Software Reliability Modeling

James LEDOUX
Centre de Mathématiques INSA & IRMAR
20 Avenue des Buttes de Coësmes
35043 Rennes Cedex
France

July 27, 2002

1 Introduction

This chapter proposes an overview of some aspects of Software Reliability (SR) engineering. Most systems are now driven by software. So that, it is well-recognized that assessing reliability of software applications is a major issue in reliability engineering, particularly in terms of cost. But predicting software reliability is not easy. Perhaps the major difficulty is that we are concerned primarily with design faults which is a very different situation from that tackled by the conventional hardware theory. A *fault* (or bug) refers to the manifestation in the code of a mistake made by the programmer or designer with respect to the specification of the software. Activation of a fault by an input value leads to an incorrect output. Detection of such an event corresponds to an occurrence of a software *failure*. Input values may be considered as arriving to the software randomly. So although software failure may be not generated stochastically, it may be detected in such a manner. Therefore, this justifies the use of stochastic models of the underlying random process that governs the software failures. We briefly recall in Section 2, basic concepts of stochastic modeling for reliability. Two approaches are used in SR modeling. The prevalent is the so-called *black-box* one, in which only the interactions of the software with

the environment are considered. Following [1] and [2], we use in Section 3 the *self-exciting point processes* as basic tool to model the failure process. That enables an overview of most of published SR models. A second approach, called the *white-box* one, incorporates in models, information on the structure of the software. This is presented in Section 4. Section 5 proposes basic techniques for calibrating black-box models. The last section tries to give an account for the current practices in SR modeling and to point out some challenging issues for future research.

Note that this chapter does not aspire to cover the whole topic of SR engineering. In particular, we do not discuss: fault prevention, fault removal, fault tolerance which are three methods to achieve reliable software. We focus here on methods to forecast failure times. For a more complete view, we refer to [3], the handbooks [4] and [5]. We have used the two recent books [2] and [6] to prepare this chapter. We also recommend to read the short paper [7], that describes, in particular, the available software reliability toolkits (with additional relevant reference [8]). Finally, the bibliography of the chapter gives a good account for journals which propose research and tutorial papers on SR.

2 Basic concepts of stochastic modeling

Reliability of a software is defined in [9] as a measure of the continuous delivery of the correct service by the software under a specified environment. This is a measure of the time to failure.

2.1 Metrics with regard to the first failure

Metrics of the first time to failure of a system are standard from [10], [11] and are now recalled. The first failure time is a random variable (rv) T with distribution function

$$F(t) = \mathbb{P}\{T \leq t\}, \quad t \in \mathbb{R}.$$

If F has a probability density function (pdf) f then we define the *hazard rate* of the rv T by

$$r(t) = \frac{f(t)}{R(t)}, \quad t \geq 0.$$

with $R(t) = 1 - F(t) = \mathbb{P}\{T > t\}$. We will also use the term *failure rate*. Function $R(t)$ is called the *survivor function* of the rv T. Hazard rate function is interpreted to be

$$\begin{aligned} r(t)dt &\approx \mathbb{P}\{t < T \leq t + dt \mid T > t\} \\ &\approx \mathbb{P}\{\text{ a failure occurs in }]t, t + dt] \text{ given that no failure occurred up to time } t \ \} \end{aligned}$$

Thus, the phenomenon of reliability growth ("wear-out") may be represented by an decreasing (increasing) hazard rate.

When F is continuous, hazard rate function characterizes the probability distribution of T through the *exponentiation formula*

$$R(t) = \exp\left(-\int_0^t r(s)ds\right).$$

Finally, the mean time to failure, denoted by MTTF, is the expectation $E[T]$ of the waiting time of the first failure. Note that $E[T]$ is also $\int_0^{+\infty} R(s)ds$.

A basic model for the nonnegative rv T is the Weibull distribution with parameters $\lambda, \beta > 0$:

$$f(t) = \lambda\beta t^{\beta-1} \exp\left(-\lambda t^{\beta}\right) \mathbf{1}_{]0,+\infty[}(t) \quad R(t) = \exp\left(-\lambda t^{\beta}\right), \quad r(t) = \lambda\beta t^{\beta-1},$$

$$\mathrm{MTTF} = \frac{1}{\lambda^{1/\beta}} \int_0^{+\infty} u^{1/\beta} \exp(-u)du.$$

Note that the hazard rate is increasing for $\beta > 1$, decreasing for $\beta < 1$, constant for $\beta = 1$. For $\beta = 1$ we obtain the exponential model with parameter λ.

2.2 Stochastic process of times of failure

The failure process can be thought of as a *point process* (pp), i.e. a sequence of rvs $(T_i)_{i\geq 0}$ where T_i is the ith failure time of the software (with $T_0 = 0$). An equivalent point of view is to

define the sequence of rv $X_i = T_i - T_{i-1}$ for $i \geq 1$. X_i is ith *inter-failure time*. We define the *counting process* $N(\cdot)$ associated with a pp by

$$N(t) = \sum_{i \geq 0} \mathbf{1}_{]0,t]}(T_i) \quad (N(0) = 0).$$

$N(t)$ is the number of observed failures up to time t. A pp will refer to any of (T_i), (X_i) or $N(\cdot)$. Standard metrics associated with a counting process are [11]:

– the mean value function: $M(t) = E[N(t)]$

– the *rate of occurrence of failures* at time t: $\mathrm{ROCOF}(t) = \dfrac{dM}{dt}(t)$.

In such a context, we define the (conditional) *reliability function* at time $t \geq 0$ by

$$R_t(s) = \mathbb{P}\{N(t+s) - N(t) = 0 \mid N(t), T_1, \ldots, T_{N(t)}\}, \quad s \geq 0.$$

This is a measure of the continuous delivery of correct service during the mission interval $]t, t+s]$. At time $t = T_i$, this function is nothing else but the conditional survivor function of rv $X_{i+1} = T_{i+1} - T_i$ given $T_1, \ldots, T_i$. This will be denoted by $R_i(s)$. We also define the (conditional) mean time to failure at time t, $\mathrm{MTTF}(t)$, by

$$\mathrm{MTTF}(t) = \int_0^{+\infty} R_t(s) ds.$$

The mean time to failure at $t = T_i$ will also denoted by MTTF_i and is $E[X_{i+1} \mid T_1, \ldots, T_i]$.

During the operational life of a software, repairs are carried out when it fails to perform correctly. In such a case, time to repair, time to reboot the system and others factors affect the dependability of a product. Thus, we may define the software availability as a measure of the delivery correct service with respect to the alternation correct and incorrect service. Availability is highly dependent on the maintenance policies of the software. We do not go into further

details on dependability in operational phase. Indeed, we focus here on the reliability attribute of the software as most of the literature on software reliability modeling does. We refer to [4, Chap 2] for an account for dependability during the operational phase.

3 Black-box software reliability models

In this section, only *dynamic models* will be discussed. That is, we are only concerned with models which consider failure process as a stochastic process. In other words, time is an essential component of the description of the models. On the other hand, *static models* are essentially capture-recapture models. For a good account for static models, we refer to [12, Chap 5], [6]. A recent evaluation of capture-recapture models in software engineering context is [13]. Our overview of dynamic models closely follows [1, Chap 2], [14], [2], [15]. We assume throughout this section that any corrective action is instantaneous and each detected fault is removed.

A basic way to represent time evolution in confidence in a software is as follows. At instant 0, the first failure occurs at time t_1 according a rv $X_1 = T_1$ with hazard rate r_1. Given time $T_1 = t_1$, we observe a second failure at time t_2 at rate r_2. Function r_2 is the hazard rate of the inter-failure rv $X_2 = T_2 - T_1$ given $T_1 = t_1$. Choice of r_2 is based on the fact that one fault was detected at time t_1. At time t_2, a third failure occurs at t_3 with failure rate r_3. Function r_3 is the hazard rate of the rv $X_3 = T_3 - T_2$ given $T_1 = t_1, T_2 = t_2$ and is selected according to the "past" of the failure process at time t_2: two observed failures at times t_1 and t_2. And so on. It is expected that, due to a fault removal activity, confidence in the software's ability to deliver a proper service will be improved during its lifecycle. Therefore, a basic model in SR has to capture a phenomenon of *reliability growth*. Reliability growth will basically follow

from a sequence of inequalities of the following form

$$r_{i+1}(t - t_i) \leq r_i(t_i) \quad \text{on } t \geq t_i \tag{1}$$

and/or from selection of decreasing hazard rates $r_i(\cdot)$. We illustrate this "modeling process" on the celebrated Jelinski-Moranda model (JM) [16]. We assume a priori that software includes only a finite number N of faults. The first hazard rate is $r_1(t\ ;\ \phi, N) = \phi N$ where ϕ is some nonnegative parameter. From time $T_1 = t_1$, a second failure occurs with the constant failure rate $r_2(t\ ;\ \phi, N) = \phi(N-1), \ldots$ In a more formal setting, the two parameters N and ϕ will be encompassed in what we call a *background history* $\mathcal{F}_0$, which is any background information that we may have about the software. Then "the failure rate" of the software is represented by the function

$$\forall t \geq 0, \quad r_C(t\ ;\ \mathcal{F}_0) = \sum_{i=1}^{+\infty} r_i(t - T_{i-1}\ ;\ \mathcal{F}_0)\mathbf{1}_{[T_{i-1}, T_i[}(t) \tag{2}$$

which is called the *concatenated failure rate function* in [2]. An appealing graphical display of a path of this stochastic function is in Figure 1 for (JM). We can rewrite (2) as

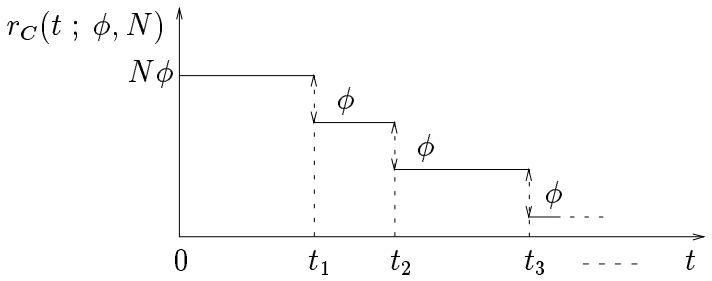

Figure 1: Concatenated failure rate function for (JM)

$$\lambda(t\ ;\ \mathcal{F}_0, N(t), T_1, \ldots, T_{N(t)}) = \phi(N - N(t)) \tag{3}$$

Function $\lambda(\cdot)$ will be called the *stochastic intensity* of the pp $N(\cdot)$. We see that stochastic intensity for (JM) is proportional to the residual number of bugs at any time t and each detection

of failure results in a failure rate whose value decreases of amount ϕ. This suggests that no new fault is inserted during a corrective action and any bug contributes in the same manner to the "failure rate" of the software.

To go further, we replace our intuitive presentation in a stochastic modeling framework. Justification for what follows is that almost all published software reliability models can be interpreted in the foregoing framework. Specifically, this allows a complete overview of the stochastic properties of the panoply of available models without referring to their original presentations.

3.1 Self-Exciting Point Processes

A slightly more formal presentation of the previous construction of the point process (T_i) would be: we have to specify all the conditional distributions

$$\mathcal{L}(X_i \mid \mathcal{F}_0, T_{i-1}, \dots T_1), \quad i \geq 1.$$

The sequence of conditioning $\mathcal{F}_0, \{\mathcal{F}_0, T_1\}, \{\mathcal{F}_0, T_1, T_2\}, \dots$ should be thought of as the natural or internal history on the pp at times $0, T_1, T_2, \dots$ respectively. So that, function r_i is the hazard rate of the conditional distribution $\mathcal{L}(X_i \mid \mathcal{F}_0, T_{i-1}, \dots, T_1)$, that is, when it has a pdf

$$r_i(t\ ;\ \mathcal{F}_0, T_{i-1}, \dots, T_1) = \frac{f_{X_i|\mathcal{F}_0, T_{i-1}, \dots, T_1}(t)}{R_{X_i|\mathcal{F}_0, T_{i-1}, \dots, T_1}(t)}. \tag{4}$$

This leads to the following expression of the stochastic intensity

$$\lambda(t\ ;\ \mathcal{F}_0, N(t), T_1, \dots, T_{N(t)}) = \sum_{i=1}^{+\infty} \frac{f_{X_i|\mathcal{F}_0, T_{i-1}, \dots, T_1}(t - T_{i-1})}{R_{X_i|\mathcal{F}_0, T_{i-1}, \dots, T_1}(t - T_{i-1})}\, \mathbf{1}_{[T_{i-1}, T_i[}(t). \tag{5}$$

If we turn back to the (JM) model, we have $\mathcal{F}_0 = \{\phi, N\}$ and

$$f_{X_i|\mathcal{F}_0, T_{i-1}, \dots, T_1}(t) = \phi(N - (i-1)) \exp\left(-\phi(N - (i-1))\, t\right) \mathbf{1}_{[0, +\infty[}(t).$$

Continuing in this way, this should lead to the martingale approach for analyzing pp, that essentially adheres to the concepts of compensator and stochastic intensity with respect to the internal history of the counting process. In particular, the left continuous version of the stochastic intensity defined in (5) may be thought of as the usual predictable intensity of a pp in the martingale point of view (see e.g. [17, Th11] and references therein). van Pul gives in [18] a good account for what can be done using the so-called *dynamic approach* of pp. We do not go into further details here. We prefer embrace the engineering point of view developed in [15].

It is clear from (5) that the stochastic intensity is excited by the history of the pp itself. Such stochastic processes are usually called a *self-exciting point process* (SEPP). What follows is from [1], [2], [15].

1. $\mathcal{H}_t = \{N(t), T_1, \dots, T_{N(t)}\}$ will denote the internal history of $N(\cdot)$ up to time t.

2. $N(\cdot)$ is said to be conditionally orderly if for any $Q_t \subseteq \mathcal{H}_t$, we have

$$\mathbb{P}\{N(t+dt) - N(t) \geq 2 \mid \mathcal{F}_0, Q_t\} = \mathbb{P}\{N(t+dt) - N(t) = 1 \mid \mathcal{F}_0, Q_t\} O(dt).$$

where $O(dt)$ is some real-valued function such that $\lim_{dt \to 0} O(dt) = 0$.

With $Q_t = \emptyset$ and Formula (6), we get $\mathbb{P}\{N(t+dt) - N(t) \geq 2 \mid \mathcal{F}_0\} = o(dt)$ where $o(dt)$ is some real-valued function such that $\lim_{dt \to 0} o(dt)/dt = 0$. This is the usual orderliness or regular property of a pp [11]. Conditional orderliness is to be interpreted as saying that given Q_t and $\mathcal{F}_0$, as dt decreases to zero, the probability of at least two failures occurring in a time interval of length dt tends to zero at a rate higher than the probability that exactly one failure in the same interval does.

Definition 3.1 *A pp $N(\cdot)$ is called a self-exciting point process if*

1. *$N(\cdot)$ is conditionally orderly;*

2. *there exists a nonnegative function $\lambda(\cdot\ ;\ \mathcal{F}_0, \mathcal{H}_t)$ such that*

$$\mathbb{P}\{N(t+dt) - N(t) = 1 \mid \mathcal{F}_0, \mathcal{H}_t\} = \lambda(t\ ;\ \mathcal{F}_0, \mathcal{H}_t)dt + o(dt) \tag{6}$$

   and $E\big[\lambda(t\ ;\ \mathcal{F}_0, \mathcal{H}_t)\big] < +\infty$ for any $t > 0$

3. $\mathbb{P}\{N(0) = 0 \mid \mathcal{F}_0\} = 1$

Function λ is called the stochastic intensity of the SEPP.

We must think $\lambda(t\ ;\ \mathcal{F}_0, \mathcal{H}_t)$ of as a function of $\mathcal{F}_0$, t and $N(t), T_1, \ldots, T_{N(t)}$. Degree to which $\lambda(t)$ depends on $\mathcal{H}_t$ is formalized in the notion of *memory*. A SEPP is of memory-m, if

– for $m = 0$: $\lambda(\cdot)$ depends on $\mathcal{H}_t$ only through $N(t)$ the number of observed failures at time t;

– for $m = 1$: $\lambda(\cdot)$ depends on $\mathcal{H}_t$ only through $N(t)$ and $T_{N(t)}$;

– for $m \geq 2$: $\lambda(\cdot)$ depends on $\mathcal{H}_t$ only through $N(t), T_{N(t)}, \ldots T_{N(t)-m+1}$;

– for $m = -\infty$: $\lambda(\cdot)$ is independent of the history $\mathcal{H}_t$ of the pp. We also say that stochastic intensity has no memory.

When stochastic intensity depends only on a background history $\mathcal{F}_0$, then we get a *Doubly Stochastic Poisson Process* (DSPP). Thus, the class of SEPP also encompasses the family of Poisson Processes. If intensity is a non-random constant λ, we have the Homogeneous Poisson Process (HPP). A SEPP with no memory and a deterministic intensity function $\lambda(\cdot)$, is a *NonHomogeneous Poisson Process* (NHPP). In particular, if we turn back to the concatenated failure rate function (see (2)), then selecting a NHPP model corresponds to selecting some continuous deterministic function as r_C. We see that, given $T_i = t_i$, the hazard rate of X_{i+1} is

$r_{i+1}(\cdot) = r_C(\cdot + t_i) = \lambda(\cdot + t_i)$. We retrieve a well-known fact for a NHPP: the (conditional) hazard rate between ith and $i+1$th failure times and the intensity function $\lambda(\cdot)$ only differ through the initial time of observation of the two functions. We list properties of SEPP [15] which are of some value for analyzing the main characteristics of models. We will omit to write the dependence in $\mathcal{F}_0$.

Counting statistics for SEPP

The probability distribution of rv $N(t)$ is strongly related to the conditional expectation

$$\widehat{\lambda}(t\ ;\ N(t)) = E\big[\lambda(t\ ;\ \mathcal{H}_t) \mid N(t)\big].$$

This function is called the *count-conditional intensity*. For a SEPP, $\widehat{\lambda}(\cdot\ ;\ N(t))$ satisfies

$$\begin{aligned}\widehat{\lambda}(t\ ;\ N(t)) &= \lim_{dt\to 0} \frac{\mathbb{P}\{N(t+dt) - N(t) = 1 \mid N(t)\}}{dt} \\ &= \lim_{dt\to 0} \frac{\mathbb{P}\{N(t+dt) - N(t) \geq 1 \mid N(t)\}}{dt}.\end{aligned}$$

Then we can obtain the following explicit representation for $\mathbb{P}\{N(t) = n\}$ with $n \geq 1$

$$\mathbb{P}\{N(t) = n\} = \int_{0<t_1<\cdots<t_n<t} \prod_{i=1}^{n} \widehat{\lambda}(t_i\ ;\ i-1) \exp\Big(-\sum_{i=0}^{n} \int_{t_i}^{t_{i+1}} \widehat{\lambda}(u\ ;\ i) du \Big) dt_1 \ldots dt_n \quad (7)$$

with $t_0 = 0$ and $t_{n+1} = t$.

$\mathrm{ROCOF}(t)$, defined as the derivate of $M(t)$, is then

$$\mathrm{ROCOF}(t) = E\left[\widehat{\lambda}(t\ ;\ N(t))\right] = E[\lambda(t\ ;\ \mathcal{H}_t)].$$

We see that notion of $\mathrm{ROCOF}(t)$ and stochastic intensity coincide only if intensity is a deterministic function of time, i.e. the pp is a NHPP.

Likelihood function for a SEPP

Assume that we observe a fixed number i of failures. Then the likelihood function is

$$f_{T_1,\ldots,T_i}(t_1,\ldots,t_i) = \lambda(t_1\ ;\ 0)\prod_{k=2}^{i}\lambda(t_k\ ;\ k-1,t_1,\ldots,t_{k-1}) \exp\left(-\int_0^{t_1}\lambda(s\ ;\ 0)ds - \sum_{k=2}^{i}\int_{t_{k-1}}^{t_k}\lambda(s\ ;\ k-1,t_1,\ldots,t_{k-1})ds\right) \quad (8)$$

If we observe the failure process up to time t, the joint distribution of $N(t), T_1, \ldots, T_{N(t)}$ is given in [15, Th6.2.2].

Reliability and MTTF functions

$$R_t(s) = \exp\left(-\int_t^{t+s}\lambda(u\ ;\ N(t),T_1,\ldots,T_{N(t)})du\right)$$

In particular at instant T_i, we obtain

$$R_i(s) = \begin{cases} \exp\left(-\int_0^s \lambda(u\ ;\ 0)du\right) & \text{if } i=0 \\ \exp\left(-\int_{T_i}^{T_i+s}\lambda(u\ ;\ i,T_1,\ldots,T_i)du\right) & \text{if } i\geq 1. \end{cases}$$

The following characterization of 0-memory SEPP is intuitively clear from the definition of the stochastic intensity of a SEPP.

Theorem 3.2 *$N(\cdot)$ is a SEPP with 0-memory is equivalent to $N(\cdot)$ is a Markov process.*

This result explains why a very large part of SR models may be developed in a Markov framework (see e.g. [12], [3, Chap 10]). In particular, all NHPP models are of Markov-type. In fact, it can be shown [15] that a SEPP with m-memory corresponds to a process (T_i) which is a m-order Markov chain, that is $\mathcal{L}(T_{i+1} \mid T_i,\ldots,T_1) = \mathcal{L}(T_{i+1} \mid T_i,\ldots T_{i-m+1})$ for $i \geq m$.

A last relevant result is concerned with 1-memory SEPP.

Theorem 3.3 *A* 1*-memory SEPP with a stochastic intensity satisfying*

$$\lambda(t\ ;\ N(t), T_{N(t)}) = f(N(t), t - T_{N(t)})$$

for some real-valued function f*, is characterized by a sequence of independent inter-failure durations* (X_i)*. In this case, density probability function of* X_i *is*

$$f_{X_i}(x_i) = f(i-1\ ;\ x_i) \exp\left(- \int_0^{x_i} f(i-1\ ;\ u) du \right). \tag{9}$$

Such a SEPP was called a generalized renewal process in [19] because T_i is the sum of independent but not identically distributed rv. Moreover, it is an usual renewal process when function f does not depend on $N(t)$.

To close this presentation of self-exciting processes, we point out that we only use a "constructive" point of view. Our purpose, here, is not to discuss the existence of point processes with a fixed concatenated failure rate function or stochastic intensity. However, we emphasize that orderliness condition and existence of the limit in (6) in Definition 3.1 are enough to specify the pp (see e.g. [20]). It is also shown in [14, Th4.1] that, under conditional orderliness condition, concatenated failure rate function well-defines a SEPP with respect to operational Definition 3.1. Moreover, an easily checked criterion for conditional orderliness is given [14, Th4.2]. In particular, if hazard rates in (4) are locally bounded then conditional orderliness holds.

3.2 Classification of SR models

We obtain from the concept of memory for a SEPP, a classification of the existing models. It is appealing to define a model with a high memory. But, as usual, the pay-off is the complexity in the statistical inference and the amount of data to be collected.

3.2.1 0-memory SEPP

A first type of 0-memory SEPP is when $\lambda(t\ ;\ \mathcal{H}_t, \mathcal{F}_0) = f(N(t), \mathcal{F}_0)$. We get major common properties of this first class of models from Subsection 3.1.

- $N(\cdot)$ is a Markov process (a pure birth Markov process).
- (X_i) are independent (given $\mathcal{F}_0$). From (9), rv X_i has an exponential distribution with parameter $f(i-1, \mathcal{F}_0)$. This easily gives a likelihood function given inter-failure durations (X_i).
- $T_i = \sum_{k=1}^{i} X_k$ is an Hypoexponential distributed rv as the sum of i independent and exponentially distributed rvs [21].
- $R_t(s) = \exp\left(-f(N(t), \mathcal{F}_0)s\right)$. The reliability function only depends on the current number of failures $N(t)$. We have $\mathrm{MTTF}(t) = 1/f(N(t), \mathcal{F}_0)$.

Example 3.1 (JM) Jelinski-Moranda model has been introduced in Section 3. This model has to be considered as a benchmark model, since all authors designing a new model emphasize that their model includes JM as particular case. The stochastic intensity is given in (3) (see Figure 1 for a path). Besides properties common to the class of SEPP considered in this paragraph, we had the following additional assumptions: the software includes a finite N of bugs in the program and no new fault is inserted during debugging. We also noted that each fault has the same contribution to the un-reliability of the software. These assumptions are generally considered as questionable. We derive from (7) that the distribution of rv $N(t)$ is Binomial with parameters N and $1-\exp(-\phi t)$. The main reliability metrics are:

$$\mathrm{ROCOF}(t) = N\phi\exp(-\phi t), \mathrm{MTTF}_i = \frac{1}{N - i\,\phi}, R_i(s) = \exp\big(-(N - i\,\phi)s\big). \qquad (10)$$

We also mention the Geometric model of Moranda [22] where the stochastic intensity is $\lambda(t\ ;\ \mathcal{H}_t, \lambda, c) = \lambda c^{N(t)}$ where $\lambda \geq 0$ and $c \in]0, 1[$.

A second class of 0-memory SEPP is when the stochastic intensity is actually a function of time t, $N(t)$ (and $\mathcal{F}_0$). In fact we only have in this category of models, SEPP with $\lambda(t\ ;\ \mathcal{H}_t, \mathcal{F}_0) = (N - N(t))\varphi(t)$ for some deterministic function $\varphi(\cdot)$ of time t. Note that $\varphi(t) = \phi$ gives (JM).

– $N(\cdot)$ is a Markov process.

– Let us denote $\int_0^t \varphi(s)ds$ by $\Phi(t)$. We deduce from (7) that $N(t)$ is a Binomial distributed rv with parameters N and $p(t) = 1 - \exp(-\Phi(t))$. This leads to the term Binomial-type model in [23]. It follows that $E[N(t)] = Np(t)$ and $\mathrm{ROCOF}(t) = N\varphi(t)\exp(-\Phi(t))$.

– $R_t(s) = \exp\big(-(N - N(t))(\Phi(s + t) - \Phi(t))\big)$.

– We get the likelihood function given failure times (T_i) from (8)

$$f_{T_1,\dots,T_i}(t_1, \dots, t_i) = \frac{N!}{(N-i)!} \exp\big(-(N-i)\Phi(t_i)\big) \prod_{j=1}^{i} \varphi(t_j) \exp\left(-\Phi(t_j)\right)$$

– The pdf of rv T_i is $f_{T_i}(t_i) = i \binom{N}{i} \exp(-\Phi(t_i))\varphi(t_i)[\exp\left(-\Phi(t_i)\right) - 1]^{i-1}$.

Littlewood's model in [24] is an instance of a Binomial model with $\varphi(t) = \alpha/(\beta + t)$ and $\alpha, \beta > 0$.

3.2.2 NHPP model: $\lambda(t\ ;\ \mathcal{H}_t, \mathcal{F}_0) = f(t; \mathcal{F}_0)$ and is deterministic

A large part of models is of NHPP-type. We refer to [25] and [6, Chap 5] for a complete list of such models. The very appealing properties of the counting process explains the widely use of this family of pp in SR modeling.

– $N(\cdot)$ is a Markov process.

– We deduce from Definition 3.1 that for any t, $N(t)$ is a Poisson distributed rv with parameter $\Lambda(t) = \int_0^t f(s; \mathcal{F}_0)ds$. In fact, $N(\cdot)$ has independent (but nonstationary) increments, that is, $N(t_1), N(t_2) - N(t_1), \ldots, N(t_{i-1}) - N(t_i)$ are independent rvs for any $(t_1, \ldots, t_i)$.

– The mean value function $M(t)$ is $\Lambda(t)$. Then $\mathrm{ROCOF}(t) = f(t; \mathcal{F}_0)$. Such a model is called a finite NHPP model if $M(+\infty) < +\infty$ and an infinite one when $M(+\infty) = +\infty$. Indeed, if $M(+\infty) < +\infty$ then we only consider a finite number of failure times with probability 1.

– $R_t(s) = \exp\big(-(\Lambda(t+s) - \Lambda(t))\big)$.

– Likelihood function given failure times (T_i) is from (8)

$$f_{T_1,\ldots,T_i}(t_1, \ldots, t_i) = \exp(-\Lambda(t_i)) \prod_{k=1}^{i} f(t_k, \mathcal{F}_0). \tag{11}$$

Example 3.2 (GO) This model is characterized by the following mean value function and intensity function: $\Lambda(t) = M(1 - \exp(-\phi t))$ and $\lambda(t\ ;\ \phi, M) = M\phi\exp(-\phi t)$ for $t \geq 0$. Parameters ϕ and M are the failure rate per fault and the finite expected (initial) number of faults contained in the software. We see that, at any time t, the intensity function is proportional to the expected remaining number of faults $\lambda(t\ ;\ \phi, M) = \phi(M - \Lambda(t))$. Thus, (GO) is essentially a NHPP-version of (JM).

Example 3.3 (MO) For the Musa-Okumoto model [3], mean value function and intensity function are $\Lambda(t) = \ln(\lambda\theta t + 1)/\theta$ and $\lambda(t\ ;\ \theta, \lambda) = \lambda/(\lambda\theta t + 1)$ respectively. λ is the initial value of the intensity function and θ is called the failure intensity decay parameter. It is easily seen that $\lambda(t\ ;\ \theta, \lambda) = \lambda\exp(-\theta\Lambda(t))$. Thus, intensity function exponentially decreases with the

expected number of failures and shows that (MO) may be understood as a NHPP version of Moranda's Geometric model.

As quoted in [26], using a NHPP model may appear inappropriate to describe reliability growth of a software. Indeed, this is debugging which modifies the reliability. Thus, the true intensity function changes probably in a discontinuous manner during corrective actions. However, Miller showed in [27] that a binomial-type model of Subsection 3.2.1 may be transformed in a NHPP variant assuming that the initial number of faults is a Poisson distributed rv with expectation N. For instance, (JM) is transformed into (GO). Moreover, Miller showed that binomial-type model and its NHPP variant are indistinguishable from a single realization of the failure process. But, these two models differ as prediction model because estimates of parameters are different.

3.2.3 1-memory SEPP with $\lambda(t\ ;\ \mathcal{H}_t, \mathcal{F}_0) = f(N(t), t - T_{N(t)}, \mathcal{F}_0)$

$N(\cdot)$ is not Markovian. But the inter-failure durations (X_i) are independent rvs given F_0 (see Theorem 3.3) and the pdf of rv X_i is given in (9).

Example 3.4 (LV) Stochastic intensity of the Littlewood-Verrall model [28] is

$$\lambda(t\ ;\ \mathcal{H}_t, \alpha, \psi(\cdot)) = \frac{\alpha}{\psi(N(t)+1) + t - T_{N(t)}} \tag{12}$$

for some nonnegative function $\psi(\cdot)$. We briefly recall the Bayesian rationale underlying the definition of this model. Uncertainty about the debugging operation is represented by a sequence of stochastically decreasing failure rates $(\Lambda_i)_{i\geq 1}$, that is

$$\Lambda_j \leq_{st} \Lambda_{j-1}, \quad \text{i.e } \big(\forall t \in \mathbb{R} : \mathbb{P}\{\Lambda_j \leq t\} \geq \mathbb{P}\{\Lambda_{j-1} \leq t\}\big). \tag{13}$$

Thus, using stochastic order allows a decay of reliability which takes place when fault are inserted. Prior distribution of rv Λ_i is Gamma with parameters α et $\psi(i)$. It can be shown that

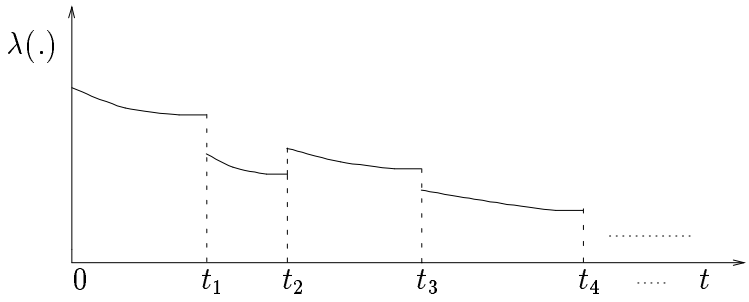

Figure 2: A path of the stochastic intensity for (LV)

inequality (13) holds when $\psi(\cdot)$ is a monotonic increasing function of i. Given $\Lambda_i = \lambda_i$, rv X_i has an exponential distribution with parameter λ_i. Unconditional distribution of rv X_i is a Pareto distribution with pdf

$$f(x_i \ ; \ \alpha, \psi(i)) = \frac{\alpha \psi(i)^\alpha}{(x_i + \psi(i))^{\alpha+1}}.$$

Thus, the "true" hazard rate of X_i is $r_i(t \ ; \ \alpha, \psi(i)) = \alpha/(t + \psi(i))$. In [29], parameters α and ψ are estimated using Bayesian method. The corresponding model is called a Hierarchical Bayesian model in [2] and is also a 1-memory SEPP. If $\psi(\cdot)$ is linear in i, we have

$$R_i(t) = \left(\frac{\psi(i+1)}{t + \psi(i+1)} \right)^\alpha, \quad \text{MTTF}_i = \frac{\psi(i+1)}{\alpha - 1}.$$

We also mention the Schick-Wolverton's model [30] where $\lambda(t \ ; \ \mathcal{H}_t, \Phi, N) = \phi(N - N(t))(t - T_{N(t)})$ and $\phi > 0, N$ are the same parameters as for (JM).

3.2.4 $\mathbf{m \geq 2}$-memory

Instances of such models are rare. For $m = 2$, we have the time-series models in [31], [32]. For $m > 2$, we have the adaptative concatenated failure rate model from [33] and the Weibull models of Pham [34]. We refer to original contributions for details.

4 White-box modeling

Most work on software reliability assessment adopts the black-box view of the system, in which only the interactions with the environment are considered. 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 [35], [36]. Specifically, the structure-based approach allows analyzing the sensitivity of the reliability of the system with respect to the reliability of its components. Up to recently, only a few papers proposed structure-based software reliability models. A representative sample was (in discrete time) [35], [37], [38] and (in continuous time) [39],[36], [19],[40]. An up-to-date review on the architecture-based approach is given in [41]. We will present the main features of the basic Littlewood's model which are common to most previous cited works. The discrete time counterpart is Cheung's model.

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 [35]. The software structure is then represented by the *call graph* of the set $\mathcal{M}$ of the modules. These modules interact by execution control transfer and, at 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 continuous time stochastic process $(X_t)_{t\geq 0}$ which indicates the active module at each time t. $(X_t)_{t\geq 0}$ is assumed to be a homogeneous Markov process on the set $\mathcal{M}$.

In a second step, Littlewood describes the failure processes associated with execution actions. 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 λ_i. When control is transferred from module $i \in \mathcal{M}$ to module $j \in \mathcal{M}$, a failure may happen with probability $\mu(i,j)$. Given a sequence of executed

modules, the failure processes associated with each state are independent. Also, the interface failure events are independent on each other and on the failure processes occurring when a module is active.

The architecture of the software is combined with the failure behavior of the modules and that of interfaces into a single model which can then be analyzed. This method is referred as the "composite-method" according to the classification of Markov models in the white-box approach proposed in [41]. Basically, we are still interested in the counting process $N(\cdot)$.

Another interesting point process is obtained by assuming that the probability of a secondary failure during a control transfer is 0. Thus, assuming that $\mu(i,j) = 0$ for all $i,j \in \mathcal{M}$ in the previous context, we get a Poisson process whose parameter is modulated by the Markov process $(X_t)_{t\geq 0}$. This is also called a Markov Modulated Poisson Process (MMPP). It is well-known (e.g. [17]) that the stochastic intensity of a MMPP with respect to the history $\mathcal{H}_t \vee \mathcal{F}_0$, where $\mathcal{F}_0 = \sigma(X_s, s \geq 0)$, is λ_{X_t}. Thus, $N(\cdot)$ is an instance of a DSPP.

Asymptotic analysis and transient assessment of distribution of rv $N(t)$ may be carried out in observing that the bivariate process $(X_t, N(t))_{t\geq 0}$ is a jump Markov process with state space $\mathcal{M} \times \mathbb{N}$. Computation of all standard reliability metrics may be performed as in [42],[43]. We refer to these papers for details and for calibration of the models.

It can be argued that models of Littlewood-type are inappropriate to capture a reliability growth of the software. In fact, Laprie et al. [19] has developed a method to incorporate such a phenomenon which can be used, for instance, in the Littlewood's model to take into account reliability growth of modules in the assessment of the overall reliability (see [43]).

In the original papers about Littlewood's model for modular software, it is claimed that if

failure parameters decrease to 0 then $N(\cdot)$ is asymptotically a HPP with parameter

$$\lambda = \sum_{i \in \mathcal{M}} \pi(i) \big[\sum_{j \in \mathcal{M}} Q(i,j)\mu(i,j) + \lambda_i \big],$$

where Q is the irreducible generator of the Markov process $(X_t)_{t \geq 0}$ and π its stationary distribution. $\pi(i)$ is to be interpreted as the proportion of time the software passed in module i (over a long time period). We just discuss the case of a MMPP. Convergence to 0 of the failure parameters λ_i ($i \in \mathcal{M}$) may be achieved in multiplying each of them by a positive scalar ε, and in considering that ε decreases to 0. So that, the new stochastic intensity of the MMPP is $\varepsilon \lambda_{X_t}$. As ε tends to 0, it is easily seen from a MMPP with a modulating two-states Markov process $(X_t)_{t \geq 0}$ that, for the first failure time T, probability $\mathbb{P}\{T > t\}$ converges to 1. Therefore, we can not obtain an exponential approximation to the distribution of rv T as ε tends to 0. Thus, we can not expect to derive a Poisson approximation to the distribution of $N(\cdot)$. In fact, the right statement is: if failure parameters are much smaller than the switching rates between modules, $N(\cdot)$ is approximately a HPP with parameter λ. For a MMPP, a proof is given in [44] using martingale theory. Moreover, the rate of convergence in total variation of finite dimensional-distributions of $N(\cdot)$ to those of the HPP is shown to be in ε. This last fact is important because user has no information on the quality of the Poissonian approximation given in [36]. However, there is no rule to decide a priori if the approximation is optimistic or pessimistic (see [43]). A similar approach to [44] is used in [45] to derive Poisson approximation and rate of convergence for more general pp than MMPP, including the complete Littlewood's counting process [46], the counting model of [43]. Such asymptotic results give a "hierarchical-method" [41] 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.

5 Calibration of model

Suppose that we have selected one of the black-box models of Section 3. We obtain reliability metrics which depend on the unknown parameters of the model. Thus, we hate to estimate these metrics from the failure data. We briefly review standard methods to get point-estimates in Subsections 5.1, 5.2.

One major goal of the SR modeling is to predict the future value of metrics from the gathered failure data. It is clear from Section 3 that a central problem in SR is to select a model because the huge number of available models. Criteria to compare SR models are listed in [3]. The authors propose quality of assumptions, applicability, simplicity and predictive validity. To assess the predictive validity of models, we need methods which are not only based on a goodness-of-fit approach. Various techniques may be used: u-plot, prequential likelihood, etc. We do not discuss this issue here. A good account for predictive validation methods are given in [47], [48], [4, Chap 4], [3], [11] where comparisons between models are also carried out. Note also that predictive quality of a SR model may be drastically improved using preprocessing of data. In particular, statistical tests have been designed to capture trend in data. Thus, reliability trend analysis allows using SR models which are adapted to reliability growth, stable reliability and reliability decrease respectively. We refer to [49], [4, Chap 10] and references therein for details. Parameters will be denoted by θ (it can be multivariate).

5.1 Frequentist procedures

Parameter θ is considered as taking an unknown but fixed value. Two basic methods to estimate the value of θ are: method of maximum likelihood (ML), the least squares method. That is, we have to optimize with respect to θ an objective function which depends on θ and collected

failure data to get a *point-estimate*. Another standard procedure, *interval estimation*, gives an interval of values as estimate of θ. We only present point-estimation by method of maximum likelihood on (JM) and NHPP models. Others models may be analyzed in a similar way. ML estimations possess several appealing properties that make the procedure widely used. Two of these properties are the consistency and the asymptotic normality to get *confidence interval* for the point-estimate. Another one is that the ML estimates of $f(\theta)$ (for one-to-one function f) is simply $f(\widehat{\theta})$ where $\widehat{\theta}$ is the ML estimate of θ. We refer to [3, Ch 12] for a complete view of the frequentist inference procedures in SR modeling context.

Example 5.1 (JM) Parameters are ϕ and N. Assume that failure data are given by observed values $\underline{x} = (x_1, \ldots, x_i)$ of rv's $X_1, \ldots, X_i$. If (ϕ, N) are the true values of parameters, then the likelihood to observe $\underline{x}$ is defined as

$$L(\phi, N \,;\, \underline{x}) = f_{X_1,\ldots,X_i}(\underline{x} \,;\, \phi, N) \tag{14}$$

where $f_{X_1,\ldots,X_i}(\cdot \,;\, \phi, N)$ is the pdf of the joint distribution of $X_1, \ldots, X_i$. The estimate $(\widehat{\phi}, \widehat{N})$ of (ϕ, N) will be the value of (ϕ, N) which maximizes the likelihood to observe data $\underline{x}$: $f_{X_1,\ldots,X_i}(\underline{x} \,;\, \widehat{\phi}, \widehat{N}) = \max_{\phi,N} L(\phi, N \,;\, \underline{x})$. Maximizing the likelihood function is equivalent to maximizing the log-likelihood function $\ln L(\phi, N \,;\, \underline{x})$. From the independence of rv's (X_i) and (14), we obtain that

$$\ln L(\phi, N \,;\, \underline{x}) = \ln \prod_{k=1}^{i} \phi(N-(k-1)) \exp\big(-\phi(N-(k-1))x_k\big).$$

Estimates $(\widehat{\phi}, \widehat{N})$ are solution of $\frac{\partial}{\partial \phi} \ln L(\phi, N \,;\, \underline{x}) = \frac{\partial}{\partial N} \ln L(\phi, N \,;\, \underline{x}) = 0$:

$$\widehat{\phi} = \frac{i}{\widehat{N} \sum_{k=1}^{i} x_k - \sum_{k=1}^{i} (k-1)x_k} \quad \sum_{k=1}^{i} \frac{1}{\widehat{N} - (k-1)} = \frac{i}{\widehat{N} - \frac{1}{\sum_{k=1}^{i} x_k} \sum_{k=1}^{i} (k-1)x_k}$$

The second equation may be solved by numerical techniques and then the solution is put into the first equation to get $\widehat{\phi}$. These estimates are plugged into formulae (10) to get ML estimates of reliability metrics.

Example 5.2 (NHPP models) Assume that failure data are $\underline{t} = (t_1, \ldots, t_i)$ the observed failure times. The likelihood function is given by (11). Thus, for (MO) model, ML estimates of parameters β_0, β_1 (where $\beta_0 = 1/\theta, \beta_1 = \lambda\theta$) are solution of [3]

$$\widehat{\beta}_0 = \frac{i}{\ln(1+\widehat{\beta}_1 t_i)}, \quad \frac{1}{\widehat{\beta}_1}\sum_{k=1}^{i}\frac{1}{1+\widehat{\beta}_1 t_k} = \frac{i\, t_i}{(1+\widehat{\beta}_1 t_i)\ln(1+\widehat{\beta}_1 t_i)}.$$

Assume now that failure data are the cumulative numbers of failures $n_1, \ldots, n_i$ at some instants $d_1, \ldots, d_i$. The likelihood function is from the independence and Poisson distribution of the increments of the counting process $N(\cdot)$

$$L(\theta\ ;\ \underline{t}) = \prod_{k=1}^{i}\frac{(\Lambda_\theta(d_k) - \Lambda_\theta(d_{k-1}))^{n_k - n_{k-1}}}{(n_k - n_{k-1})!}\exp\big(-(\Lambda_\theta(d_k) - \Lambda_\theta(d_{k-1}))\big) \quad (d_0 = n_0 = 0),$$

where $\Lambda_\theta(\cdot)$ is the mean value function of $N(\cdot)$ given that θ is the true value of the parameter. For (GO) model, parameters are M, ϕ and their ML estimates are

$$\widehat{M} = \frac{n_i}{1-\exp(-\phi d_i)} \quad \frac{d_i\exp(-\phi d_i)n_i}{1-\exp(-\phi d_i)} = \sum_{k=1}^{i}\frac{(n_k - n_{k-1})(d_k\exp(-\phi d_k) - d_{k-1}\exp(-\phi d_{k-1}))}{\exp(-\phi d_{k-1}) - \exp(-\phi d_k)}.$$

The second equation is solved by numerical techniques and the solution is incorporated into the first equation to get $\widehat{M}$. Estimation of reliability metrics are obtained as for (JM).

5.2 Bayesian procedure

An alternative to frequentist procedures is to use Bayesian statistics. Parameter θ is considered as a value of a rv Θ. A *prior distribution* of Θ has to be selected. This distribution represents the a priori knowledge on the parameter and is assumed to have a pdf $\pi_\Theta(\cdot)$. Now, from the

failure data $\underline{x}$, we have to update our knowledge on Θ. If $L(\theta\ ;\ \underline{x})$ is the likelihood function of the data, then Bayes theorem is used as updating formula: pdf of the *posterior distribution* of Θ given data $\underline{x}$ is

$$f_{\Theta\,|\underline{x}}(\theta) = \frac{L(\theta\ ;\ \underline{x})\ \pi_\Theta(\theta)}{\int_{\mathbb{R}} L(\theta\ ;\ \underline{x})\pi_\Theta(\theta)d\theta}.$$

Now, we find an estimate $\widehat{\theta}$ of θ by minimizing the so-called *posterior expected loss*: $\int_{\mathbb{R}} l(\widehat{\theta},\theta)\, f_{\Theta\,|\underline{x}}(\theta)d\theta$ where $l(\cdot,\cdot)$ is the loss function. With a quadratic loss function $l(\widehat{\theta},\theta) = (\widehat{\theta}-\theta)^2$, it is well-known that the minimum is obtained by the conditional expectation

$$\widehat{\theta} = E[\Theta \mid \underline{x}] = \int_{\mathbb{R}} \theta f_{\Theta\,|\underline{x}}(\theta)d\theta.$$

It is just the mean of the posterior distribution of Θ. Note that all SR models involve two or more parameters, so that the previous integral must be considered as multidimensional. Therefore computation of such integrals are by numerical techniques. Since a decade, progress has been made on such methods: e.g. Monte Carlo Markov Chain methods (see e.g. [50]).

Note that prior distribution can also be parametric. In general, Gamma or Beta distributions are used. So that, additional parameters need to be estimated. This may be carried out by using the ML method. For instance, this is the case for function $\psi(\cdot)$ in the (LV) model in [28]. However, such estimates may be derived in the Bayesian framework and we obtain a hierarchical Bayesian analysis of the model. Many models of Section 3 have been analyzed from a Bayesian point of view using MCMC methods like Gibbs sampling, or data augmentation method (see [51] for (JM), (LV); [52] for NHPP model, [53] for S-shaped models and reference therein).

6 Current issues

6.1 Black-box modeling

We list some issues which are not new but covering well-documented limitations of popular black-box models. Most of them have been recently addressed and practical validation is needed. We will see that the prevalent approach is to use the NHPP modeling framework. Indeed, an easy way to incorporate in a NHPP model various factors affecting the reliability of a software, is to select a suitable parameterization of the intensity function (or ROCOF). Any alternative model combining most these factors will be of value.

6.1.1 Imperfect debugging

The problem of imperfect debugging may be naturally addressed in the Bayesian framework. Reliability growth is captured through deterministically nonincreasing sequence of failure rates $(r_i(\cdot))$ (see (1)). In Bayesian framework, parameters of $r_i(\cdot)$ are considered as random. So that, we can deal with stochastically decreasing sequence of rvs $(r_i)_i$ (see (13)), which allows to take into account the uncertainty on the effect of a corrective action. An instance of this approach is given by the (LV) model (see also [29], [34]).

Note that the binomial class of models can incorporate a defective correction of a detected bug. Indeed, assume that each fault detected has a probability p to be removed from the software. The hazard rate after $(i-1)$ repairs is $\phi(N - p(i-1))$ (see [23]). But, the problem of eventual introduction of new faults is not addressed. Kremer [54] solves the case of a single insertion using a nonhomogeneous birth-death Markov model. This has been extended to multiple introductions in [55]. Shanthikumar and Sumita [56] proposed a multivariate model where multiple removing and insertions of faults are allowed at each repair. This model involved complex

computational procedures and is not considered in literature. Recent advance in addressing the problem of eventual insertion of new faults is concerned with finite NHPP models. It consists in generalizing the basic proportionality between the intensity function and the expected number of remaining faults at time t of (GO) model (see Example 3.2) in

$$\lambda(t) = \phi(t)[n(t) - \Lambda(t)]$$

where $\phi(t)$ represents a time-dependent detection-rate of a fault; $n(t)$ is the number of faults in the software at time t, including those already detected and removed and those inserted during the debugging process ($\phi(t) = \phi$ and $n(t) = M$ in (GO)). Making $\phi(\cdot)$ time-dependent allows representing a phenomenon of learning process which is closely related to the changes in the efficiency of testing. This function can monotonically increase during testing period. Select a nondecreasing S-shaped curve as $\phi(\cdot)$ gives an usual S-shaped NHPP model. Further details may be found in [6, Chap 5] and references therein.

Another basic way to consider insertion of new faults is to use a marked point process (MPP) (e.g. [15, Chap 4]). We have the pp of failure detection times $T_1 < T_2 < \cdots$ and with each date T_i, we associate a mark M_i which represents the cumulative number of faults removed and inserted during the debugging phase. We retrieve a usual pp if all marks are 1. For such a model, we are interested in the mark-accumulator process $\sum_{i=0}^{N(t)} M_i$ ($M_0 = 0$). A basic instance of MPP is the compound Poisson Process where $N(\cdot)$ is a HPP and M_i's are i.i.d. and independent of $N(\cdot)$. This may also be used to model clustering of failures (see [57]). Such MPP are not SEPP of Section 3.1 because orderliness condition fails. This framework was used by van Pul [18] to extend models of (JM) type to incorporate the possibility of inserting new faults during repair.

6.1.2 Early prediction of software reliability

A major limitation of SR models of Section 3 for software engineering community is to provide no help for managing in the earlier phase of development (or testing) of a product. Indeed, calibration of these black-box models requires a relatively large set of failure data. This is rarely encountered in the earlier life-cycle of a software. In some sense, we are now concerned with the general topic of software quality assessment (which includes dependability concepts) with no failure data. Thus, we have to develop statistical models of quality which are not directly related to the knowledge of a part of the failure process. In such a case, model must be based on a priori information on the product: judgment of experts, quality of the development process, similar existing products, software complexity, etc. Incorporating subjective information leads naturally to Bayesian statistics. We refer to [2, Chap 5,6] for discussion in this context. Since we are mainly interested in reliability assessment, we restrict ourselves to more and less recent issues relying quality control to the software reliability.

A widespread idea is that complexity of a software is an influent factor of the reliability attributes. Much work has been devoted to quantify the software complexity through *software metrics* (see e.g. [58]). Typically, we compute Halstead and McCabe metrics which are program size and control flow measures respectively. It is worthy of note that most software complexity metrics are strongly related to the concept of structure of software code. Thus, including complexity factor in SR may be thought of as a first attempt to take into account the architecture of a software in reliability assessment. We turn back to this issue in Subsection 6.2. Now, how to include complexity attributes in earlier reliability analysis? Most of recent research focus on the identification of software modules which are likely fault-prone from data of various complexity metrics. In fact, we are faced with a typical problem of data analysis that explains why

literature on this subject is mainly concerned with procedures of multivariate analysis: linear and nonlinear regression methods, classification methods, techniques of discriminant analysis. We refer to [4, Chap 12], [59],[60] and references therein for details.

Another empirical evidence suggests that the higher the test coverage, the higher the reliability of the software would be. Thus, a model which incorporates information on functional testing as soon as it is available is of value. This issue is addressed in a NHPP model proposed in [61]. It consists in defining an appropriate parameterization of a finite NHPP model which relates software reliability to the measurements that can be obtained from the code during functional testing. Let a be the expected number of faults that would be detected given infinite time testing. The intensity function $\lambda(\cdot)$ is assumed to be proportional to the expected number of remaining failures: $\lambda(t) = [a - \Lambda(t)]\phi(t)$ where $\phi(t)$ is the hazard rate per-fault. Finally, the time-dependent function $\phi(t)$ is of the form

$$\phi(t) = \frac{\frac{dc(t)}{dt}}{1 - c(t)}$$

where $c(t)$ is the coverage function. That is, the ratio of the number of potential fault-sites covered by time t divided by the total number of potential fault-sites under consideration during testing. Function $c(t)$ is assumed to be continuous and monotone as function of time testing. Specific forms of function $c(\cdot)$ allow retrieving some well-known finite failure models: exponential function $c(t) = 1 - \exp(-\phi t)$ corresponds to the (GO); Weibull coverage function $c(t) = 1 - \exp\left(-\phi t^{\gamma}\right)$ corresponds to the generalized (GO) model [62]; S-shaped coverage function correspond to S-shaped models [25], etc. Gokhale et al propose to use a log-logistic function. We refer to [61] for details. Such a parameterization leads to estimate a and the parameters of function $c(\cdot)$. The model may be calibrated according to the different phase of the software life-cycle. Here, in early phase of testing, an approach is to estimate a from software

metrics (using procedures of multivariate analysis) and measure coverage during the functional testing using a coverage measurement tool (see e.g. [4, Chap 13]). Thus, we get early prediction of reliability (see [63] for an alternative using information from testing phases of similar past projects).

6.1.3 Environmental factors

Most SR models in Section 3 ignore the factors affecting software reliability. In some sense, previously issues discussed in this section can be considered as an attempt to capture some environmental factors. Imperfect debugging is related to the fact that new faults may be inserted during a repair. Complexity attributes of a software is strongly correlated to its fault-proness. Empirical investigations show that the development process, testing procedure, programmer skill, human factors, the operational profile and many others factors affect the reliability of a product (see e.g. [64], [4, Chap 13], [65], [66] and references therein). A major issue is to incorporate all these attributes into a single model. At the present time, investigation focus on functional relationship between the hazard rate $r_i(\cdot)$ of the software and quantitative measures of the various factors. In this context, a well-known model is the so-called Cox proportional hazard model (PHM) where $r_i(\cdot)$ is assumed to be an exponential function of the environmental factors:

$$r_i(t) = r(t) \exp\left(\sum_{j=1}^{n} \beta_j z_j(i)\right) \tag{15}$$

where $z_j(\cdot)$'s, called the explanatory variables or *covariates*, are the measures of the factors and β_j's are the regression coefficients. $r(\cdot)$ is a baseline hazard rate that gives the hazard rate when all covariates are set to 0. Therefore, given $\underline{z} = (z_1, \ldots, z_n)$, the reliability function R_i is

$$R_i(t \mid \underline{z}) = R(t) \exp\left(\sum_{j=1}^{n} \beta_j z_j(i)\right) \tag{16}$$

with $R(t) = \exp\left(-\int_0^t r(s)ds\right)$. Formula (15) expresses the effect of accelerating or decelerating the time to failure given $\underline{z}$. Note that covariates may be time-dependent, random. In this last case the reliability function will be the expectation of function in (16). Baseline hazard rate may be any of the hazard rates used in Section 3. The family of parameters can be estimated using ML. Note that one of the reasons for the popularity of PHM is that the unknown β_j's may be estimated by the partial likelihood approach without putting a parametric structure on the baseline hazard rate. We refer to [67], [68], [11], [69] for general discussion on Cox regression models. Applications of PHM to software reliability modeling are given in [12, Chap 7], [70], [71] and references therein. Recently, Pham derives in [72] an enhanced proportional hazard (JM) model.

A general way to represent influence of environmental factors on reliability is to assume that stochastic intensity of the counting process $N(\cdot)$ is a function of some m stochastic processes $E_1(t), \ldots, E_m(t)$ or covariates

$$\lambda(t\ ;\ \mathcal{H}_t, \mathcal{F}_0) = f(t, E_1(t), \ldots, E_m(t), T_1, \ldots, T_{N(t)}, N(t))$$

where $\mathcal{H}_t$ is the past up to time t of the pp and $\mathcal{F}_0$ encompasses the specification of the paths of all covariates. Thus, function $\lambda(t\ ;\ \mathcal{H}_t, \mathcal{F}_0)$ may be thought of as the stochastic intensity of a SEPP driven or modulated by the multivariate environmental process $(E_1(t), \ldots, E_m(t))$. DSPP of Subsection 3.1 is a basic instance of such models and have been widely used in communication engineering and in reliability. Castillo and Siewiorek proposed in [73], a DSPP with a cyclo-stationary stochastic intensity to represent the effect of the workload (measure of system usage) on failure process. That is, intensity is a stochastic process assumed to have periodic mean and autocorrelation function. A classic form for intensity of a DSPP is $\lambda(E_t)$, where (E_t) is a finite Markov process. This is the MMPP discussed in Section 4, where (E_t)

represented the control flow structure of the software. In the same spirit of system in a random environment, Özekici and Sofer [65] use a pp whose stochastic intensity is $(N - N(t))\lambda(E_t)$, where $N - N(t)$ is the remaining number of faults at time t and E_t is the operation performed by the system at t. (E_t) is also assumed to be a finite Markov process. Transient analysis of $N(\cdot)$ may be carried out as in [42] from the Markov property of the bivariate process $(N - N(t), E_t)$. We refer to [65] for details. Note that both pp are instances of SEPP with respective stochastic intensities $E[\lambda(E_t) \mid \mathcal{H}_t]$ and $(N - N(t))E[\lambda(E_t) \mid \mathcal{H}_t]$ ($\mathcal{H}_t$ is the past of the counting process up to time t). The practical purpose of such models has to be addressed. In particular, further investigations are needed to estimate parameters (see [74]).

6.1.4 conclusion

There exists other issues which are of value in SR engineering. In the black-box modeling framework, we can think about alternative to approaches reported in Section 3. The problem of SR growth assessing may be thought of as a problem of statistical analysis of data. Therefore, prediction techniques developed in this area of research can be used. For instance, some authors have considered neural networks (NN). The main interest as SR model is to be nonparametric. Thus, we rejoin discussion on statistical issues of Subsection 6.3. NN may also be used as a classification tool. For instance, identifying fault-prone modules may be performed with a NN classifier. We refer to [4, Chap 17] and references therein for an account of the NN approach. Empirical comparison of the predictive performance of NN models and recalibrated standard models (as defined in [75]) is given in [76]. NN is found to be a good alternative to the standard models.

Computing dependability metrics is not an end in itself in software engineering. A major question is the time to release a software. In particular we have to decide when to stop testing.

Optimal testing time is a problem of decision making under uncertainty. A good account of Bayesian decision theory for solving such a problem is in [2, Chap 6]. In general, software release policies are based on reliability requirement and cost factors. We do not go into further details here. We refer to [12, Chap 8] for a survey up to 1990s, and [77], [78] for more recent contributions to these topics.

6.2 White-box modeling

A challenging issue in SR modeling is to define models taking into account information about the architecture of the software. To go further, software interacts with hardware to make a system. In order to derive model for a system made up of software and hardware, the only point of view is a white-box approach (see [49] and [79] for an account on this topic). We focus on the software product here. Many reasons advocate for a structure-based approach in SR modeling:

- Advancement and widespread used of object oriented systems designs. Reuse of components.
- Softwares are developed in a heterogeneous fashion using components-based software development.
- Early prediction methods of reliability have to take into account the influence about the structure of a software, of testing and reliability of its components.
- Early failure data are prior to the integration phase and thus concern testing part of the software, not the whole product.
- Addressing problem of reliability allocation, resource allocation for modular software.
- Analyze sensitivity of the reliability of the software to the reliability of its components.

As noted in Section 4, the structure-based approach has been largely ignored. Foundations of the Markovian models presented in Section 4 are old. Some limitations of Littlewood's model have been recently addressed in [43], in particular to obtain availability measures. Asymptotic considerations in [45] show that such reliability model tends to be of Poisson-type (homogeneous or not depending on stationarity or not of the failure parameters) when the product has achieved a good level of reliability. It is important to point out that no experience with such kinds of models is reported in the literature. Maybe it is related to questionable assumptions of modeling, as the Markov exchanges of control between modules.

An alternative way to represent the interactions between components of software is to use one of the available modeling tools which are based on stochastic petri nets, SAN network, etc. But, if many of them offer a high flexibility in the representation of the behavior of the software, computation of various metrics are very often performed using automatic generation of Markov chain. So that, these approaches are subject to traditional limitation of Markov modeling: failure rate of the components are not time-dependent; the generated state-space is intractable from the computational point of view; etc.

To overcome limitations of an analytic approach, a widespread method in performance analysis of system is discrete-event simulation. This point of view was initiated by Lyu (see [4, Chap 16]) for software dependability assessment. The idea is to represent the behavior of each component as a nonhomogeneous Markov process whose dynamic evolution only depends on a hazard rate function. At any time t, this hazard rate function depends on the number of failures observed from the component up to time t, as well as the execution time experienced by the component up to time t. Then rate-based simulation technique may be used to obtain a possible realization of such a Markovian arrivals process. The overall hazard rate of the software is actu-

ally a function of the number of failures observed from each component up to time t and of the amount of execution time experienced by each component. We refer to [4, Chap 16] and [80] for details. In some sense, the approach is to simulate the failure process from the stochastic intensity of the counting process of failures.

Long since, the theory of "coherent-system" allows analyzing a system made up of n components through the so-called *structure function*. If x_i ($i = 1, \ldots, n$) denotes the state of component i ($x_i = 1$ if component i is up and 0 otherwise) then the state of the system is obtained from computation of the structure function

$$\Phi(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if system is up} \\ 0 & \text{if system is down.} \end{cases}$$

Function Φ describes the functional relationship between the state of system and the state of its components. Many textbooks on reliability review methods for computing reliability from complex function Φ assuming that state of each component is a Bernoulli random variable (see e.g. [17, Chap 2]). An instance of representation of a 2-module software by a structure function taken into account control flow and data flow is discussed in [2, Chap 7]. This "coherent-system" approach is widely used to analyze the reliability of communication networks. However, it is well-known that exact computation of reliability is then a NP-hard problem. Thus, only structure functions of few dozens of components can be exactly analyzed. Large systems have to be assessed by Monte-Carlo simulation techniques (see e.g. [81], [82]). Moreover, in case of fault-tolerant software, we are faced with a highly reliable system which involves sophisticated simulation procedures to overcome limitations of standard ones. In such a context, an alternative consists in using binary decision diagrams (see [4, Chap15], [83]). We point out that structure function is mainly a functional representation of the system. Thus, many issues discussed in the context black-box modeling also have to be addressed. For instance, how to

incorporate environmental factors identified in [34]?

As we can see, a great deal of research is needed to obtain a white-box model which offers the advantages motivating development of such an approach. Opening the "black-box" to get accurate models is a hard task. Many aspects have to be addressed: definition of what is the structure or architecture of software; what kind of data can be expected to future calibration of models and so on. A first study of the potential sources of SR data available during development is given in [84]. This would help the creation of some benchmark data sets which will allow validating white-box models. What is clear is that actual progress in white-box modeling can only be achieved from an active interaction between the statistics and software engineering communities. All this surely explains why the prevalent approach in SR is the black-box's one.

6.3 Statistical issues

A delicate issue in SR is the statistical properties of estimators used to calibrate models. Main drawbacks of ML method are well-documented in the literature. Finding ML estimators require solving equations which may not always have a solution or may give an inappropriate solution.

For instance, Littlewood and Verrall give in [85], a criterion for $\widehat{N} = \infty$ and $\widehat{\phi} = 0$ (with finite nonzero $\widehat{\lambda} = \widehat{N}\widehat{\phi}$) to be the unique solution of ML equations for (JM) model. Problem of solving ML equations in SR modeling is addressed in [86], [87], [88], [89]. Another well-known drawback is that such ML estimators are usually unstable with small data sets. This situation is basic in SR. Moreover, note that certain models like (JM) assume that software contains a finite number of faults. So, using standard asymptotic properties of ML estimators may be questionable. Such asymptotic results are well-known in the case of i.i.d. sample. But in SR, sample are not i.i.d. That explains why recent investigations on asymptotic normality and consistency of ML estimators for standard SR models use framework of martingale theory

which allows dependence in data. Note that overcoming conceptually finite (expected) number of faults needs unusual concept of asymptotic properties. A detailed discussion is given in [18] and references therein, in [88] for NHPP models. Such works are important because these asymptotic properties are the foundations of interval estimation (a standard alternative to point estimate), of the derivation of confidence interval for parameters, of studies on asymptotic variance of estimators, etc. All these topics must be addressed in details to improve the predictive quality of SR models.

It is clear that any model works well with failure data which correspond to the basic assumptions of the model. But given data, a large part of models is inappropriate. A natural way to overcome too stringent assumptions, in particular of distributional type, is to use nonparametric models. However, the parametric approach remains highly prevalent in SR modeling. Major attempt to gap this fill is [90], where a completely monotonic ROCOF is estimated by regression techniques (see also [91]). A recent work of Littlewood and co-authors [92] uses nonparametric estimates for the distribution of inter-failure times (X_i). This is based on kernel methods for pdf estimation (see [93]). Conclusion of authors is that results are not very impressive but more investigation is needed, in particular using various kernel functions. We can think for instance to wavelets ([94]). Similar discussion may be done in Bayesian approach of SR modeling. Specifically, most bayesian inference for NHPP assumes a parametric model for ROCOF and proceeds with prior assumption on the unknown parameters. In such a context, an instance of a nonparametric bayesian approach has recently been used in [95]. Conceptually, nonparametric approach is promising, but is computationally intensive in general and is not easy to comprehend.

These developments may be viewed as preliminary works using statistical methods based

on the so-called dynamic approach of counting processes as reported for instance in the book of Andersen et al [69] (the bibliography gives a large account for research on this area). The Aalen pioneer-work was on the multiplicative intensity model which, roughly speaking, writes the stochastic intensity associated with a counting process as

$$\lambda(t\ ;\ \mathcal{H}_t, \sigma(Y_s, s \leq t)) = \lambda(t)\, Y(t)$$

where $\lambda(\cdot)$ is a nonnegative deterministic function, whereas $Y(\cdot)$ is a nonnegative observable stochastic process whose value at any time t is known just before t ($Y(\cdot)$ is a predictable process). Nonparametric estimation for such a model is discussed in [69, Chap 4]. A Cox-type model may be obtained in choosing $Y(t) = \exp\left(\sum_j \beta_j Z_j(t)\right)$ with stochastic processes as covariates Z_j (see [96]). We can also consider additive intensity models when the multiplicative form involves multivariate functions $\lambda(\cdot)$ and $Y(\cdot)$ (e.g. see [97]). Conceptually, this dynamic approach is appealing because it is well supported by a lot of theoretic results. It is worthy of note that martingale theory may be of some value for analyzing static models as capture-recapture models (see [98], [99] for details). Thus, applicability of the dynamic point of view on pp in the small data set context of software engineering is clearly a direction of further investigations (see e.g.[18] for such an account). Moreover, if it is shown that gain in predictive validity is high with respect to standard approaches, then a user-oriented "transfer of technology" must follow. That is, friendly tools for using such statistical material must be developed.

References

[1] Gaudoin O. *Statistical tools for software reliability evaluation (in french)*. PhD thesis, Université Joseph Fourier - Grenoble I, 1990.

[2] Singpurwalla N.D., Wilson S.P. *Statistical Methods in Software Engineering: Reliability and Risk*. Springer, 1999.

[3] Musa J.D., Iannino A., Okumoto K. *Software Reliability: Measurement, Prediction, Application.* McGraw-Hill International Editions, Computer Science Series, 1987.

[4] Lyu M.R. editor. *Handbook of software reliability engineering*. McGraw-Hill, 1996.

[5] *Software reliability estimation and prediction handbook*. American Institute of Aeronautics and Astronautics, 1992.

[6] Pham H. *Software Reliability*. Springer, 2000.

[7] Everett W., Keene S., Nikora A. Applying software reliability engineering in the 1990s. *IEEE Trans. Reliability* 1998; 47:372–378.

[8] Ramani S., Gokhale S.S., Trivedi K.S. Software reliability estimation and prediction tool. *Performance Evaluation* 2000; 39:37–60.

[9] Laprie J.-C. *Dependability: Basic Concepts and Terminology.* Springer, 1992.

[10] Barlow R.E., Proschan F. *Statistical Theory of Reliability and Life Testing.* Holt, Rinehart and Winston, NY, 1975.

[11] Ascher H., Feingold H. *Repairable Systems Reliability.* Volume 7 Lecture Notes in Statistics. Marcel Dekker, Inc., NY, 1984.

[12] Xie M. *Software Reliability Modeling*. World Scientific Publishing, UK, 1991.

[13] Briand L.C., El Emam K., Freimut B.G. A comprehensive evaluation of capture-recapture models for estimating software defect content. *IEEE Trans. Software Eng.* 2000; 26:518–540.

[14] Chen Y., Singpurwalla N.D. Unification of software reliability models by self-exciting point processes. *Adv. Appl. Probab.* 1997; 29:337–352.

[15] Snyder D.L., Miller M.I. *Random Point Processes in Time and Space*. Springer, 1991.

[16] Jelinski Z., Moranda P.B. Software reliability research. In W. Freiberger, editor. *Statistical Methods for the Evaluation Of Computer System Performance*. Academic Press, 1972; 465–484.

[17] Aven T., Jensen U. *Stochastic models in reliability*. Volume 41 of *Applications of Mathematics*. Springer, 1999.

[18] van Pul M.C. A general introduction to software reliability. *CWI Quaterly* 1994; 7:203–244.

[19] Laprie J.-C., Kanoun K., Béounes C., Kaâniche M. The KAT (knowledge-action-transformation) approach to the modeling and evaluation of reliability and availability growth. *IEEE Trans. Software Eng.* 1991; 17:370–382.

[20] Cox D.R., Isham V. *Point Processes*. Chapman & Hall, 1980.

[21] Trivedi K.S. *Probability and Statistics with reliability, queueing and computer science applications*. Prentice-Hall, Engelwood Cliffs NJ, 1982.

[22] Moranda P.B. Predictions of software reliability during debugging. In *Annual Reliability and Maintainability Symposium* 1975; 327–332.

[23] Shanthikumar J.G. Software reliability models: A review. *Microelectronics and Rel.* 1983; 23:903–943.

[24] Littlewood B. Stochastic reliability-growth : a model for fault-removal in computer programs and hardware designs. *IEEE Trans. Rel.* 1981; 30:313–320.

[25] Osaki S., Yamada S. Reliability growth models for hardware and software systems based on nonhomogeneous poisson processes: A survey. *Microelectronics and Rel.* 1983; 23:91–112.

[26] Littlewood B. Forecasting software reliability. In S. Bittanti, editor. *Software Reliability Modeling and Identification*. Lecture Notes in Computer Science 341. Springer, 1988; 141–209.

[27] Miller D.R. Exponential order statistic models for software reliability growth. *IEEE Trans. Software Eng.* 1986; 12:12–24.

[28] Littlewood B., Verrall J.L. A bayesian reliability growth model for computer software. *Appl. Statist.* 1973; 22:332–346.

[29] Mazzzuchi T.A., Soyer R. A bayes empirical-bayes model for software reliability. *IEEE Trans. Rel.* 1988; 37:248–254.

[30] Schick G.J., Wolverton R.W. Assessment of software reliability. In *Operation Research*. Physica-Verlag, 1973; 395–422.

[31] Singpurwalla N.D., Soyer R. Assessing (software) reliability growth using a random coefficient autoregressive process and its ramification. *IEEE Trans. Software Eng.* 1985; 11:1456–1464.

[32] Singpurwalla N.D., Soyer R. Nonhomogeneous autoregressive processes for tracking (software) reliability growth, and their bayesian analysis. *J. Roy. Statist. Soc. Series B* 1992; 54:145–156.

[33] Al-Mutairi D., Chen Y., Singpurwalla, N.D. An adaptative concatenated failure rate model for software reliability. *J. Amer. Statist. Ass.* 1998; 93:1150–1163.

[34] Pham L., Pham H. Software reliability models with time-dependent hazard function based on bayesian approach. *IEEE Trans. Systems, Man and Cyber. Part A* 2000; 30:25–35.

[35] Cheung R.C. A user-oriented software reliability model. *IEEE Trans. Software Eng.* 1980; 6:118–125.

[36] Littlewood B. Software reliability model for modular program structure. *IEEE Trans. Rel.* 1979; 28:241–246.

[37] Siegrist K. Reliability of systems with Markov transfer of control. *IEEE Trans. Software Eng.* 1988; 14:1049–1053.

[38] Kaâniche M., Kanoun K. The discrete time hyperexponential model for software reliability growth evaluation. In *Int. Symp. on Software Reliability (ISSRE)*. 1992; 64–75.

[39] Littlewood B. A reliability model for systems with Markov structure. *Appl. Statist.* 1975; 24:172–177.

[40] Kubat P. Assessing reliability of modular software. *Oper. Res. Letters* 1989; 8:35–41.

[41] Goseva-Popstojanova K., Trivedi K.S. Architecture-based approach to reliability assessment of software systems. *Performance Evaluation* 2001; 45:179–204.

[42] Ledoux J., Rubino G. Simple formulae for counting processes in reliability models. *Adv. Appl. Probab.* 1997; 29:1018–1038.

[43] Ledoux J. Availability modeling of modular software. *IEEE Trans. Rel.* 1999; 48:159–168.

[44] Kabanov Y.M, Liptser R.S., Shiryayev A.N. Weak and strong convergence of the distributions of counting processes. *Theor. Prob. Appl.* 1983; 28:303–336.

[45] Gravereaux J.B., Ledoux J. Poisson approximation for some point processes in reliability. Technical report, Institut National des Sciences Appliquées, Rennes, France, 2001.

[46] Ledoux J. Littlewood reliability model for modular software and Poisson approximation In *Mathematical Methods for Reliability*. 2002; 367–370.

[47] Abdel-Ghaly A.A, Chan P.Y., Littlewood B. Evaluation of competing software reliability predictions. *IEEE Trans. Software Eng.* 1986; 12:950–967.

[48] Brocklehurst S., Kanoun K., Laprie J-C., Littlewood B., Metge S., Mellor P. ,et al. Analyses of software failure data. Technical report No91173, Laboratoire d'Analyse et d'Architecture des Systèmes, Toulouse, France, May 1991.

[49] Kanoun K. Software dependability growth: characterization, modeling, evaluation (in french). Technical Report 89.320, LAAS, Doctor ès Sciences thesis, Polytechnic National Institute, Toulouse, 1989.

[50] Gilks W.R., Richardson S., Spiegelhalter D.J., editors. *Markov Chain Monte Carlo in practice*. Chapman & hall, 1996.

[51] Kuo L., Yang T.Y. Bayesian computation of software reliability. *J. Comp. and Graphical Stat.* 1995; 4:65–82.

[52] Kuo L., Yang T.Y. Bayesian computation for nonhomogeneous poisson processes in software reliability. *J. Amer. Statist. Ass.* 1996; 91:763–773.

[53] Kuo L., Lee J.C., Choi K., Yang T.Y. Bayes inference for s-shaped software-reliability growth models. *IEEE Trans. Rel.* 1997; 46:76–80.

[54] Kremer W. Birth-death and bug counting. *IEEE Trans. Rel.* 1983; 32:37–47.

[55] Gokhale S.S., Philip T., Marinos P.N. A non-homogeneous markov software reliability model with imperfect repair. In *Int. Performance and Dependability Symposium*. 1996; 262–270.

[56] Shanthikumar J.G., Sumita U. A software reliability model with multiple-error introduction and removal. *IEEE Trans. Rel.* 1986; 35:459–462.

[57] Sahinoglu H. Compound-poisson software reliability model. *IEEE Trans. Software Eng.* 1992; 18:624–630.

[58] Fenton N.E., Pfleeger S.L. *Software metrics : a rigorous and practical approach (2nd Edition)* International Thomson Computer Press, 1996.

[59] Khoshgoftaar T.M., Allen E.B, Wendell D.J., Hudepohl J.P Classification-tree models of software-quality over multiple releases. *IEEE Trans. Rel.* 2000; 49:4–11.

[60] Khoshgoftaar T.M., Allen E.B. A practical classification-rule for software-quality models. *IEEE Trans. Rel.* 2000; 49:209–216.

[61] Gokhale S.S., Trivedi K.S. A time/structure based software reliability model. *Ann. Software Eng.* 1999; 8:85–121.

[62] Goel A.L. Software reliability models: Assumptions, limitations, and applicability. *IEEE Trans. Software Eng.* 1985; 11:1411–1423.

[63] Xie M., Hong G.Y., Wohlin C. A practical method for the estimation of software reliability growth in the early stage of testing. In *ISSRE*. 1997; 116–123.

[64] Pasquini A., Crespo A.N., Matrella P. Sensitivity of reliability-growth models to operational profile errors vs. testing accuracy. *IEEE Trans. Rel*. 1996; 45:531–540.

[65] Özekici S., Soyer R. Reliability of software with an operational profile. Technical report, The George Washington University, Department of Management Science, 2000.

[66] Zhang X., Pham H. An analysis of factors affecting software reliability. *J. of Systems and Software* 2000; 50:43–56.

[67] Kalbfleisch J.D., Prentice R.L. *The statistical analysis of failure time data*. Wiley, 1980.

[68] Cox C.R., Oakes D. *Analysis of Survival Data*. Chapman and Hall, London, 1984.

[69] Andersen P.K., Borgan O., Gill R.D., Keiding N. *Statistical models on counting processes*. Springer Series in Statistics. Springer, 1993.

[70] Saglietti F. Systematic software testing strategies as explanatory variables of proportional hazards. In *SAFECOMP'91*. 1991; 163–167.

[71] Wright D. Incorporating explanatory variables in software reliability models. In *Second Year Report of PDCS, Vol 1*. Esprit BRA Project 3092, May 1991.

[72] Pham H. Software reliability. In J.G. Webster, editor, *Wiley Encyclopedia of Electrical and Electronics Engineering*. John Wiley & Sons, 1999; 565–578.

[73] Castillo X., Siewiorek D.P. A workload dependent software reliability prediction model. In *12th Int. Symp. Fault-Tolerant Computing* 1982; 279–286.

[74] Koch G., Spreij P. Software reliability as an application of martingale & filtering theory. *IEEE Trans. Rel.* 1983; 32:342–345.

[75] Brocklehurst S., Chan P.Y., Littlewood, Snell J. Recalibrating software reliability models. *IEEE Trans. Software Eng.* 1990; 16:458–470.

[76] Sitte R. Comparison of software-reliability-growth predictions: Neural networks vs parametric-recalibration. *IEEE Trans. Rel.* 1999; 49:285–291.

[77] Pham H., Zhang X. A software cost model with warranty and risk costs. *IEEE Trans. Computers* 1999; 48:71–75.

[78] Pham H., Zhang X. Software release policies with gain in reliability justifying the costs. *Ann. Software Eng.* 1999; 8:147–166.

[79] Laprie J.-C., Kanoun K. X-ware reliability and availability modeling. *IEEE Trans. Software Eng.* 1992; 18:130–147.

[80] Lyu M.R., Gokhale S.S., Trivedi K.S. Reliability simulation of component-based systems. In *ISSRE*. 1998; 192–201.

[81] Ball M.O., Colbourn C., Provan J.S. Network models. In Monma C. Ball M.O., Magnanti T. and Nemhauser G., editors. *Handbook of Operations Research and Management Science*. Elsevier, 1995; 673–762.

[82] Rubino G. Network reliability evaluation. In Bagchi K. and Walrand J., editors. *State-of-the-art in performance modeling and simulation*. Gordon and Breach Books, 1998.

[83] Editors: Limnios N., Rauzy A. Special issue on binary decision diagrams and reliability. *European Journal of Automation* 1996; 30(8).

[84] Smidts C., Sova D. An architectural model for software reliability quantification: sources of data. *Rel. Eng. and System Safety* 1999; 64:279–290.

[85] Littlewood B., Verrall J.L. Likelihood function of a debugging model for computer software reliability. *IEEE Trans. Rel.* 1981; 30:145–148.

[86] Huang X.Z. The limit condition of some time between failure models of software reliability. *Microelectronics and Rel.* 1990; 30:481–485.

[87] Hossain S.A., Dahiya R.C. Estimating the parameters of a non-homogeneous poisson-process model for software reliability. *IEEE Trans. Rel.* 1993; 42:604–612.

[88] Zhao M., Xie M. On maximum likelihood estimation for a general non-homogeneous poisson process. *Scand. J. Statist.* 1996; 23:597–607.

[89] Knafl G., Morgan J. Solving ML equations for 2-parameter poisson-process models for ungrouped software-failure data. *IEEE Trans. Rel.* 1996; 45:43–53.

[90] Miller D.R., Sofer A. A nonparametric software-reliability growth model. *IEEE Trans. Rel.* 1991; 40:329–337.

[91] S. Brocklehurst, B. Littlewood. New ways to get accurate reliability measures. *IEEE Software* 1992; 9(4):34–42.

[92] Barghout M., Littlewood B., Abdel-Ghaly A.A. A non-parametric order statistics software reliability model. *J. Testing, Verification and Rel.* 1998; 8:113–132.

[93] Silverman B.W. *Density estimation for statistics and data analysis.* Chapman & Hall, 1986.

[94] Antoniadis A., Oppenheim G., editors. *Wavelets and Statistics.* Volume 103 of *Lecture Notes in Statistics.* Springer, August 1995.

[95] Kuo L., Ghosh S.K. Bayesian nonparametric inference for nonhomogeneous poisson processes. Technical Report 9718, University of Connecticut, Storrs, 1997.

[96] Slud E.V. Some applications of counting process models with partially observed covariates. *Telecommunication Systems* 1997; 7:95–104.

[97] Pijnenburg M. Additive hazard models in repairable systems reliability. *Rel. Eng. and System Safety* 1991; 31:369–390.

[98] Lloyd C.J, Yip P.S.F, Chan Sun K. Estimating the number of errors in a system using a martingale approach. *IEEE Trans. Rel.* 1999; 48:369–376.

[99] Yip P.S.F, XI L., Fong D.Y.T, Hayakawa Y. Sensitivity-analysis and estimating the number-of-faults in removing debugging. *IEEE Trans. Rel.* 1999; 48:300–305.