September 23, 1997

Rare Events in Stochastic Dynamical Systems and Failures in Ultra-Reliable Reactive Programs

A. Galves
Universidade de São Paulo
M. C. Gaudel
Universidade de São Paulo and Université de Paris Sud

Summary.

This paper presents a model of reactive process-control programs as stochastic processes. This makes it possible to prove that the law of the first failure time in a ultra-reliable reactive program is approximately exponential. This fact is usually assumed on the basis of a constant probability of failure and the independence of the successive inputs. These hypotheses *a priori*, do not hold for reactive process-control programs. Our result provides a rigourous justification of the exponential assumption for a large class of ultra-reliable reactive programs.

Keywords. Software reliability, ultrareliability, Markov chains, occurrence time of a rare event, exponential law.

AMS 1991 classification numbers. 60B12, 60K10, 68M15.

1. Introduction.

Reactive programs are a class of software whose behavior is highly dependent of the environment. All the computerized process-control systems, such as nuclear plant controllers, flight navigation assistants, etc, embed programs of this category. These programs are not supposed to terminate, but to iterate a cyclic behavior. Namely, at the beginning of every cycle, some inputs/signals are entered from the system under control; then, depending on those inputs and on the internal state of the program, some treatment is performed; finally, some outputs/signals are emitted. These programs must be ultra-reliable, a generally admitted notion of ultra-reliability being a probability of catastrophic failure smaller than 10^{-7} per hour (Butler and Finelli 1993).

The basic problem of estimating quantities such as failure rates of ultra-reliable programs has been found difficult, since, by definition, large samples of failure occurrences are not easily available (Littlewood and Strigini 1993, Butler and Finelli 1993). Besides the problem of sampling, making precise statements about the laws of occurrence times of failures in such systems is an even more difficult question.

In this paper we present a model of ultra-reliable reactive programs which leads to a rigorous general result concerning the asymptotical law of the occurrence times of a first failure in this kind of system. This result says that this law can be approximated in a sharp way by an exponential law. Moreover, the more reliable is the system, the sharper is the approximation.

In this model, we consider two types of failures which are likely to occur when executing such programs:

i) Failures which are due to the fact that the values of the inputs are perturbated by an external random noise;

ii) Failures which are due to a fault in the program.

Failures of the first type belong to the class of transient, or temporary failures.

The idea that the distribution of a failure time can be approximated by an exponential law is not new in software reliability. The most widely used model for failure process characterization assumes the time to failure to be exponentially distributed. Weibull distribution or Gamma distributions are used in some cases, but most software reliability models rely upon the *exponential assumption*.

Software reliability models have focused on growing reliability, taking into account that after a failure, the source of the failure is corrected in the program (with variants depending on the correction and maintenance policy, cf. Laprie and Kanoun, 1996). There are too many models to mention all of them. There are excellent surveys, for instance Musa, Ianino and Okumoto (1987), Xie (1993), and Lyu (1996).

Among the main models, one must mention: the basic execution time model (Musa, 1975); the hyperexponential model (Laprie, 1984); the logarithmic Poisson model (Musa and Okumoto, 1984); and the Littlewood-Verral model, which is a bayesian model reflecting the fact that if no failures occur, the reliability estimation grows since the confidence in the software increases (Littlewood and Verral, 1973).

All these models assume an exponential distribution of the occurrence of failures.

The exponential assumption is usually based on the other assumption that the occurrence of a failure at each step of the program has a probability which is essentially constant and strictly positive, and is independent of the former steps. In this scenario, the system is supposed to behave as a sequence of independent tests with a fixed probability of failure, as in tossing a coin repeatedly. This is the classical Bernouilli trials model. Under these assumptions, the exponential approximation follows immediately as an elementary result of Probability Theory (cf. Keilson 1979, Laprie and Kanoun 1996).

However, for reactive systems, such as those we are interested in, the independence assumption does not hold, due to the iterative nature of the program and the physical constraints of the system under control. There are few discussions of this point in the literature, with some interesting exceptions (Crow and Singpurwala 1984, Strigini 1996). This last paper in particular argues extensively against the classical Bernouilli trials model for process control software.

Moreover, as for the constant probability of failures assumption, for ultra-reliable programs, it seems natural to assume that this probability is zero for most of the steps.

With our model it is possible to prove for a large class of ultra- reliable reactive programs that the exponential approximation holds, even if the assumption of independence at each step does not hold. Informally speaking, a weak independence property, such as a mixing property, is enough.

Exponential approximation for the laws of occurrence time of rare events in stochastic processes is an old subject which has attracted a lot of interest recently. The pioneer paper in this area is Doeblin (1940), who studied the Poisson approximation for the Gauss transformation. In the context of Markov chains, the convergence of the occurrence time of a rare event to the exponential law was first studied by Bellmann and Harris (1951) and Harris (1953). Then after a long period in which apparently nothing appeared in the area, several papers and books studying the problem for different types of Markov processes were published, starting with Keilson (1979), Aldous (1982), Korolyuk and Sil'vestrov (1984), Cassandro, Galves, Olivieri and Vares (1984), Kipnis and Newman (1985), Cogburn (1985), Lebowitz and Schonmann (1987), among others. Let us quote also two recent papers by Olivieri and Scoppola (1995 and 1996) who present a very elegant treatment of the problem in the context of Markov processes. In the context of dynamical systems the question was considered by Galves and Schmitt (1990), Collet, Galves and Schmitt (1992), Collet and Galves (1993 and 1995).

Part of these papers were motivated by the modeling of physical phenomena like metastability or intermitency. In any case, with the exception of Aldous (1982), all the others are only interested in a qualitative result, and don't present sharp upper bounds for the rate of convergence to the exponential law. Recently this was done for reversible and finite Markov chains by Aldous and Brown (1992, 1993), for some interacting Markovian systems by Ferrari, Galves and Landim (1994) and Ferrari, Galves and Liggett (1995) and for ϕ-mixing dynamical systems by Galves and Schmitt (1997).

The paper is organized as follows. In section 2, we give a toy example of the kind of reactive programs we are considering. In section 3, we describe the model. In section 4, we apply the model to the program of section 2. In section 5 we show how the model leads to an exponential approximation for the law of failures times. Section 6 summarizes our conclusions and presents future directions of research.

2. A toy example: the overspeed control.

In this section we present a toy program which will help understanding why dynamical systems can be used as a model of some reactive programs, namely iterative programs controlling continuous physical processes. The model will be presented in the next section.

The following toy example is freely derived from the specification of the overspeed control module of the automatic pilot of a train (Dauchy, Gaudel and Marre, 1993). The actual program is on board of the train and receives some information from the captors of the train (from which it derives the current speed, among other pieces of information) and from a central post of control (for instance the position of the preceeding train). Depending on some internal data (for instance, the geography of the railway), it computes and updates several quantities, among them the maximum allowed speed. If there is a risk for the train to pass over this maximum speed during the current cycle, it raises an alarm and commands an emergency stop. Thus, at every cycle, the program transforms a set of values, namely the input signals, the values of its internal variables and the output signals, into a new one.

In the toy version below, $Acceleration$, $Measured\_Speed$, $Calculated\_Speed$ are the variables where are stored the successive values of the current acceleration, the current speed and the intended speed at the next iteration. $Measured\_Speed$ is an input and $Acceleration$ is an output. The computation and the check of $Calculated\_Speed$, the intended speed at the next iteration, is a classical way of preventing failure by anticipating what is likely to happen at the next step. $EMERGENCY\_STOP$ is a procedure call which provokes the emission of the adequate signals to stop the train as quickly as possible. low_speed, $medium\_speed$, $speed\_limit$, $a1$, $a2$, $a3$, and $delta\_t$ are constants.

The program can be summarized by:

- When the speed of the train is less than low_speed, the program chooses an acceleration $a1$;

- When the speed is between low_speed and $medium\_speed$, the program chooses an acceleration $a2$;

- When the speed is between $medium\_speed$ and $speed\_limit$, there is no more acceleration and the train slows down naturally.

There are the obvious constraints *low_speed* $<$ *medium_speed* $<$ *speed_limit* and $a1 > a2$ on the constants.

Note that this is a very simplified version. The train is supposed in an environment where there is no reasons to stop or to slow down: the only role of the toy program is to maintain a sufficient speed and to ensure that this speed is never above the limit.

A failure is to let the train reach a speed above the limit.

The text of the program is given below.

```
begin SPEED_CONTROL ;
Acceleration, Measured_Speed, Calculated_Speed := 0 ;
while true loop ;
  read(Measured_Speed) ;
  if Measured_Speed < low_speed
      then Acceleration := a1 ;
          Calculated_Speed :=
            Measured_Speed + (a1 - a3) * delta_t ;
  else if low_speed =< Measured_Speed < medium_speed
      then Acceleration := a2 ;
          Calculated_Speed :=
            Measured_Speed + (a2 - a3) * delta_t ;
  else if medium_speed =< Measured_Speed
      then Acceleration := 0 ;
          Calculated_Speed := Measured_Speed - a3 * delta_t ;
  end if ;
  if Calculated_Speed > speed_limit
      then EMERGENCY_STOP ;
  else write(Acceleration) ;
  end if ;
end loop ;
end SPEED_CONTROL
```

In the next section, we present a stochastic model for this type of program.

3. A stochastic model of the behavior of reactive programs.

The presentation of the model is given in three parts for a better readibility.

3.1 A preliminary model.

Let us start with a preliminary version of the model, describing the program and ignoring the environment.

We consider the program as a function

$$f : E \longrightarrow E,$$

where E is the set of the lists inputs, internal variables, and outputs. E is a discretized, finite, cartesian space

$$E = E_1 \times E_2 \times \cdots \times E_m.$$

More precisely, the program is defined by cases; namely there is a disjoint partition defined by

$$I : E \longrightarrow \{1, 2, \ldots, k\},$$

and k functions $f_1, f_2, \ldots, f_k$ from E into E. The function f is defined as

$$x \longrightarrow f_{I(x)}(x).$$

Thus, the execution of the program can be considered as a dynamical system

$$X_n = f_{I(X_{n-1})}(X_{n-1}),$$

where the initial state X_0 is suitably defined.

An important point to be added to our model is the existence of safety constraints on the behavior of the system under control. For all reachable state x, a certain property $\mathcal{C}(x)$ must hold.

It is easy to see that the toy program of section 2 follows this pattern. In section 4 we develop the complete model for this program.

3.2 Introducing interactions with the environment.

Until now we have a model for the behavior of a program in an ideal world, *i.e.* a world which does not interact with the system under control. A real system does not follow exactly the model above. Let us suppose the system starts with an initial state X_0. In an ideal world, after n steps the system would be at state $f^{(n)}(X_0)$. Actually, due to external circumstances, the system turns out to be in a state Y_n, which is produced from X_0 by a sequence of random perturbations of f.

Formally,

$$Y_n = R(f(Y_{n-1}), \xi_n) \ ,$$

where where $\xi_n, n \geq 1$, is an independent sequence of real random variables with uniform distribution on E, and R is a suitable function.

Moreover, the measures received by the program may not be completely accurate. Thus, at the nth step, the state of the program is not exactly the state of the system, Y_n, but some aproximation of it, namely

z

$$Z_n = M(Y_n, \zeta_n) \ , \tag{3.1}$$

where the $\zeta_n, n \geq 1$, are independent uniform random variables on E, and M is a suitable function.

Let $(\Omega, \mathcal{F}, I\!P)$ be the probability space where the independent uniform random variables ξ_n and $\zeta_n, n \geq 1$, are defined.

What is intended in the program is

$$f_{I(Z_{n-1})}(Z_{n-1}) \ .$$

What happens in reality is

y

$$Y_n = R(f_{I(Z_{n-1})}(Y_{n-1}), \xi_n) \ . \tag{3.2}$$

The behavior of the program can thus be modeled by a stochastic process $(W_n)_{n \geq 0}$ defined on $(\Omega, \mathcal{F}, I\!P)$ by

$$W_n = (Y_n, Z_n) \ .$$

This stochastic process is a Markov chain since, for each $n \geq 1$, we can write W_n as

$$W_n = F(W_{n-1}, \xi_{n-1}, \zeta_{n-1}) \ ,$$

where the function F is defined by applying (3.1) and (3.2) and the random variables ξ_n and ζ_n are independent of $(W_1, \cdots, W_{n-1})$.

The occurrence of a failure corresponds to the visit to a state of the chain $W_n = (Y_n, Z_n)$, where $\mathcal{C}(Y_n)$ does not hold. Namely, $I(Z_{n-1})$, the case chosen in the program, maps the actual state of the system Y_{n-1} to a state Y_n where $\mathcal{C}(Y_n)$ does not hold. To complete the definition of the Markov chain $(W_n)_{n \geq 0}$, we must specify what happens after a failure.

It depends on the characteristics of the system under control. Different cases must be considered and we postpone this discussion to part 5 below.

To summarize, the behavior of a reactive program can be modeled by

i. a partition $I : E \longrightarrow \{1, 2, \ldots, k\}$

ii. a finite family of functions $f_1, f_2, \ldots, f_k$ from E into E,

iii. a Markov chain $W_n = (Y_n, Z_n)$ where $Y_n = R(f_{I(Z_{n-1})}(Y_{n-1}), \xi_n)$ and $Z_n = M(Y_n, \zeta_n)$, ξ_n is the noise introduced by the environment, and ζ_n is the noise introduced by the measures. Besides, $Y_0 = Z_0 = X_0$.

iv. a constraint $\mathcal{C}(x)$, on the states of the system under control. A failure is a visit to a state $W_n = (Y_n, Z_n)$ of the Markov chain such that $\mathcal{C}(Y_n)$ does not hold.

Remark 1. An originality of this model is that both the state of the program and the state of the system under control are considered. Failures result from some discrepancy between them. It is different from more usual approaches where the program alone is considered. It makes it possible to describe failures as events in a Markov chain. This will be exploited in section 5.

Remark 2. In this model, we consider both cases where the program is "correct" and where it contains some fault. A fault means that there exists a subset ϕ of the states reachable by the program such that when $z \in \phi$, $\mathcal{C}(f_{I(z)}(z))$ does not hold. Since we consider ultra-reliable programs, the probability to reach a state belonging to ϕ is very small. A program is said to be correct with respect to $\mathcal{C}$, when I, $f_1, f_2, \ldots, f_k$, and $\mathcal{C}(x)$ are such that for any z such that $\mathcal{C}(z)$ holds, then $\mathcal{C}(f_{I(z)}(z))$ also holds. In words, the program preserves the constraints. This is a logical property of the program. In this case, the only source of failures is the environment represented in the model by the action of the random variables ξ_n and ζ_n through the functions R and M. Thus we have only what is usually called transient failures. It is clear that this notion of correction is debatable since the characteristics of the environment must be taken into account when developing a program.

3.3 Adding independent signals.

This model is sufficient to describe the behavior of some reactive programs, for instance the one given in section 2. However, in general, in addition of the inputs a reactive program receives some signals, which indicate some external event and induce some change in the behavior of the system (for instance, in the example of section 2, we could add the fact that a passenger can raise an alarm and stop the train.) Such signals correspond to random events which are independent of the former history of the system. Let us denote by $S_1, S_2, \ldots, S_n, \ldots$, with $S_i \in \{0, 1, \ldots, a\}$, the corresponding stochastic process. The impact of these external signals on the program can be described by a more precise partition which depends at step n on both Z_n and S_n. This leads to a more elaborated model. Let us consider

$$\bar{I} : E \times \{0, 1 \ldots, a\} \longrightarrow \{1, 2, \ldots, k\}$$

The successive states of the system are now $(\bar{Y}_n, S_n)_{n \geq 0}$ where

$$\bar{Y}_n = R(f_{\bar{I}(\bar{Z}_{n-1}, S_{n-1})}(\bar{Y}_{n-1}), \xi_n)$$

and

$$\bar{Z}_n = M(\bar{Y}_n, \zeta_n) \ .$$

Thus, we make a distinction in the model between signals and input values. The reason is that in a control-command interaction, the successive input values are not an independent process from the successive states of the system. They strongly depend on the previous state and on the commands (i.e. the outputs) emitted at the previous cycle. On the contrary, a signal can occur at any moment, independently of what have happened before.

For the sake of readibility, we do not consider the case with signals in the sequel. We briefly come back to it at the end of part 5.

4. The toy example revisited.

We come back to the example of section 2.

Let us call *Num* the set of values of the numeric type of the three variables, *Acceleration*, *Measured_Speed*, and *Calculated_Speed*. Thus

$$E = Num \times Num \times Num.$$

The cases considered in the program only depend on the second and third variables, namely *Measured_Speed* and *Calculated_Speed*, and we have

$$I(a, m, c) = \begin{cases} 1, \text{ if } \ (m < low\_speed) \wedge (c \leq speed\_limit) \ ; \\ 2, \text{ if } \ (low\_speed \leq m < medium\_speed) \wedge (c \leq speed\_limit) \ ; \\ 3, \text{ if } \ (medium\_speed \leq m) \wedge (c \leq speed\_limit) \ ; \\ 4, \text{ if } \ c > speed\_limit. \end{cases}$$

In an ideal world, the train behaves exactly as intended in the program, i.e. the measured speed is always the calculated speed of the previous step, thus

$$f_1(a, m, c) = (a1, c, m + (a1 - a3) \times \Delta t) \ ;$$

$$f_2(a, m, c) = (a2, c, m + (a2 - a3) \times \Delta t) \ ;$$

$$f_3(a, m, c) = (0, c, m - a3 \times \Delta t) \ ;$$

and

$$f_4(a, m, c) = (0, 0, 0) \ .$$

The function $f_4(a, m, c)$ corresponds to the $EMERGENCY\_STOP$ procedure.

Therefore, the program defines the following dynamical system (noted $(X_n)_{n\geq 0}$ in part 3 above)

$$X_0 = (a_0, m_0, c_0) = (0, 0, 0) \ ,$$

and, for any $n \geq 1$,

$$X_n = (a_n, m_n, c_n) = f_{I(a_{n-1}, m_{n-1}, c_{n-1})}(a_{n-1}, m_{n-1}, c_{n-1}) \ .$$

The constraint on the train is that m, the current speed, must be less or equal to $speed\_limit$, i.e.

$$\mathcal{C}(a, m, c) = (m \leq speed\_limit) \ .$$

Actually, at each iteration the actual speed is different from the previous calculated one because of the environment, but it is clearly dependent of it. Since the perturbation of the speed is due to many external random events, it seems reasonable to state

$$Y_n = (a_n, m_n + \xi_n, c_n), \text{ with } (a_n, m_n, c_n) = f(Y_{n-1}) \ ,$$

where the ξ_n are independent random variables, following a normal law of mean 0 and of small variance.

The program does not receive exactly this actual value, but an approximation of it which depends on the way the measures are performed. For instance

$$Z_n = (a'_n, m'_n + \zeta_n, c'_n), \text{ with } (a'_n, m'_n, c'_n) = Y_n \ ,$$

where the ζ_n are independent random variables, following a uniform law of mean 0 and of small variance. Consequently, the next choice of the case by the program is based on $m'_n + \zeta_n$, but the acceleration will be applied to a train with speed m'_n.

5. Failures as rare events in a Markov chain.

We recall that in section 3.2 we defined a failure as a visit of the chain $W_n = (Y_n, Z_n)$ to a state $w = (y, z)$, where the safety constraint $\mathcal{C}(y)$ does not hold. Let us call these states of the chain *failure states* and let us denote the set of all failure states by Λ.

To avoid uninteresting and trivial pathologies, we shall assume that it is always possible, in a finite number of steps, to reach Λ from any state outside Λ. We shall assume, that from any state outside Λ it is possible to reach any other state outside Λ in a finite number of steps.

When modeling the behavior of reactive programs, the first assumption is rather realistic, since it is very rare to have completely safe states, disconnected from the other states. The second assumption may not be verified, for systems with different functioning modes, with no switching between modes. A natural way to deal with this situation is to consider one Markov chain by functioning mode.

As pointed out in section 3.2, to complete the definition of the Markov chain W_n we need to specify its transition probability after the occurrence of a failure.

One can imagine systems where a failure always results in a catastrophe. One way to deal with such systems is to assume that the failure states are absorbing states of the Markov chain. Actually this is not a good solution, since in this case, the unique invariant probability measure for the Markov chain is a convex combination of Dirac delta measures concentrated on the absorbing states. Therefore this invariant probability do not tell us anything about the behavior of the processes before a failure.

A better solution is to make the chain restart with a fixed departure state, in the next step after the occurrence of a failure. Since we are assuming that is always possible, in a finite number of steps, to reach any state from any state, the Markov chain defined this way is irreducible. In this case, the unique invariant measure probability for the Markov chain has the entire set E as support, and gives a rich information concerning the behavior of the chain before a failure occurs for the first time.

One may also imagine systems where a failure does not always imply a catastrophe. In this case, the system may enter in the failure set Λ and nevertheless goes on functioning. In the next step, it may either leave the failure set, or remain inside Λ. It was empirically observed that in such systems failures may occur in clusters as reported in Crow and Singpurwala (1984) and discussed in Strigini (1996). This point will be discussed in the concluding section.

The above discussion can be illustrated by the toy example. This program prevents a failure by triggering the $EMERGENCY\_STOP$ procedure each time the calculated speed for the next cycle is above *speed_limit*, which leads the system to the state $(0,0,0)$. This is a normal behavior and does not correspond to the occurrence of a failure.

A failure may occur when the program has failed to recognize that the actual speed of the train at the next step will be above the *speed_limit*. There are several possible behaviors after the occurrence of a failure.

A catastrophe may or may not occur. If a catastrophe occurs we are in the situation described at the beginning of this section, and we decide that the next state is $(0,0,0)$.

Otherwise, if the failure does not provoke a catastrophe, at the next step the speed of the train may be above or below the *speed_limit*. In the case it is above, the program may or may not recognize it, and so on. This may produce a cluster of occurrence times of failures.

Based on the above discussion, we consider that the Markov chain $(W_n)_{n\geq 0}$ is irreducible, therefore excluding any state to be absorbing. We shall also assume that the chain is aperiodic. This is a very weak restriction. For instance, it follows from the existence of a state where the chain can remain for two successive steps.

Let us indicate the starting point of the chain as a superscript. Therefore, we shall denote $(W_n^w)_{n\geq 0}$ the chain starting at the state w.

Let us call T_Λ^w the first time the Markov chain $(W_n^w)_{n\geq 0}$ reaches the failure set

$$T_\Lambda^w = \inf\{n \geq 0 : W_n^w \in \Lambda\} \ .$$

Let μ be the unique probability measure invariant with respect to the chain. In case the initial state of the chain is a random point, chosen with probability μ, we shall denote it by $(W_n^\mu)_{n\geq 0}$ and the corresponding hitting time will be denoted T_Λ^μ.

We want to obtain an exponential approximation for the law of T_Λ^μ. This is assured by the following theorems.

Theorem(*Aldous 1982*) If the departure state is chosen according to the invariant probability measure μ, then there exists a positive constant C, such that

$$\sup_{t\geq 0}\left| I\!P\left\{ \frac{T_\Lambda^\mu}{I\!E(T_\Lambda^\mu)} > t \right\} - e^{-t} \right| \leq C \frac{\tau}{I\!E(T_\Lambda^\mu)} \ ,$$

where

$$\tau^w = \min\{n \geq 0 : d_{TV}(\mathcal{L}(W_n^w), \mu)\} \leq 1/2e \text{ , for all } w \in E\} \ ,$$

d_{TV} is the total variation distance between two probability measures on E, and $\mathcal{L}(W_n^w)$ is the law of W_n^w.

The above theorem says that the exponential approximation is sharp if the time the chain takes to get close to the invariant probability measure is much shorter than the mean time for the chain in equilibrium to reach the failure set Λ. This is a reasonable property for ultra-reliable programs (Laprie and Kanoun, 1996, page 45).

The hypothesis that the departure state is chosen according to the invariant probability measure μ is not a restriction in practice. In fact, if the time the chain takes to get close to the invariant probability measure is much shorter than the time to reach the failure set, we may consider that we start observing the system when it reaches its nominal way of functioning.

More recently, Aldous and Brown (1992), obtained a sharper version of the above result for continuous time Markov chains with a finite number of states. In this version, they make the strong assumption that the process is time reversible. With this, they succeeded in replacing τ by τ_1 which is the inverse of the second eigenvalue of the process.

However, when modeling process control programs, the time reversibility assumption does not always hold and reduces significantly the applicability of this result.

A recent paper by Galves and Schmitt (1997), proves an exponential approximation for discrete time ϕ-mixing stationary stochastic processes with a finite number of states.

We recall that a stochastic process $(U_n)_{n\in\mathbb{Z}}$ is ϕ-mixing if there is a sequence $(\phi(l))_{l\geq 1}$ of positive numbers, decreasing to zero such that, for all integers n and l larger than or equal to one, we have

$$\sup_{A\in\mathcal{F}_0^n\,,\,B\in\mathcal{F}_{n+l}^\infty} \frac{|\mathbb{P}(A\cap B)-\mathbb{P}(A)\mathbb{P}(B)|}{\mathbb{P}(A)\mathbb{P}(B)} \leq \phi(l)\ ,$$

where $\mathcal{F}_r^s$ denotes the σ-algebra generated by $\{U_n; r\leq n\leq s\}$. Informally speaking the ϕ-mixing property means that any event A, which depends on the history of the process between steps 0 to n, and any event B, which depends on the history of the process after step $n+l$, have a correlation which goes to zero, when the gap l increases.

In Galves and Schmitt (1997) the exponential approximation is proved for the time of a first occurrence of a fixed sequence of states. More precisely, for a fixed integer $N\geq 1$, and a fixed choice of a sequence of states $u_1,\cdots,u_N$, let S be the first time this sequence occurs in the process. Formally

$$S=\inf\{n\geq N: U_{n-N+1}=u_1,\cdots,U_n=u_N\}\ .$$

Theorem(*Galves and Schmitt 1997*) Let $\{U_n; r\leq n\leq s\}$ be ϕ-mixing and let us suppose that the sequence $(\phi(l))_{l\geq 1}$ is summable. Then, there exist four strictly positive constants

$\Gamma_1, \Gamma_2, \beta, C$, such that for any N and any sequence of states $u_1, \cdots, u_N$, there exists $\gamma = \gamma(u_1, \cdots, u_N) \in [\Gamma_1, \Gamma_2]$, for which the following inequality holds

$$\sup_{t>0} \left| I\!P \left\{ S > \frac{t}{\gamma I\!P\{U_1 = u_1, \cdots, U_N = u_N\}} \right\} - e^{-t} \right| \leq C I\!P\{U_1 = u_1, \cdots, U_N = u_N\}^{\beta} \ .$$

It is worth noting that aperiodic and irreducible Markov chains like those used in our model are ϕ-mixing with $(\phi(l))_{l\geq 1}$ decreasing exponentially fast to zero. But the result above is applicable to a larger class of models, with long term dependencies.

This theorem is based on the control of short time returns to the rare event, after a first occurrence. Informally speaking, what is required is that once the process reaches the failure set Λ, the time it takes to leave the neighborhood of Λ is much shorter than the time it needs to get close to its invariant probability measure.

Actually, this theorem deals specifically with occurrence times of a fixed sequence of states. This enables a control of short time returns to the rare event, after a first occurrence. To apply it to the first visit to a single state, we must verify the above condition. For instance, this control is ensured if the system jumps immediately from the failure state to a fixed departure state, as discussed in the beginning of this section.

For technical details, we refer the reader to Galves and Schmitt (1997).

We conclude this section, by observing that exponential approximation remains true, and the proofs of the results are exactly the same, when the model considers the introduction of independent external signals, discussed in section 3.3.

6. Conclusions.

We have presented a mathematical justification of the widely used exponential assumption in a case where its usual justification does not hold, namely reactive programs controlling continuous physical processes.

More precisely, our result states that the occurence time of the first failure can be approximated by an exponential random variable under the condition that the program is ultra-reliable The more reliable is the program, the sharper is the approximation.

Our justification is based on one Markov model embedding both behaviors of the program and of the system under control.

Markov models of software have been used frequently. For instance, Littlewood (1975) considered structured programs with independent sub-units and a temporally homogeneous switching among them. Under the assumption that the failures of the sub-units occur according to Poisson point processes, it was shown that the overall failure process is also Poissonian.

Our model is different. Since both behaviors of the program and the system under control are taken into account, failure occurences are modeled and there is no assumption on the failure rate, or on the occurences of failure, of the various parts of the program. The price to pay is some complexity of the model.

It is worth remarking that the problem of studying the behavior of ultra- reliable system before a first failure occurs is reminiscent of the metastable behavior of thermodynamical systems (cf. Cassandro, Galves, Olivieri and Vares, 1984). In both cases, what is observed is an illusory stable evolution, until suddenly the systems falls in a completely different mode of behavior.

It is interesting to note that it was possible to state a general result for the ocurrence of a first failure, but that consecutive failures turned out to be much more difficult to study. This is due to the fact that the behavior after a failure is very dependent of the physical characteristics of the system under control, as it has been pointed out in (Strigini, 1996).

In practice, if the execution does not end after the first failure, it may be the case that the internal state of the program is corrupted, and this increases the probability of subsequent failures. Moreover, depending on the kind of controlled system, different situations may happen. Strigini gives some examples of such situations

- some kinds of system may proceed without any problem after an isolated failure, but will definitely stop after a sequence of failures longer than a certain limit;

- failures can have different levels of severity, i. e. there are benign failures and catastrophic failures, sequences of benign failures leading to a stop of the system, as above;

- failures may carry some cost, and a too expansive execution is of no interest, etc.

This must be taken into account in the Markov model for stating any precise result on the occurences of failures after the first one.

A last remark is that we have shown exponential approximation for a kind of transient failures of software, namely those due the conjunction of unexpected environment conditions and measure inaccuracies. This is different from hardware, where it has been observed that transient failures follow Weibull distributions rather than exponential ones (Castillo, McConnel and Sievorek, 1982).

Acknowledgments. This work was partially supported by CNPq (grant 301301/79), FAPESP (*Projeto Temático* 95/0790-1), Pronex (grant 41.96.0923.00), and the *DEVA Long Term Research Project 20072* of the ESPRIT program.

References.

D. J. Aldous (1982) Markov chains with almost exponential hitting times. *Stochastic Process. Appl.* **13** 305-310.

D. J. Aldous, M. Brown (1992) Inequalities for rare events in time-reversible Markov chains I *in* M. Shaked and Y. L. Tong *Stochastic Inequalities* IMS Lecture Notes, vol 22, 1-16.

D. J. Aldous, M. Brown (1993) Inequalities for rare events in time-reversible Markov chains II. *Stochastic Processes Appl.* **44** 15-25.

R. Bellman and T. E. Harris (1951) Recurrence times for the Ehrenfest model. *Pacific J. Math.* **1** 179.

R. W. Butler, and G. B. Finelli (1993) The infeasibility of quantifying the reliability of life-critical real-time software. *IEEE Trans. on Software Engineering*, **19-1**, 3-12.

M. Cassandro, A. Galves, E. Olivieri and M. E. Vares (1984) Metastable behavior of stochastic dynamics: a pathwise approach. *J. Stat. Phys.* **35**:603-633.

X. Castillo, S. R. McConnel and D. P. Sievorek (1982) Derivation and calibration of a transient error reliability model. *IEEE Trans. on Computers* **C-31-7**, 658-671.

R. Cogburn (1985) On the distribution of first passage and return times for small sets. *Ann. Probab.* **13**, 1219-1223.

P. Collet and A. Galves (1993) Statistics of close visits to the indifferent fixed point of an interval map. *J. Stat.Phys.* **72**, 459-478.

P. Collet and A. Galves (1995) *Asymptotic distribution of entrance times for expanding maps of the interval* in *Dynamical Systems and Applications*, WSSIAA **4**, 139-152.

P. Collet, A. Galves and B. Schmitt (1992) Unpredictability of the occurrence time of a long laminar period in a model of temporal intermittency. *Ann. Inst. H. Poincaré, Section A,* **57**, 3:319-331.

L. H. Crow and N. D. Singpurwalla (1984) An empirically developed Fourier serie model for describing software failures *IEEE Trans. Reliability* **33,** 176-183.

P. Dauchy , M. C. Gaudel, and B. Marre (1993) Using algebraic specifications in software testing: a case study on the software of an automatic subway. *Journal of Systems and Software*, **21-3**, 229-244.

W. Doeblin (1940) Remarques sur la théorie métrique des fractions continues. *Compositio*

Math. **7**, 353-371.

P. A. Ferrari, A. Galves and C. Landim (1994) Exponential waiting time for a big gap in a one dimensional zero range process. *Ann. Probab.* **22**, 284-288.

A. Galves and B. Schmitt (1990) Occurrence times of rare events for mixing dynamical systems. *Ann. Inst. H. Poincaré* **52**, 267-281.

A. Galves and B. Schmitt (1997) Inequalities for hitting times in mixing dynamical systems. *Random and Comput. Dyn.*, to appear. Preprint available as http://www.ma.utexas.edu/mp_ arc/c/97/97-128.ps.gz.

T. E. Harris (1953) First passage and recurrence distributions. *Trans. Amer. Math. Soc.* **73**, 471-486.

J. Keilson (1979) *Markov Chain Models. Rarity and Exponentiality.* Springer-Verlag.

C. Kipnis and C. Newman (1985) The metastable behavior of infrequently observed, weakly random, one-dimensional diffusion processes *SIAM J. Appl. Math.* **45**, 972.

D. V. Korolyuk and D. S. Sil'vestrov (1984) Entry times into asymptotically receding domains for ergodic Markov chains. *Theory Prob. Appl.* **19**, 432-442.

J.C. Laprie (1984) Dependability evaluation of software systems in operation. *IEEE Trans. on Soft. Eng.* **10-6**, 701-714.

J.C. Laprie and K. Kanoun (1996) Software reliability and system reliability. Chapter 2 of (Lyu, 1996)

J. L. Lebowitz and R. H. Schonmann (1987) On the asymptotics of occurrence times of rare events for stochastic spin systems. *J. Stat. Phys.* **48**, 727-751.

B. Littlewood and J. Verral (1973) A Bayesian reliability growth model for computer software. *Jal of Royal Stat. Soc.*, **C-22-3**, 332-346.

B. Littlewood (1975) A reliability model for systems with Markov structure. *Appl. Statist.* **24**, 172-177.

B. Littlewood and L. Strigini (1993) Validation of ultra-high dependability for software-based systems. *Communications of the A. C. M.* **36-11**, 69-80.

M. R. Lyu ed. (1996) *Handbook of software reliability engineering*, IEEE Computer Society Press and McGraw-Hill.

J. D. Musa and K. Okumoto (1984) A logarithmic Poisson execution time model for software reliability measurement, *7th IEEE Int. Conf. on Soft. Eng.*, 230-238.

J. D. Musa, A. Iannino and K. Okumoto (1987) *Software reliability - measurement, prediction, application*, McGraw-Hill, New-York.

E. Olivieri and E. Scoppola (1995) Markov chains with exponentially small transition probabilities: first exit problem from a general domain. I. The reversible case. *J. Stat.*

Phys. **79**, 613-647.

E. Olivieri and E. Scoppola (1996) Markov chains with exponentially small transition probabilities: first exit problem from a general domain. II. The general case. *J. Stat. Phys.* **84**, 987-1041.

L. Strigini (1996) On testing process control software for reliability assesment: the effects of correlation between successive failures. *Software Testing, Verification and Reliability* **6**, 33-48.

M. Xie (1993) Software reliability models - a selected bibliography. *Journal of Software Testing, Verification and Reliability* **3**, 3-28.

A. Galves
Instituto de Matemática e Estatística
Universidade de São Paulo
BP 66 281
05315-970 São Paulo SP
Brasil
galves@ime.usp.br

M. C. Gaudel
Laboratoire de Recherche en Informatique
Université de Paris-Sud et CNRS
Bâtiment 490
91405 Orsay Cedex
France
mcg@lri.lri.fr