2 Expected number of failures over $]0, t]$

Let us define the column vector D^T over $\mathcal{E}$ by

$$D^\mathsf{T} = \begin{pmatrix} (D^{(p)} + D^{(s)}) \mathbf{1}^\mathsf{T} \\ \mathbf{0} \end{pmatrix}.$$

The scalar $D^\mathsf{T}(i)$ ($i \in \mathcal{C}$) gives the occurrence rate of a failure “from” the component i. Vector $\widehat{D^\mathsf{T}}$ will denote D^T/u. The expectation of the r.v. F_t is obtained in computing the series $\sum_{k \geq 0} \Pr(F_t > k)$

with help of formula (1):

$$\forall t > 0 : \quad \mathrm{E}[F_t] \;=\; \sum_{h=1}^{\infty} e^{-ut}\frac{(ut)^h}{h!}\,(\alpha, \mathbf{0})\left(\sum_{k=0}^{h-1} \widehat{Q^*}^{k}\right)\widehat{D}^{\mathsf{T}} \tag{3}$$

where $\widehat{Q^*} = I + Q^*/u$. This series may be evaluated in the same way as for the distribution function. We deduce $\mathrm{E}[F_t^{(p)}]$ and $\mathrm{E}[F_t^{(s)}]$ from the relations $\mathrm{E}[F_t^{(p)}] = \mathrm{E}[F_t]\,\big|_{D^{(s)}=\mathbf{0}}$ (see Remark below) and $\mathrm{E}[F_t^{(s)}] = \mathrm{E}[F_t] - \mathrm{E}[F_t^{(p)}]$.

Failure intensity function. In conditioning with respect to the visited state by the Markov chain X^* at time t, the failure intensity function $h(t)$ is easily derived:

$$h(t) = \lim_{dt \to 0} \frac{1}{dt}\mathrm{Pr}(F_{t+dt} - F_t = 1) = (\alpha, \mathbf{0})e^{Q^* t}D^{\mathsf{T}}.$$

We note that this function tends to $\lambda^* = \sum_{i \in \mathcal{C}} \pi(i)[\sum_{j \in \mathcal{C}} D^{(s)}(i,j) + \sum_{j \in \mathcal{R}} D^{(p)}(i,j)]$ as t grows.

The previous expression of the failure intensity allows us to represent $\mathrm{E}[F_t]$ as

$$\mathrm{E}[F_t] = \int_0^t h(s)ds = \sum_{i \in \mathcal{C}} \mathrm{E}[S_i(t)]\;[\sum_{j \in \mathcal{C}} D^{(s)}(i,j) + \sum_{j \in \mathcal{R}} D^{(p)}(i,j)] \tag{4}$$

where $\mathrm{E}[S_i(t)]$ is the mean execution time of the component i over $]0, t]$. Finally, we obtain from (4) that

$$\forall t > 0, \quad \mathrm{E}[F_t] = (\pi D^{\mathsf{T}})t + (\alpha, \mathbf{0})(I - e^{Q^* t})(\mathbf{1}^{\mathsf{T}}\pi - Q^*)^{-1}D^{\mathsf{T}}. \tag{5}$$

The rate $\lambda_d^* = \pi D^{\mathsf{T}}$ is the so-called *fundamental rate* of the (failure) point process in the Neuts's terminology (see [14]). Note that $\lambda_d^* = \sum_{i \in \mathcal{C}} \pi(i)\;[\sum_{j \neq i} Q(i,j)\mu(i,j) + \mu_i]$ for the Littlewood's model (with $R = S = \mathbf{0}$), where π is actually the stationary distribution of the execution process. This the parameter of the Poisson process used in [6] as approximate failure point process. In the model MMPP, we have $\sum_{i \in \mathcal{C}} \pi(i)\mu_i$.

Remark If we are not interested in counting the secondary failures, that is, if we want to evaluate the distribution of $F_t^{(p)}$ only, it is sufficient to set parameters $(\mu(i,j), \mu_i)_{i,j \in \mathcal{C}}$ to 0 in the definitions of matrices A and $D^{(s)}$. This remark holds also for evaluate the number of failures corresponding to a subset of components in the Littlewood's model. Indeed, since all failures have no influence on the execution process, i.e. $X^* = X$, it is sufficient to assume that all the parameters associated with the events having no incidence on the interesting part of the system are 0. For instance, if we are only interested in the failures invoked by the component i then we set to 0 all the parameters $\mu_k,\ \mu(k,j)$ $k \neq i, j \in \mathcal{C}$, and we use Result 2.1 and (3).

2.4 Estimation and use

A major limitation to the validation of software reliability models was during a long time the lack of published data. It seems to be overcome in the context of black-box modeling [16]. Typical data associated with this approach is the sequence of failure instants or simply the number of failures observed over a fixed period. In general, some parametric model of the system is chosen and the observations of the failure times (or of the number of failures) are used to adjust the parameter values. However, the problem of data is not covered with structural modeling. To the best of our knowledge, there exists no published data which can give support to the general validation of the structural model approach. Concerning its application to a specific system, let us briefly outline the principal problems related to the estimation of the parameters (see also [17] for a complete discussion).

Prior to the complete system integration it can be envisaged that the failure rates of the components will be estimated, for example, by means of simulations or using the numerous black-box software reliability models [1]. A careful examination of the consequences of these failures must be done to classify them into primary or secondary. Of course, we will not necessarily have $\lambda_i \neq 0$ for each component i, and the same concerning the secondary failures.

Let us say now a few words about the behavior of the repair time. The problem of fitting a PH-distribution from the observation of its two first moments, for instance, is standard. Recall that, in particular, this allows to represent regular delays, where the exponential assumption would be too far from reality. Again, it is likely that the user will be able to establish a priori what component will be executed after a given recovery period following a specific primary failure. It can then be expected that parameters $S(i, j)$ can be evaluated prior to the integration phase. The same remark holds with respect to the parameters $\alpha(i, j)$. Indeed, the constant probability $\alpha(i, j)$ can be viewed as the proportion of primary failures (attached to component i) which involve a delay distributed according to a phase-type distribution with first state j.

Let us look now at the remaining matrices that need to be estimated. The simpler way to have information on the interface failures consists in measuring the proportion of $i \to j$ component transfers which fail. The estimation of the $Q(i, j)$ transition rates implies in turn to count the number of times each of the possible transitions occur. Many transition rates $Q(i, j)$, probabilities $\lambda(i, j)$, $\ldots$ will be zero since most of the transitions will probably be impossible. Some of the remaining non-null values can perhaps be “exactly” known, for instance from the analysis of the programs structure. In the same way, we can suppose that in many applications there will be no secondary event in the model. In any case, a general observation about model design is that the final number of components should be kept as small of possible, the main reason for this being the availability of data concerning failure rates and failure probabilities. This leads to look for a tradeoff between the description power of the model and the problem of estimating the parameters. Note that an alternative to analytical models as discussed in this paper may be based on discrete-event simulation (see [18]).

3 Various comments

3.1 Further analysis of Example 2

The proposed structural model allows to analyze the sensitivity of the dependability of the whole system with respect to the dependability of its components. For instance, suppose that we want to achieve a probability greater than 0.95 to observe at most two failures over a time period of at least ten days (i.e. over $]0, t_{inf}]$ with $t_{inf} = 240h$). A failure may happen only in component C_4 or C_5 according to the respective parameters λ_4 and λ_5. It is desirable to decide a priori what component must be mainly examined to achieve the fixed objective and what is the required amount of work. We can observe for our simple example that component C_4 is the relevant one. The computations confirm this choice since if the residual failure rate λ_5 decreases by a factor of 10 then we only obtain a gain of two hours for the initial lower bound (i.e. $t_{inf} \simeq 90h$). On the other hand, dividing the rate λ_4 by 3 allows to attain $t_{inf} \simeq 257h$ that ensures the fixed requirement. The 0-failure and 2-failures characteristics are both given by Figure 4 for a residual failure rate λ_4 of 0.01 and 0.03.

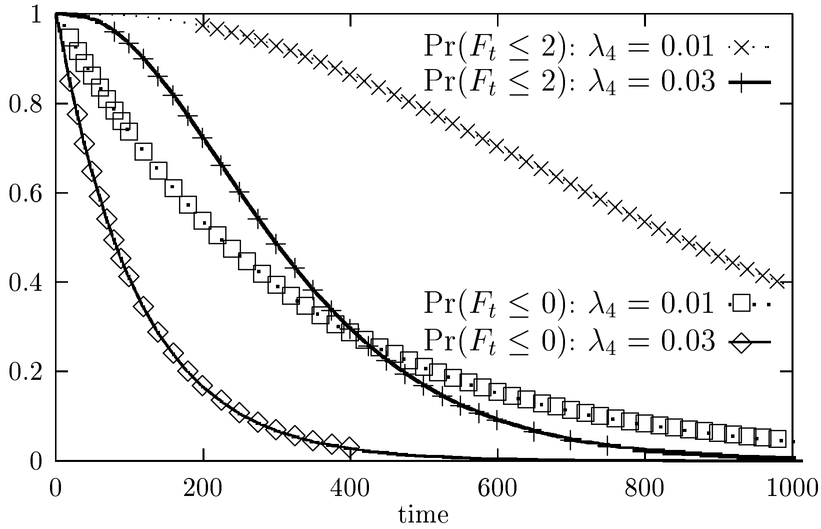


Figure 4: Function $(\mathrm{Pr}(F_t \leq k))_{t \geq 0}$ with 0.01 and 0.03 as value of λ_4 in the model $(Op)_1$.

The perturbation on the distribution of F_t generated by the change of first executed component after occurrence of a failure, betwen the models $(Op)_1$ and $(Op)_2$, may be easily evaluated. It is clear that function $t \to \Pr(F_t \leq k)$ will be all the less perturbated as the number k of considered failures is smaller. In particular, for $k = 0$, the reliability function is identical for the two models since no recovery mechanisms have been invoked over $]0, t]$. For our example, the difference between the values of $\Pr(F_t \leq k)$ for $(Op)_1$ and $(Op)_2$ attains 10^{-2} over the interval $[50h, 250h]$ with $k = 1$, is more than 10^{-2} over $[110h, 450h]$ with $k = 2$ and is close to 2.10^{-2} over $[200h, 450h]$ with $k = 3$. Note that we obtain $t \leq 95h$ as answer to the question raised in Example 3 for model $(Op)_2$. That represents a difference of about 8% on the upper bound time with respect to the first model.

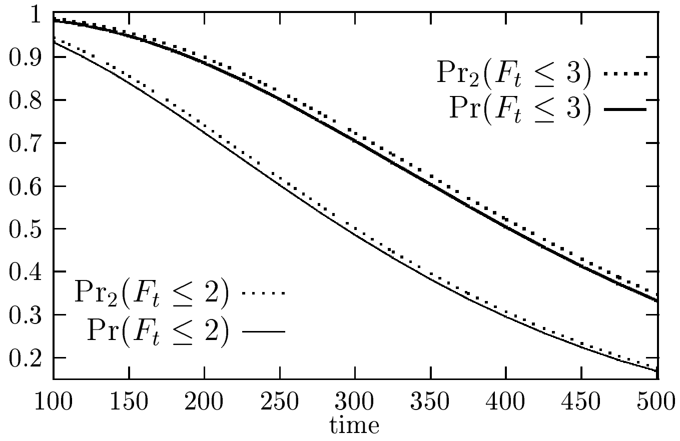


Figure 5: Function $(\Pr(F_t \leq k))_{t \geq 0}$ with $k = 2, 3$ for the models $(Op)_1$ and $(Op)_2$

3.2 Reliability growth

The main purpose of the software reliability models using the black-box view of the system is to take into account the stochastic phenomenon of reliability growth. It takes place when efficient fault removal procedures are used. In a structural approach, this situation has been addressed by Laprie et al. [5]. The basic idea is to choose a black-box model to represent the reliability growth of a component and to use the so-called transformation approach for introduce this single-component model in the Markovian structural one. The failure in a component are assumed to occur according to a hyperexponential non-homogeneous Poisson process. The transformation approach is based on the Markovian interpretation of this black-box model (see [5]). We can also simulate the reliability growth of some components in our model using this technique. The transition graphs associated with models $(Op)_1$ and $(Op)_2$ become those given in the Figure 6, when the reliability of component C_4 grows.

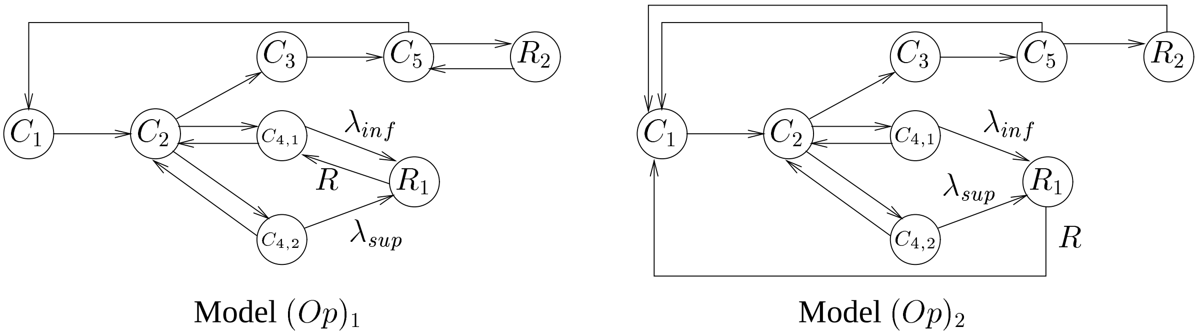


Figure 6: Operational models taking into account the reliability growth of component C_4.

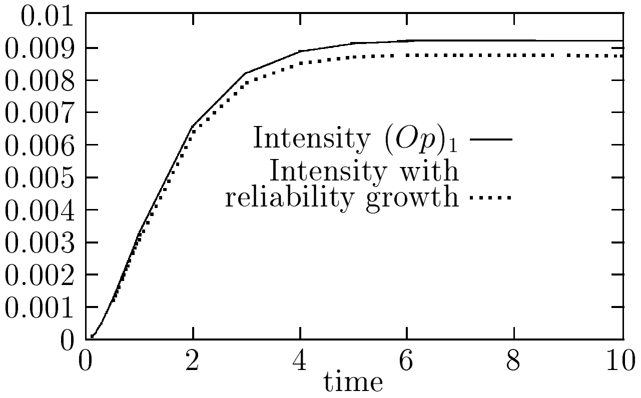


Figure 7: Intensity functions for $(Op)_1$ and $(Op)_1$ with the reliability growth of component C_4.

As a result of the algorithm associated with Result 2.1, we state that the r.v. F_t is s-non-increasing, that is, if $(F_t)_{\text{gr}}$ denotes the number of failures observed over $]0, t]$ taking into account the reliability growth of C_4, $\Pr(F_t \leq k) \leq \Pr((F_t)_{\text{gr}} \leq k)$ for each $k \in \mathbb{N}$.

Let us return to the question "up to what time t the probability to observe at most 2 failures is greater than 0.95?" for the model $(Op)_1$ and for the model derived from the reliability growth of component C_4 with parameters $\lambda_{sup} = 0.0585$ and $\lambda_{inf} = 0.0015$. That gives a growth ratio $\lambda_4/\lambda_{inf} = 20$ and an initial value $h(0) = \lambda_4$. We determined $t \leq 88h$ for $(Op)_1$ and now we get $t \leq 97h$. We see that the gain on the upper bound is about 10%.

3.3 Poissonian approximation

The generator Q of the nominal (Markov) model X was assumed to be irreducible in Subsection 2.1; hence matrix A is also an irreducible sub-generator. It is well-known [14] that a phase-type distribution with an irreducible matrix A is asymptotically exponential with the eigenvalue of maximal real part $-\lambda$ of matrix A as parameter. So, we obtain

$$\Pr(T > t) = e^{-\lambda t} + o(e^{-\lambda t}),$$

with $o(e^{-\lambda t})/e^{-\lambda t} \to 0$ when $t \to +\infty$. The scalar λ may be interpreted as the asymptotic failure rate of the system when t tends to $+\infty$. Indeed, we have

$$\lambda(t) = \sum_{i \in \mathcal{C}} \frac{\alpha e^{At}(i)}{\alpha e^{At}\mathbf{1}^\mathsf{T}} D^\mathsf{T}(i) \underset{t \to +\infty}{\longrightarrow} \lambda(\infty) = \sum_{i \in \mathcal{C}} v(i) D^\mathsf{T}(i)$$

where v is the unique probability distribution satisfying to $vA = \lambda v$. Consequently, we have $\lambda = \lambda(\infty)$. We can note that for all $i \in \mathcal{C}$, $v(i)$ and $\pi(i)/\sum_{i \in \mathcal{C}} \pi(i)$ are distinct quantities (π is the stationary distribution of X^*). Finally, if the distribution of the r.v. T is asymptotically exponential then it does not mean that the counting process of the events is necessarily distributed as a Poisson process with parameter λ.

We have already noted that, when $t \to +\infty$, the failure intensity function becomes the following constant value $h(\infty) = \pi D^\mathsf{T}$ and that, for the Littlewood's model (with $R = S = \mathbf{0}$), the fundamental rate λ_d^* is equal to the parameter of the Poisson distribution given in [6] as approximated distribution for the counting r.v. F_t. In this case, recall that π is also the stationary distribution of the nominal model X. Let us try to compare the exact distribution of F_t for model $(Op)_1$ with the Poisson one with the asymptotic failure rate $\pi(4)D^{(p)}(4,6)+\pi(5)D^{(p)}(5,7) = \pi(4)0.03+\pi(5)0.01$ as parameter. We state that the Poissonian approximation gives a very little under-estimation of the dependability of the system associated with the model $(Op)_1$. Consequently, the computation of the measures with this Poisson distribution ensures that the reliability specifications are met. However, the same conclusion does not hold for the model $(Op)_2$. Indeed, the same type of comparison shows that the Poissonian approximation, first over-estimates the dependability of the software, next under-estimates it. Therefore, we have a priori any order relation between the reliability indicators obtained from the Poisson process and those resulting from the exact initial model.

For a Markov model, we know that the asymptotic probability $\pi(i)$ to be in state i is equal to $\lim_{t\to\infty}(\mathrm{E}[S_i(t)]/t)$. When the observation period becomes large, formulas (4) and (5) provide the following asymptotic expressions of the expected characteristics

$$\lim_{t\to\infty}\frac{\mathrm{E}[F_t]}{t} = \pi D^{\mathsf{T}} = \lambda_d^* \qquad \mathrm{E}[F_t] = \lambda_d^* t + [(\alpha,\mathbf{0})-\pi](\mathbf{1}^{\mathsf{T}}\pi - Q^*)^{-1}D^{\mathsf{T}} + o(1).$$

We can see that if we consider the stationary version of the point process of the failures instants (i.e. the initial distribution $(\alpha,\mathbf{0})$ of X^* is replaced by its stationary distribution π), then we have, for all $t>0$, $\mathrm{E}[F_t] = \lambda_d^* t$ and $h(t) = \lambda_d^*$.

References

[1] M. Xie, *Software Reliability Modelling,* 1991; World Scientific Publishing.

[2] R.C. Cheung, “A user-oriented software reliability model”, *IEEE Trans. Software Engineering*, vol 6, 1980, pp 118–125.

[3] B. Littlewood, “Software reliability model for modular program structure”, *IEEE Trans. Reliability*, vol 28, 1979, pp 241–246.

[4] J.C. Laprie, “Dependability evaluation of software systems in operation”, *IEEE Trans. Software Engineering*, vol 16, 1984, pp 701–714.

[5] J.C. Laprie et al., “The KAT (knowledge-action-transformation) approach to the modeling and evaluation of reliability and availability growth”, *IEEE Trans. Software Engineering*, vol. 17, 1991, pp 370–382.

[6] B. Littlewood, “A reliability model for systems with Markov structure”, *Appl. Statist.*, vol. 24, 1975, pp 172–177.

[7] K. Siegrist, “Reliability of systems with Markov transfer of control”, *IEEE Trans. Software Engineering*, vol 14, 1988, pp 1049–1053.

[8] E. de Souza e Silva and H.R. Gail, “Calculating availability and performability measures of repairable computer systems using randomization”, *Journal of the ACM*, vol 36, 1989, pp 171–193.

[9] M. Kaâniche and K. Kanoun, “The discrete time hyperexponential model for software reliability growth evaluation”, *Proc. Int. Symp. on Software Reliability*, 1992 Oct, pp 64-75.

[10] J.C. Laprie and K. Kanoun, “X-ware reliability and availability modeling”, *IEEE Trans. Software Engineering*, vol 18, 1992, pp 130–147.

[11] S. Gokhale and K.S. Trivedi, “Structure-base software reliability prediction”, *Proc. 5th Int. Conference on Advanced Computing*, 1997, pp 447-452.

[12] B. Littlewood J.F. Meyer and D.R. Wright, “Dependability of modular software in a multiuser operational environment”, *Proc. Int. Symp. on Software Reliability*, 1995, pp 170–179.

[13] B. Beizer, *Software testing techniques*, 1990; Van Nostrand, Reinhold.

[14] M.F. Neuts, *Structured Stochastic Matrices of M/G/1 Type and Their Applications*, 1989; Marcel Dekker Inc.

[15] J. Ledoux and G. Rubino, “Simple formulae for counting processes in reliability models”, *Advances in Applied Probability*, vol 29, 1997, pp 1018–1038.

[16] P. Comer, “Software data collection and the software data library”, *Proc. 6th EUREDATA Conf.*, 1989, pp 824–839.

[17] K.S. Trivedi S. Gokhale, E.W. Wong and J.R. Horgan, “An analytical approach to architecture-based software reliability prediction”, *Proc. Int. Performance and Dependability Symp.*, 1998, pp 13–22.

[18] M.R. Lyu S. Gokhale and K.S. Trivedi, “Reliability simulation of component-based systems”, *Proc. Int. Symp. on Software Reliability*, 1998, pp 192–201.

Author

Dr. James Ledoux; Institut National des Sciences Appliquées; 20 avenue des Buttes de Coësmes; 35043 Rennes Cedex; FRANCE

Internet (e-mail): jledoux@insa-rennes.fr

James Ledoux received a PhD in Computer Science from the University of Rennes in 1993. He is associate professor at the INSA engineering school in Rennes. His research activity focus on reliability models and related stochastic processes.