Reliability Growth of Fault-Tolerant Software

Karama Kanoun
LAAS-CNRS, Toulouse
Mohamed Kaâniche
LAAS-CNRS, Toulouse
Christian Béounes
LAAS-CNRS & INRIA, Toulouse
Jean-Claude Laprie
LAAS-CNRS, Toulouse
Jean Arlat
LAAS-CNRS, Toulouse

Key Words — **Fault-tolerant software, Reliability growth, Reliability modeling, Reliability evaluation, Corrective maintenance.**

Reader Aids —
General purpose: Investigate fault-tolerant software techniques
Special math needed for explanations: Probability theory and Markov chains
Special math need to use results: None
Results useful to: Designers of fault-tolerant software

Summary & Conclusions — **Fault-tolerant software approaches have given rise to numerous reliability models. However, all these models assume stable reliability, *ie*, they do not consider reliability growth resulting from progressive removal of design-induced faults. This paper —**

- **addresses an issue which has not hitherto been treated;**
- **is aimed at modeling and estimating the influence of reliability growth of the fault-tolerant software components on the reliability of the software system in operation.**

Two fault-tolerant software techniques are investigated: recovery block and N-version programming. For each, the stable reliability model is transformed into a model that considers reliability growth via the transformation approach based on the hyper-exponential model. Analytic and numeric processing of the transformed models identify the influence of fault removal on the reliability of the fault-tolerant software approaches.

The modeling approach is based on the transformation of a Markov chain of the fault-tolerant software system in stable reliability into another, modified Markov chain which enables reliability growth to be considered. This approach has allowed reliability growth relative to the classes of faults (independent, related) affecting fault-tolerant software to be identified and evaluated. The evaluations apply to systems of short successive mission durations with respect to the system life-time. Using generalized stochastic Petri nets to model the fault-tolerant software systems allows for an automatic application of the transformation technique.

Analytic expressions are derived only to analyze explicitly the impact of fault-removal of each class. In practice, reliability measures can be directly evaluated by available tools for numerical processing of the Markov chains.

Even though this work is a first attempt, the results are important since they show the influence of reliability growth on the reliability of fault-tolerant software systems. These results: a) confirm, from the reliability growth perspective, the importance of the faults whose occurrence can lead to common-mode failures, *eg*, *decider* faults and *related* faults and b) enable the impact of these faults to be quantified.

1. INTRODUCTION

Numerous papers have been devoted to the modeling and evaluation of fault-tolerant software reliability. They may be put into two major classes:

- Modeling software diversity: Based on the detailed analysis of the dependencies in diversified software [4,9,23,28].
- Modeling fault-tolerant software system behavior: Aimed primarily to evaluate some dependability measures for specific types of software systems [1,8,11,12,16,29].

All the published work assumes stable reliability, *ie*, it does not consider the impact, on software reliability, of reliability growth due to corrective maintenance. Nevertheless, reliability growth is real as evidenced by the failure data collected on real software systems during their life cycle, including operational life [14,21].

The modifications due to the progressive removal of residual design faults from the software components cause their reliability to grow, which in turn causes the reliability of the fault-tolerant software to grow. The reliability of fault-tolerant software can grow due to the removal of faults in the software components without having failed in operation. For instance, activation of a fault which is tolerated does not lead to system failure and reliability of the software system grows due to the removal of this type of fault.

The extent to which software reliability is improved via corrective maintenance is related to the nature of the activated fault and the consequence of the failure:

- Some faults are more difficult to identify than others; thus they may be, a) removed long after their activation, leading to slow reliability growth, or b) not removed at all.
- Faults leading to critical failures are likely to be removed very quickly, whereas those leading to non-critical failures might be removed later (*eg*, with the introduction of a new software version).

For multi-component software, reliability growth can be different for the various components, *ie*, some components might be more failure-prone and undergo several corrective actions resulting in important reliability growth, while other components might be less activated or more reliable and thus exhibit less reliability growth.

As a result, software reliability is related to the:

- reliability of its components at a given time
- evolution of the reliability of these components.

Since the reliability growth of the components is not identical, one has to determine the influence of component-reliability growth on the software-reliability growth.

Modeling fault-tolerant software without considering the impact of cumulative corrective-maintenance actions, leads to results that are relevant only for a given period of time over which the software reliability is stable; these results might not represent the software evolution in a maintenance environment. This paper models & estimates the influence of component-reliability growth on the reliability of a fault-tolerant software system. It uses a general approach to model the reliability of fault-tolerant software, and is based on our previous work [1,18]. The models of two software-fault tolerance approaches are established; they enable sensitivity analyses for the influence of several factors on the reliability growth of the fault-tolerant software system. The software is assumed to be in operational life with corrective maintenance actions.

These fault-tolerant software techniques correspond to the two best-documented techniques for tolerating software faults:

- recovery block [27]
- N-version programming [6].

Section 2 presents the modeling approach, which is then applied in the subsequent sections to RB & NVP. Section 3 details the steps of RB modeling and the evaluation results. Section 4 summarizes the main results for NVP. Section 5 compares RB & NVP.

Acronyms

GSPN	generalized stochastic Petri nets
NHPP	Non-Homogeneous Poisson Process
NVP	N-version programming
RB	recovery block.

Notation

$\bar{}$	implies 1's complement, *eg*, $\bar{\omega} \equiv 1-\omega$
$\rightarrow$	stable reliability
$\nearrow$	reliability growth.

Nomenclature (Failure, Error, Fault) [20]

- A *system failure* occurs when the delivered service no longer complies with the specification. The specification is an agreed description of the system function and/or service.
- An *error* is that part of the system state which is liable to lead to subsequent failure; an error affecting the service indicates that a failure occurs or has occurred.
- A *fault* is the adjudged or hypothesized cause of an error.◀

Standard notation & nomenclature are given in "Information for Readers & Authors" at the rear of each issue.

2. MODELING & EVALUATION APPROACH

Our objective is to model and evaluate the dependability of fault-tolerant software systems during their operational life, taking into account the reliability growth of their components due to design-fault removal in order to highlight the influence of corrective maintenance. So far, two kinds of software reliability models have been developed:

- Models dedicated to reliability growth of software systems — dealing with the reliability growth of 1-component software, the black-box approach, *eg*, [23,31]
- Models taking into account the software structure — dealing mainly with stable reliability [16,22,30].

Our approach to model the reliability of fault-tolerant software systems is more general than previous ones as it combines information on the software structure and on the reliability growth of the components. It is based on the hyperexponential reliability-growth model [18]. An important property of the hyperexponential model is the transformation of a stable reliability Markov model into another Markov model that can handle reliability growth [18]. This is particularly relevant here as it allows the reliability growth of a fault-tolerant software system to be modeled from the reliability growth of its components. The transformation is based on the interpretation of the hyperexponential model as a Markov model.

To our knowledge, our approach is the only one allowing easy modeling of fault-tolerant software systems; it is systematic as opposed to the ad hoc methods that: a) use reliability-growth models and then consider the structure of the software (which would require restrictive independence assumptions), or b) extend structural models to account for reliability growth which, again, is feasible only for specific cases.

The main modeling difficulty arises from the statistical dependence among the components — due principally to common-mode failures of the components. Our approach overcomes this difficulty.

Sections 2.1 - 2.3 present: a) basic results of the transformation approach, b) our assumptions for modeling fault-tolerant software, and c) definition of the dependability measures that will be evaluated.

2.1. *Hyperexponential Model and Transformation Approach* [18][1]

The hyperexponential model is a NHPP reliability-growth model with intensity function[2]:

[1]The hyperexponential model has already been used as a reliability-growth model to evaluate the reliability of software systems; its ability to fit data collected on real systems has been shown, *eg*, [13,15].
[2]It is also called failure intensity in our context.

$$h_{HE}(t) = \frac{\omega\ \zeta_{sup}\ \exp(-\zeta_{sup}t) + \bar{\omega}\ \zeta_{inf}\ \exp(-\zeta_{inf}t)}{\omega\ \exp(-\zeta_{sup}t) + \bar{\omega}\ \exp(-\zeta_{inf}t)},$$

$$0 \le \omega \le 1. \qquad (1)$$

$h_{HE}(t)$ is non-increasing with time when $0 < \omega < 1$, from $h(0) = \omega\zeta_{sup} + \bar{\omega}\zeta_{inf}$ to $h(\infty) = \zeta_{inf}$ (see figure 1).

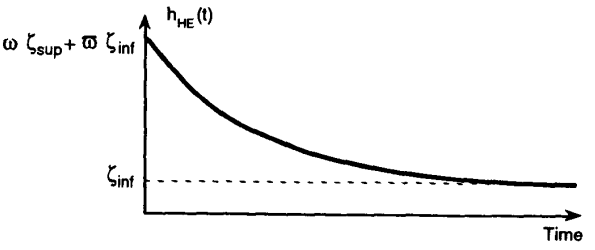


Figure 1. Failure Intensity of the Hyperexponential Model

The transformation approach is based on interpreting the hyperexponential model as a Markov model corresponding to a 2-stage hyperexponential Cox law — as illustrated by a simple example.

Example

A software system has 2 components, A & B. Component A is executed first; component B is invoked upon failure of A. Component B failure leads to system failure. The solicitation rate is constant; its reciprocal is the mean idle time.

Notation for Example

I, W	[idle, working] software
σ	software system solicitation rate
X	represents component A or B
λ_X	failure rate of component X
γ_X	execution rate of component X
ω_X	initial probabilities defined by the hyperexponential model.

Figure 2a gives the Markov chain in stable reliability. Figure 2b shows the transformed Markov chain accounting for component A reliability growth. Figure 2d shows the transformed Markov chain accounting for reliability growth of both components A & B.

Consider the model of figure 2b. The transformed Markov chain enables the reliability growth to be simulated, via the modeling of the non-stationary process induced by reliability growth [18]. To avoid misinterpretation, consider 2 points:

- The interpretation of figure 2b as representing a behavior such as reaching stationarity after leaving transient states I_1 and W_{A1} is misleading, since such interpretation does not account for the fact that initial probabilities of states I_1 & I_2 are not 0 or 1.
- The Markov chain of figure 2b is different from the Markov chain which would be transformed via the conventional device of stages to account for a distribution of time-to-failure with a decreasing hazard rate [7]. Figure 2c gives the latter and corresponds to the following pdf for component A:

$$\omega_{A1}\zeta_{A1}\exp(-\zeta_{A1}t) + \omega_{A2}\zeta_{A2}\exp(-\zeta_{A2}t),\ 0 \le \omega_{A1},\ \omega_{A2} \le 1,\ \omega_{A1} + \omega_{A2} = 1.$$

◄

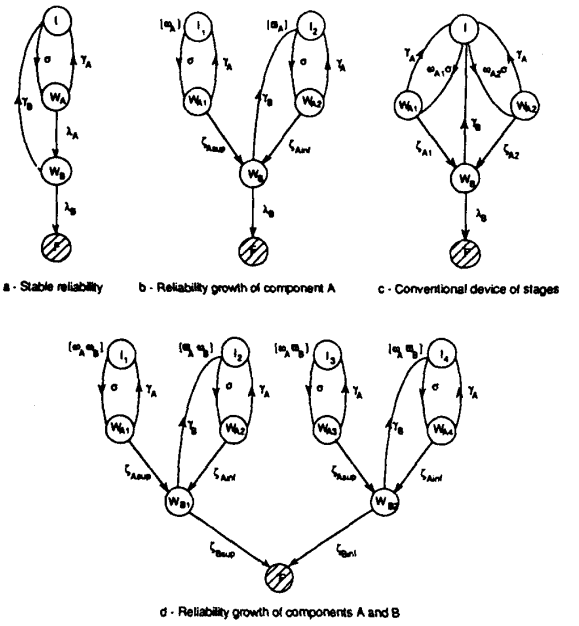


Figure 2. Markov Chains for a 2-Component Software System

For multi-component systems whose component behavior can be s-dependent, the transformation is conveniently implemented using GSPN [24]. The model construction steps are:

1. Build the GSPN model for the system behavior in stable reliability.
2. Transform the GSPN model from step 1 into a GSPN model in reliability growth, according to the GSPN representation of the hyperexponential model. For the example in section 2.1 — accounting only for component A reliability growth — steps 1 & 2 are given in figure 3.
3. Derive the reachability graph of the reliability growth GSPN, which is the transformed Markov chain accounting for reliability growth.

Notation for GSPN Models

□ x	instantaneous transition with probability x
■ x	timed transition with rate x
①	place with initial marking equal to 1
—o—	inhibitor arc

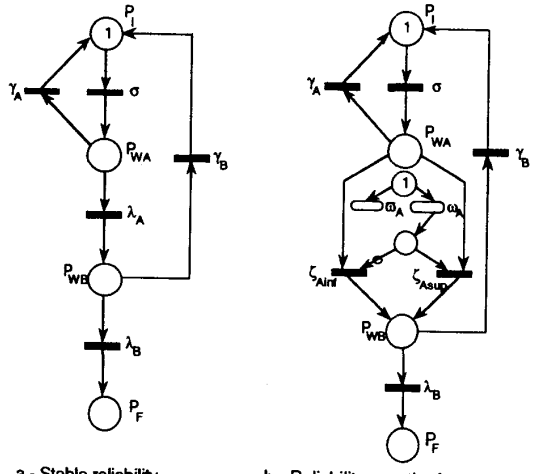


Figure 3. GSPN for a 2-Component Software System

2.2. *Assumptions for Fault-Tolerant Software Modeling*

We consider fault-tolerant architectures aimed at tolerating a single software fault. Their behavior in stable reliability can be modeled as a Markov chain [16,22]. In [1], we modeled the fault-tolerance in stable reliability via discrete-time Markov chains, the relation to continuous-time being provided by the software solicitation rate. Here, in order to ease the application of our continuous-time model of reliability growth, we directly derive continuous-time Markov chains. Following the transformation approach in section 2.1, the behavior of the fault-tolerant architectures considering reliability growth is modeled via Markov chains also.

Two classes of faults are considered: *related* & *independent*.

- *Related* faults result from: a) a specification fault common to the alternates in RB (versions, in NVP) or to the alternates and the acceptance test in RB (versions and the decider, in NVP), or b) from *s*-dependencies in the separate design and implementations of the various components.
- *Independent* faults are non *related* faults.

Related faults lead to common-mode failures, whereas *independent* faults usually cause distinct errors and separate failures. To construct the model, we do not detail the sources of *related* faults[3], rather we focus on the influence of reliability growth.

2.3. *Dependability Measures*

When considering an operating system, the interest is in failure-free time intervals whose starting instants are not necessarily conditioned on system failures, *ie*, they can occur at any time. In this case, the user is interested in the reliability over a given time interval, independent of the number of failures.

[3]The models in stable reliability are similar to the simplified models of [1], rather than to the detailed models therein.

Notation

t	age of software (system put into operation at $t=0$)
u	desired failure-free time interval $[t,\ t+u]$
$h(u)$	failure intensity
$\lambda(t)$	system failure rate
$R(t,t+u)$	system reliability over u; mission reliability
$r(t)$	reliability evaluated from the transformed Markov chain for modeling reliability growth
$o(\phi)$	any function such that $\text{limit}_{\phi\to 0}\{o(\phi)/\phi\}=0$.

Assumptions

1. For all mission reliability calculations, $u << t$.
2. The failure process is modeled by an NHPP whose failure intensity is continuously decreasing.
3. It is not considered whether failures are caused by detected or undetected errors. (Removing this assumption would be straightforward[4]). ◄

$$R(t,t+u) = r(t+u)/r(t). \tag{2}$$

Due to space limitation, the heuristic justification of this relation is outlined:

- For stable reliability, for $u << t$, a conventional result of stationary renewal theory is [10: p 105]:

$$R(t,t+u) = 1 - h(t)\cdot u + o(u) \tag{3}$$

  This result holds for reliability growth, either for NHPPs [2: p 30], or more generally for non-stationary processes [19].
- The (conventional) conditional reliability is:

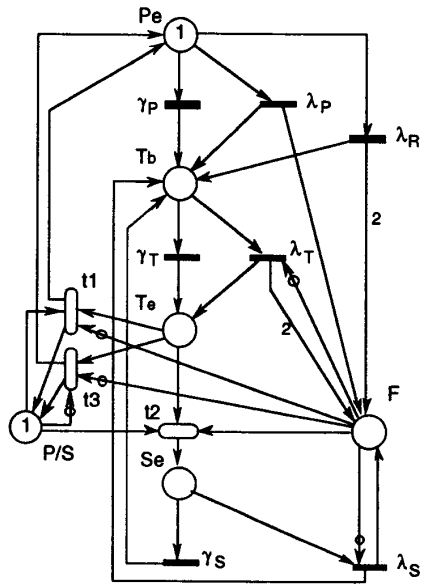
$$r(t+u)/r(t) = \exp\left[-\int_t^{t+u} \lambda(\tau)\, d\tau,\right.$$

$$r(t+u)/r(t) = 1 - \lambda(t)\cdot u + o(u), \text{ for } u << t. \tag{4}$$

- Consider again the failure process of the fault-tolerant software. The system failure rate is piecewise decreasing: transition from one step to the next is due to the modification of software components; the software is generally used several times at each step. The failure rate of the NHPP cannot be distinguished from its failure intensity [26] which can be viewed as the result of smoothing the decreasing piecewise failure rate.[5] Eq (2) holds due to: a) the NHPP assumption, and b) the fact that the failure rate evaluated from the transformed Markov chain can be viewed as the failure intensity of the process simulated by this chain. ◄

[4]The distinction between detected & undetected errors has already been considered [1] for fault-tolerant software systems in stable reliability.

[5]It is impossible to distinguish between certain families of i.i.d. order statistic models and related families of NHPP models when a single realization of the process is observed [25].



Place		Transition	
Name	Meaning	Name	Meaning
Pe	Primary is executing	γ_P	Correct execution of P
		λ_P	Incorrect execution of P due to P failure
		λ_R	Incorrect execution of P due to related faults
		γ_T	Correct execution of T
Tb	Acceptance test is executing	λ_T	Incorrect execution of T due to T failure
Te	Deciding what to execute next	t_1, t_3	Correct execution of RB
F	A failure has occurred	t_2	Start of execution of S
P/S	T has been executed after P or S		
Se	Secondary is executing	γ_S	Correct execution of S
		λ_S	Incorrect execution of S due to S failure

Figure 4. GSPN for RB in stable reliability

The influence of reliability growth on a fault-tolerant software is assessed by comparing $R(t,t+u)$ evaluated from the transformed model with $R^*(t,t+u)$ evaluated from the stable reliability Markov chain (see section 2.1).

We consider a RB made up of two alternate components (a primary and a secondary) and an acceptance test.

3. RB MODELING & EVALUATION

Notation

P, S	[primary, secondary] component
T	acceptance test
γ_X	execution rate of X, $X \in \{P,S,T\}$
λ_X, h_X	failure [rate, intensity] associated with the *independent* faults of X, $X \in \{P,S,T\}$
λ_R, h_R	failure [rate, intensity] corresponding to the *related* faults
k_X	$h_X(0)/h_X(\infty)$, $X \in \{P,S,T,R\}$: reliability growth factor
η_X	$h_X(\infty)/\gamma_X$.

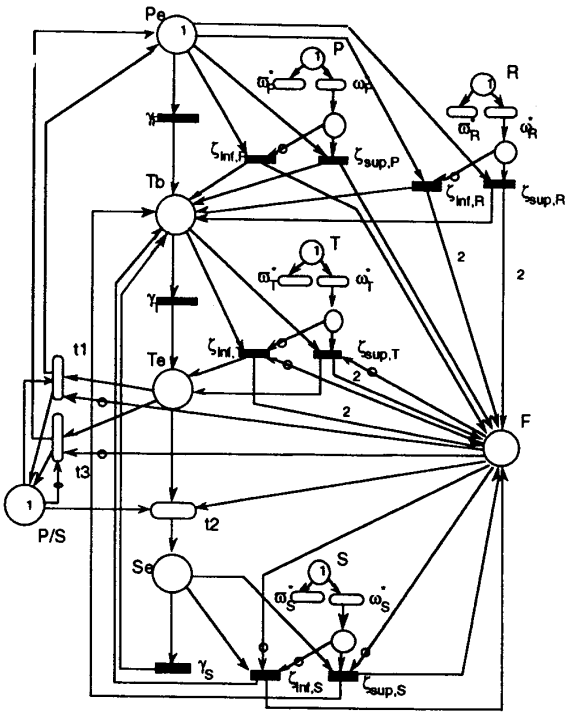
3.1. *Model Derivation*

Additional Assumptions (see section 2.2)

1. Only one type of fault (*independent* or *related*) can occur during a single execution of the whole RB.
2. No error compensation takes place. ◄

Figure 4 shows the GSPN model for RB in stable reliability as well as the meaning of the model places and transitions.

Figure 5 shows the GSPN model, for RB in reliability growth, resulting from the application of the transformation approach to the model of figure 4.



$\omega_X^* = \omega_X/[\Sigma\, m(Y)]$, $\bar{\omega}_X^* = \bar{\omega}_X/[\Sigma\, m(Y)]$
$X,Y \in \{P,T,S,R\}$
$m(Y)$ implies: marking of place Y

Figure 5. GSPN for RB in Reliability Growth

- $\{\omega_X, \zeta_{\sup,X}, \zeta_{\inf,X}\}$, for $X = P,S,T$, denote the hyperexponential model parameters characterizing RB reliability growth due to the removal of the *independent* faults of X
- $\{\omega_R, \zeta_{\sup,R}, \zeta_{\inf,R}\}$ denote the hyperexponential model parameters characterizing RB reliability growth due to removal of the *related* faults.

Figure 5 incorporates the basic model of figure 3.

The corresponding Markov chains for stable reliability and reliability growth are given in figures 6 & 7 respectively. Figure 7 shows a general model allowing reliability growth of all failure processes to be taken into account. Reliability growth of one or more processes can be modeled from figure 7, making $\omega_X = 0$ and $\zeta_{\inf,X} = \lambda_X$ for the other processes. For example, reliability growth due to the elimination of the *independent* faults of P only is modeled using $\omega_X = 0$ and $\zeta_{\inf,X} = \lambda_X$, $X \in \{S,T,R\}$; the model in stable reliability (figure 6) can be deduced from figure 7; thus:

$$\omega_X = 0 \text{ and } \zeta_{\inf,X} = \lambda_X,\ X \in \{P,S,T,R\}.$$

The corresponding Markov chains for stable reliability and reliability growth are given in figures 6 & 7 respectively. Figure

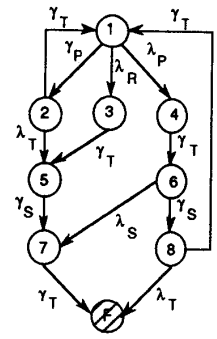


Figure 6. Markov Chain for RB in Stable Reliability

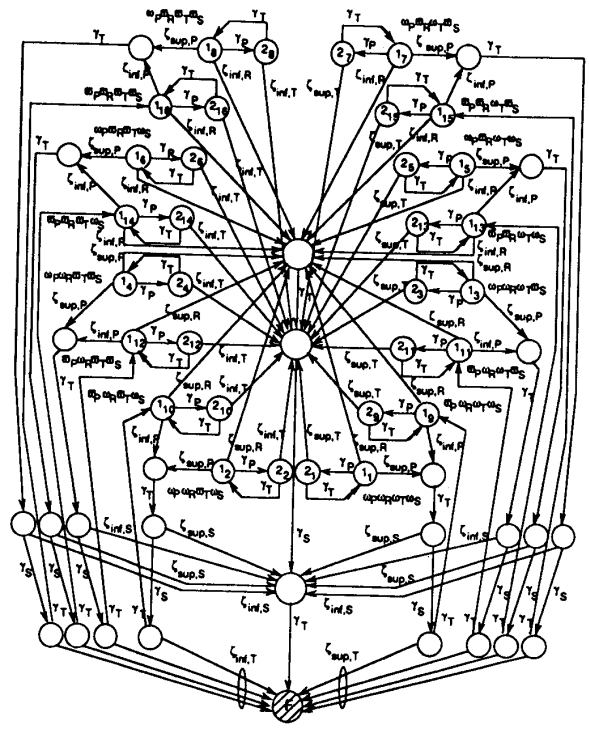


Figure 7. Markov Chain for RB in Reliability Growth

7 shows a general model allowing reliability growth of all failure processes to be taken into account. Reliability growth of one or more processes can be modeled from figure 7, making $\omega_X = 0$ and $\zeta_{\inf,X} = \lambda_X$ for the other processes. For example, reliability growth due to the elimination of the *independent* faults of P only is modeled using $\omega_X = 0$ and $\zeta_{\inf,X} = \lambda_X$, $X \in \{S,T,R\}$; the model in stable reliability (figure 6) can be deduced from figure 7; thus:

$$\omega_X = 0 \text{ and } \zeta_{\inf,X} = \lambda_X,\ X \in \{P,S,T,R\}.$$

3.2. *Model Processing*

RB in reliability growth and in stable reliability are examined and compared.

3.2.1 RB in reliability growth

As indicated in section 2, reliability of the RB is evaluated by processing the transformed Markov chain in figure 7. Numerical solutions can be obtained using an evaluation tool such as SURF [3]. Nevertheless, to analyze the influence on the RB reliability of each failure process, when dealing with the reliability growth aspects, analytic expressions are derived before presenting numerical results.

The transformed Markov chain in figure 7 is quite complex because its state space is so large; it is therefore impractical to obtain an exact solution of the reliability measure. However, the model of figure 7 is stiff, since the execution rates γ_X are several orders of magnitude higher than the failure rates $\zeta_{\sup,X}$. The aggregation technique in [5] for systematically converting a stiff Markov chain into a non-stiff Markov chain with a smaller state space can be used; this technique gives good approximate solutions and, since the state space is reduced, analytic expressions can be derived.

In the aggregation algorithm, the transient states of the Markov chain are partitioned into:

- fast states for which there is at least one high-rate output transition
- slow states for which there are only low-rate output transitions.

The algorithm proceeds by first analyzing the transition matrix to determine states that cause stiffness (fast states). Then it separates them into near ergodic (recurrent non-null) subsets and/or a transient subset.

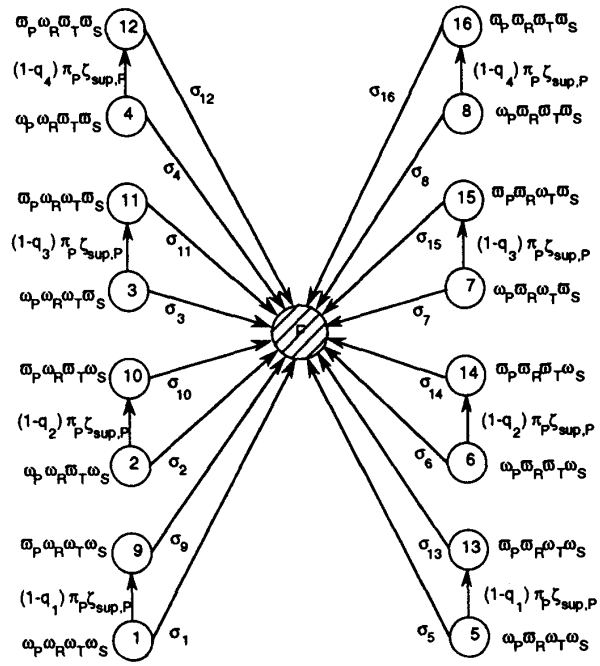
A steady-state analysis of each recurrent subset is carried out and each such subset is replaced by a slow state. The fast transient subset is analyzed to determine the conditional branching probabilities and the subset is replaced by a probabilistic switch.

Application of the aggregation technique to the model of figure 7 allows reduction to the non-stiff model in figure 8. Each state subset $\{1_i, 2_i\}$, $i \in \{1,\ldots,16\}$ of figure 7 forms a fast recurrent subset and is replaced by a slow state i in figure 8.

The successive application of the transformation approach and the aggregation technique appreciably reduces the number of states, *viz*, from 60 (figure 7) to 17 (figure 8), at the expense of more complex expressions for the inter-state transition rates.

Analytic processing of the model in figure 8 leads to:

$$r_{RB}(\tau) = [\omega_R \cdot \exp(-\pi_P \zeta_{\sup,R}\tau) + \bar{\omega}_R \cdot \exp(-\pi_P \zeta_{\inf,R}\tau)] \cdot [\exp(-\pi_T\, \zeta_{\sup,T}\tau)(\phi_1(\tau)+\phi_3(\tau)) + \exp(-\pi_T\, \zeta_{\inf,T}\tau)(\phi_2(\tau)+\phi_4(\tau))], \quad (5)$$



$\sigma_1 = (q_1\, \zeta_{\sup,P} + \zeta_{\sup,R})\, \pi_P + \zeta_{\sup,T}\, \pi_T$
$\sigma_2 = (q_2\, \zeta_{\sup,P} + \zeta_{\sup,R})\, \pi_P + \zeta_{\inf,T}\, \pi_T$
$\sigma_3 = (q_3\, \zeta_{\sup,P} + \zeta_{\sup,R})\, \pi_P + \zeta_{\sup,T}\, \pi_T$
$\sigma_4 = (q_4\, \zeta_{\sup,P} + \zeta_{\sup,R})\, \pi_P + \zeta_{\inf,T}\, \pi_T$
$\sigma_5 = \sigma_1 + (\zeta_{\inf,R} - \zeta_{\sup,R})\, \pi_P$
$\sigma_6 = \sigma_2 + (\zeta_{\inf,R} - \zeta_{\sup,R})\, \pi_P$
$\sigma_7 = \sigma_3 + (\zeta_{\inf,R} - \zeta_{\sup,R})\, \pi_P$
$\sigma_8 = \sigma_4 + (\zeta_{\inf,R} - \zeta_{\sup,R})\, \pi_P$
$\sigma_9 = \sigma_1 + (\zeta_{\inf,P} - \zeta_{\sup,P})\, \pi_P$
$\sigma_{10} = \sigma_2 + (\zeta_{\inf,P} - \zeta_{\sup,P})\, \pi_P$
$\sigma_{11} = \sigma_3 + (\zeta_{\inf,P} - \zeta_{\sup,P})\, \pi_P$
$\sigma_{12} = \sigma_4 + (\zeta_{\inf,P} - \zeta_{\sup,P})\, \pi_P$
$\sigma_{13} = \sigma_9 + (\zeta_{\inf,R} - \zeta_{\sup,R})\, \pi_P$
$\sigma_{14} = \sigma_{10} + (\zeta_{\inf,R} - \zeta_{\sup,R})\, \pi_P$
$\sigma_{15} = \sigma_{11} + (\zeta_{\inf,R} - \zeta_{\sup,R})\, \pi_P$
$\sigma_{16} = \sigma_{12} + (\zeta_{\inf,R} - \zeta_{\sup,R})\, \pi_P$
$\pi_P \equiv \gamma T/(\gamma T+\gamma P)$ = proportion of time spent in P when no failure occurs.
$\pi_T \equiv 1-\pi_P$ = proportion of time spent in T when no failure occurs.
$Q(a,b) \equiv [1 + \gamma_S/\zeta_{a,S}]^{-1} + [(1 + \zeta_{a,S}/\gamma_S)(1 + \gamma_T/\zeta_{b,T})]^{-1}$
$q_1 \equiv Q(\sup,\sup)$, $q_2 \equiv Q(\sup,\inf)$
$q_3 \equiv Q(\inf,\sup)$, $q_4 \equiv Q(\inf,\inf)$

Figure 8. Aggregated Markov Model for RB in Reliability Growth

$$\phi_i(\tau) \equiv \delta_i[A_i \cdot \exp(-\pi_P \zeta_{\sup,P}\tau) + \bar{A}_i \cdot \exp(-q_i \pi_P \zeta_{\inf,P}\tau)];\ i \in \{1,\ldots,4\}$$

$\delta_1 \equiv \omega_S\omega_T$; $\delta_2 \equiv \omega_S\bar{\omega}_T$; $\delta_3 \equiv \bar{\omega}_S\omega_T$; $\delta_4 \equiv \bar{\omega}_S\bar{\omega}_T$;

$$A_i \equiv \omega_P(1 - \xi_P)/(1 - \xi_P/q_i);$$

$$\xi_X \equiv \zeta_{\sup,X}/\zeta_{\inf,X}$$

$$r_{\mathrm{RB}}(\tau) = \sum_{i=1}^{16} \alpha_i \exp(-\beta_i \tau); \quad \sum_{i=1}^{16} \alpha_i = 1; \tag{6}$$

where the α_i, β_i are suitably defined.

As indicated in section 2, our measure of interest is:

$$R_{\mathrm{RB}}(t,t+u) = r_{\mathrm{RB}}(t+u)/r_{\mathrm{RB}}(t).$$

$$= \sum_{i=1}^{16} \alpha_i \cdot \exp(-\beta_i(t+u)) / \sum_{i=1}^{16} \alpha_i \cdot \exp(-\beta_i t) \tag{7}$$

$$\approx 1 - h_{\mathrm{RB}}(t)u, \text{ for } u << t \tag{8}$$

$$h_{\mathrm{RB}}(t) = \sum_{i=1}^{16} \alpha_i\beta_i \cdot \exp(-\beta_i(t+u)) / \sum_{i=1}^{16} \alpha_i \cdot \exp(-\beta_i t) \tag{9}$$

$h_{\mathrm{RB}}(t)$ is non-increasing with time and asymptotically reaches a constant, which implies stable reliability.

$$k_X = \omega\xi_X + \bar{\omega}. \tag{10}$$

Use (9) & (10) and the notation in figure 8; then —

$$h_{\mathrm{RB}}(0) = \pi_P k_R h_R(\infty) + \pi_T k_T h_T(\infty)$$

$$+ \pi_P k_P h_P(\infty) \sum_{i=1}^{4} \delta_i q_i; \tag{11}$$

$$h_{\mathrm{RB}}(\infty) = \pi_P h_R(\infty) + \pi_T h_T(\infty) + \pi_P h_P(\infty) q_4. \tag{12}$$

The γ_X are several orders of magnitude higher than the $\zeta_{\sup,X}$; thus (11) & (12) can be simplified:

$$h_{\mathrm{RB}}(0) = \pi_P k_R h_R(\infty) + \pi_T k_T h_T(\infty)$$

$$+ \pi_P k_P h_P(\infty)(k_S\eta_S + k_T\eta_T), \tag{13}$$

$$h_{\mathrm{RB}}(\infty) = \pi_P h_R(\infty) + \pi_T h_T(\infty) + \pi_P h_P(\infty)(\eta_S + \eta_T). \tag{14}$$

Eq (13) & (14) are highly relevant since they enable the important parameters to be identified, and give an insight into the relationships between the RB failure intensity and those corresponding to the activation of faults within its components when the extreme situations (maximum and minimum unreliability) are considered. The removal of *independent* faults from the acceptance test and of *related* faults have the most important impact. In fact, in (13) for instance, the activation of *independent* faults in P & S affect RB reliability only if:

$$(\pi_P k_P k_S h_P(\infty)\eta_S \text{ or } \pi_P k_P k_T h_P(\infty)\eta_T)$$

$$\sim (\pi_P k_R h_R(\infty) \text{ or } \pi_T k_T h_T(\infty));$$

where $\sim$ implies "same order of magnitude", *viz*, within a factor of 10.

In practice, this seldom occurs because γ_S & γ_T are generally many orders of magnitude higher than $h_P(\infty)$ & $h_S(\infty)$. It follows that the maintenance effort devoted to improving the reliability of RB should focus on the removal of faults that lead to common mode failures, *viz*, *independent* faults in the acceptance test and *related* faults between alternates, or between alternates and the acceptance test.

Considering again the various sources of failures — common mode failures and failures due to the activation of *independent* faults in P & S, if the influence of *independent* faults in P & S is neglected, the failure intensity (9) becomes:

$$h_{\mathrm{RB}}(t) = \pi_P h_R(\pi_P t) + \pi_T h_T(\pi_T t). \tag{15}$$

This result is important because it highlights the relationship between RB failure intensity and those of its components. Furthermore, (15) extends the result in [11,16] concerning the relationship between the equivalent failure rate of the RB and those of its components in stable reliability (when neglecting the influence of *independent* faults in P & S), to a more general relation between associated failure intensities when reliability growth is accounted for.

3.2.2 RB in stable reliability

The RB reliability in a stable-reliability situation can be evaluated either directly from the Markov chain of figure 6 or from the transformed and aggregated Markov chain of figure 8, making:

$$\omega_X = 0 \text{ and } \zeta_{\inf,X} = \lambda_X,\ X \in \{P,S,T,R\} \text{ in (5).}$$

This leads to:

$$r^*_{\mathrm{RB}}(t) = \exp[-(((\lambda_S+\lambda_T\mu_T\gamma_S)\lambda_P\mu_S+\lambda_R)\pi_P$$

$$+ \lambda_T\pi_T)t] = \exp(-\lambda_{\mathrm{eq}}t), \tag{16}$$

$$\mu_X \equiv 1/(\lambda_X + \gamma_X)$$

where λ_{eq} is the equivalent failure rate of the RB, corresponding to stable reliability. This result can be verified by direct processing of the model of figure 6 using the aggregation technique. This equivalent failure rate also corresponds to the failure intensity in (13) & (14) when all the reliability growth factors take on their minimum value ($k_X=1$) and $h_X(\infty)=\lambda_X$, $X \in \{P,S,T,R\}$.

For stable reliability:

$$R^*_{RB}(t,t+u) = \exp[-(\lambda_{eq}(t+u)]/\exp(-\lambda_{eq}t) = \exp(-\lambda_{eq}u) \approx 1 - \lambda_{eq}u. \quad (17)$$

As anticipated, it does not depend on time t.

3.3. *Sensitivity Analysis: Numerical Examples*

These numerical examples illustrate the analytic results in section 3.2. We consider the following ranges for the λ_X & k_X:

$$5 \le k_P \le 10,\ 2 \le k_T \le 5,\ 2 \le k_R \le 5,\ 1 \le k_S \le 5.$$

$$10^{-3} \le \lambda_T/\lambda_P \le 10^{-1},\ 10^{-4} \le \lambda_R/\lambda_P \le 10^{-2},$$

$$1 \le \lambda_S/\lambda_P \le 10.$$

This choice is due to the nature of an RB. The primary is executed several times before execution of the secondary. Reliability growth can thus be higher for the primary and might even be non-important for the secondary: since it is less frequently executed, it fails less frequently and then fewer modifications are therefore anticipated. Regarding the acceptance test, its reliability growth can be lower than that of the primary because, generally —

- emphasis is put on the validation of the acceptance test which is common to both alternates
- it is simpler than the alternates.

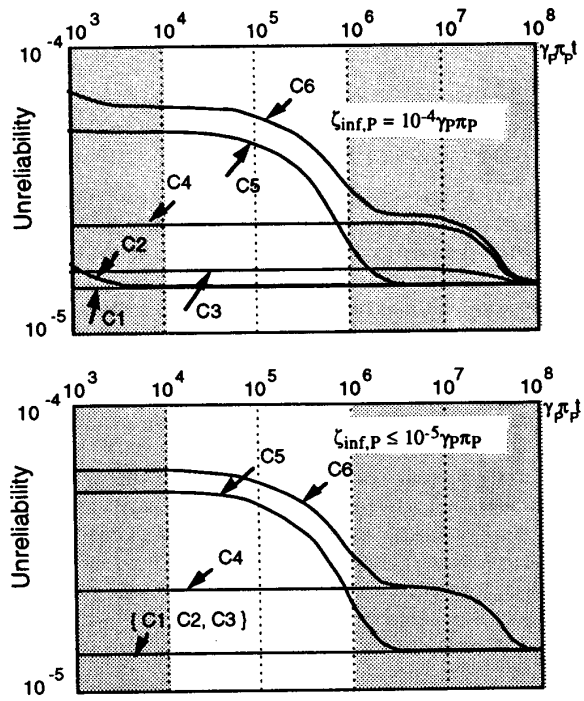
In operation, the reliability improvement of the acceptance test is anticipated to be not very high. With respect to *related* faults, it is anticipated that, thanks to the engineering techniques used during software specification and development to enhance software diversity, only few *related* faults will remain in the software when released for operation. As a result, the k_R is anticipated to be relatively small.

3.3.1. Influence of *independent* and *related* faults

Figure 9 plots unreliability, $1-R_{RB}(t,t+u)$, with respect to $(\gamma_P\pi_P t)$ which corresponds to the mean number of executions performed by the RB in the absence of failure. Various curves are displayed and a large range of variation of $\gamma_P\pi_P t$ is considered to illustrate the impact on the reliability growth of RB of the reliability growth associated with each failure process. The range of time variation that is likely to be important in practice corresponds to the non-shaded regions of figure 9. The values of the failure rates are given with respect to $\zeta_{inf,P}$. To highlight the impact of activation of *independent* faults in P or S, two values of $\zeta_{inf,P}$ are used, as shown on figure 9.

The main results from figure 9 are derived below. Two extreme situations are displayed:

- $C1$ corresponds to stable reliability with λ_X, $X \in \{P,S,T,R\}$, taking on its asymptotic minimum values ($\lambda_X = \zeta_{inf,X}$),



$k_P = 10$, $\omega_P = 0.95$
$k_S = 3$, $\omega_S = 0.5$
$k_T = 5$, $\omega_T = 0.2$
$k_R = 3$, $\omega_R = 0.95$
$\gamma_S = \gamma_P$
$\gamma_T = 10\gamma_P$
$\zeta_{inf,S} = \zeta_{inf,P}$
$\zeta_{inf,T} = \zeta_{inf,P}/100$
$\zeta_{inf,R} = \zeta_{inf,P}/2000$
$u = 100/(\gamma_P\pi_P)$

Curve	P	S	T	R
C1	→	→	→	→
C2	↗	→	→	→
C3	→	↗	→	→
C4	→	→	→	↗
C5	→	→	↗	→
C6	↗	↗	↗	↗

Figure 9. Unreliability Curves for RB

- $C6$ displays the evolution of RB unreliability when reliability growth due to the progressive elimination of the considered types of faults is taken into account.

The discrepancy between $C1$ & $C6$ illustrates error estimations that would be made when RB reliability is estimated assuming stable reliability, even though as is usually the case, the failure intensity function is not yet stabilized to its lower asymptotic value and the software-reliability is still growing. This discrepancy is directly related to the reliability growth factors of the processes and to the considered values for the failure rates

corresponding to stable reliability. Because these failure rates take on their asymptotic values, optimistic results are obtained (curve $C1$).

Comparison of $C2$ – $C5$ with $C6$ allows us to analyze how the RB reliability evolves when maintenance is performed in order to remove design faults:

- removal of *independent* faults from P or S generally leads to non-important variations in RB reliability, the most important variations are observed when the acceptance test design faults or *related* faults are progressively removed,
- when the activation rate of the P *independent* faults is high ($\zeta_{\inf,P} = 10^{-4}\gamma_P\pi_P$), the reliability improves in the beginning of the operational phase. When the elapsed time since the beginning of the operational use of RB is of the order of magnitude of $1/h_P(0)$ (which is generally small), the influence of *independent* fault removal from P becomes vanishingly small. After removal of faults with high activation rates (*independent* faults of P or S), low rate activation faults are revealed, (*related* faults and acceptance test faults). Figure 9 shows that the removal of these faults leads to an important decrease in RB unreliability ($C4$ & $C5$). The maximum value and the rate of decrease of RB unreliability depends on the reliability growth factors k_T & k_R and on ω_T & ω_R.

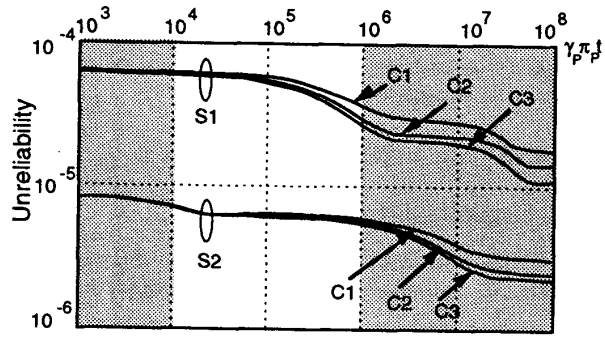
For the assumed values of the parameters associated with the acceptance test faults and *related* faults, the removal of faults from T appears to have the most important influence. The impact of reliability growth originating from the elimination of *related* faults is observed later during the operational phase, (when t is of the order of magnitude of $1/h_R(0)$ which in practice can be very high). If the failure intensity associated with fault activation in the acceptance test is lower than the failure intensity corresponding to *related* faults, a behavior analogous to that illustrated in figure 9 is observed. However, in this case, RB reliability is more sensitive to elimination of *related* faults than to elimination of acceptance test faults.

3.3.2. Influence of k_T & k_R

Figure 10 illustrates the relative influence of acceptance test faults and *related* faults on RB reliability: two sets of $C1$ – $C3$ are displayed, for each one of them, fixed values for $h_T(0)$ & $h_R(0)$, and various values for k_T k_R are assumed. For example, different values for $h_T(0)$ & $h_R(0)$ correspond to two recovery block software packages with various qualities at the beginning of the operation phase, while different values of k_T & k_R correspond to various maintenance strategies adopted during the RB operation phase. Thus the RB reliability improvement depends on the extent to which *related* faults and the acceptance test faults are removed and on the maximum values of failure intensities at the beginning of the operation phase.

3.3.3. Influence of the execution rates γ_P, γ_S, γ_T

In the previous examples, RB reliability was evaluated with respect to $\gamma_P\pi_P t$ = mean number of executions performed in the environment in which the software is run, and the same values of γ_S & γ_T being considered. Nevertheless, it is also interesting to study the impact of these parameters on RB reliability.



$h_P(0) = 10^{-4}\gamma_P\pi_P$
$h_S(0) = 3h_P(0)$
S1: $h_T(0) = 0.5h_P(0)$
$h_R(0) = 0.015h_P(0)$
S2: $h_T(0) = 0.05h_P(0)$
$h_R(0) = 0.0015h_P(0)$

Curve	k_T	k_R
$C1$	3	5
$C2$	5	3
$C3$	6.5	4

The other parameters are the same as in figure 9

Figure 10. RB Reliability Variation *vs* Removal of *Related* Faults and T Faults

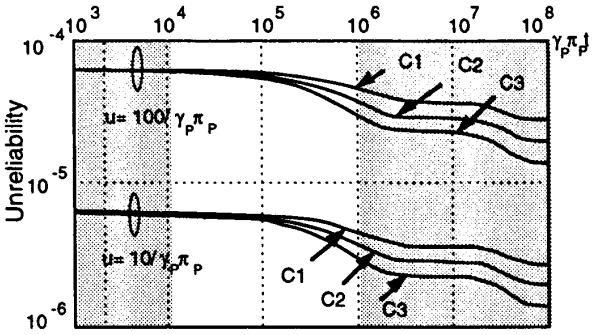
A sensitivity analysis of RB reliability with respect to the execution rate variations has been conducted. These parameters appreciably affect RB reliability only when the failure intensities associated with activation of *independent* faults in P or S are high compared to the failure intensities relative to the activation of *independent* faults in T or *related* faults. When this does not occur, as anticipated from (15), only π_P & π_T affect the RB reliability. These parameters are specific to the environment in which the software is run and are not directly related to the software maintenance process.

3.3.4. Influence of mission duration

The influence of mission duration, u, on system unreliability is illustrated in figure 11 which plots 2 sets of $[1 - R_{RB}(t,t+u)]$ corresponding to 2 values of u; for each set, 3 values of k_T are considered. As anticipated from (3), u does not affect the shape of the unreliability curves.

3.4. *Practical considerations*

One of the key results from the sensitivity analyses is that RB reliability growth originates mainly from the removal of faults leading to common mode failures (acceptance test faults



$h_P(0) = 10^{-4}\gamma_P\pi_P$
$h_S(0) = 3h_P(0)$
$h_T(0) = 0.5h_P(0)$
$h_R(0) = 0.015h_P(0)$

Curve	k_T
C1	5
C2	3
C3	2

The other parameters have the same values as in figure 9.

Figure 11. Unreliability Curves for 2 Mission Durations *vs* k_T

and *related* faults) rather than from the elimination of *independent* faults revealed in the alternates. As a result, with respect to the maintenance process, since no major improvement in RB reliability is anticipated from the removal of *independent* faults, immediate correction is not necessary. Corrections can be delayed and then introduced within a new software version. However, the maintenance effort must not be limited to the identification and correction of faults leading to common mode failures, since the activity of identifying *independent* faults might reveal the presence of other residual faults whose consequence could be critical. An additional reason for removing *independent* faults in the primary is that their activation leads to performance overhead due to the secondary execution.

4. NVP MODELING & EVALUATION

Since the software systems considered in this paper tolerate a single fault, the architecture has 3 versions and a decider.

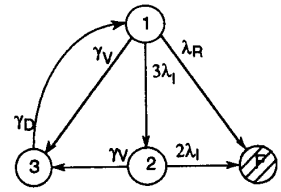
4.1. *Model Derivation & Processing*

Since we do not distinguish between detected and undetected errors, the system failures are due to:

- decider failures, or
- activation of *related* faults, or
- activation of at least two *independent* faults in two versions (leading or not to similar results).

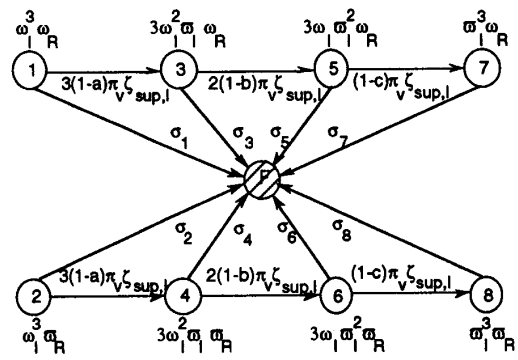
Because the failures of the decider lead to NVP failure, the failure rate of the decider is merged with the failure rate corresponding to *related* faults.

The same modeling approach is adopted as for RB. The models and the corresponding expressions for stable reliability are summarized in figure 12a and for reliability growth (corresponds to the aggregated model) are summarized in figure 12b.



γ_V execution rate of the versions
γ_D execution rate of the decider
λ_I failure rate due to *independent* faults in one version
λ_R common mode failure rate.
$r^*_{NVP}(t) = \exp[-(\lambda_z+\lambda_R)\pi_V t) = \exp(-\lambda_{eq}t)$
$\lambda_z \equiv 6\lambda_I/(2 + \gamma_V/\lambda_I)$

Figure 12a. NVP Markov Model for Stable Reliability



$\sigma_1 = (3a\zeta_{sup,I} + \zeta_{sup,R})\pi_V$
$\sigma_2 = \sigma_1 + (\zeta_{inf,R} - \zeta_{sup,R})\pi_V$
$\sigma_3 = (2b\zeta_{sup,I} + a\zeta_{inf,I} + \zeta_{sup,R})\pi_V$
$\sigma_4 = \sigma_3 ! (\zeta_{inf,R} - \zeta_{sup,R})\pi_V$
$\sigma_5 = (c\zeta_{sup,I} + 2b\zeta_{inf,I} + \zeta_{sup,R})\pi_V$
$\sigma_6 = \sigma_5 + (\zeta_{inf,R} - \zeta_{sup,R})\pi_V$
$\sigma_7 = (3\,c\,\zeta_{inf,I} + \zeta_{sup,R})\pi_V$
$\sigma_8 = \sigma_7 + (\zeta_{inf,R} - \zeta_{sup,R})\pi_V$
$\pi_V \equiv \gamma_D/(\gamma_D+\gamma_V)$
$a \equiv [1 + \gamma_V/(2\zeta_{sup,I})]^{-1}$
$b \equiv [1 + \gamma_V/(\zeta_{sup,I} + \zeta_{inf,I})]^{-1}$
$c \equiv [1 + \gamma_V/(2\zeta_{inf,I})]^{-1}$

$r_{NVP}(\tau) = (\omega_R\cdot\omega_I^3)\cdot[\exp(-\zeta_{sup,R}\cdot\tau') + \theta_R\cdot\exp(-\zeta_{inf,R}\cdot\tau')]$
$\cdot[A\cdot\exp(-3\zeta_{sup,I}\cdot\tau') + B\cdot\exp(-2\zeta_{sup,I}\cdot\tau') + C\cdot\exp(-\zeta_{sup,I}\cdot\tau') + D\cdot\exp(-3c\cdot\zeta_{inf,I}\cdot\tau')]$
$\theta_x \equiv (1/\omega_x)-1,\ x = I, R$
$\tau' \equiv \pi_V\cdot\tau$
$Q(x_1, x_2) \equiv x_1/[(x_2\cdot\zeta_{inf,I}/\zeta_{sup,I}) - 1]$
$A \equiv [1 + Q(3\bar{a},a)\cdot[1 + Q(b,b)\cdot[1 + Q(\bar{c}/3,c)]]]$
$B \equiv 3[\theta_I - Q(\bar{a},a)]\cdot[1 + Q(2\bar{b},2b-a)\cdot[1 + Q(\bar{c},3c-a)]]$
$C \equiv 3[\theta_I^2 - Q(2\bar{b},2b-a)\cdot[\theta_I - Q(\bar{a}/2,b)]]\cdot[1+ Q(\bar{c},3c-2b)]$
$D \equiv \theta_I^3 - 3Q(\bar{c},3c-2b)\cdot[\theta_I^2 - Q(\bar{b},(3c-a)/2)\cdot[(\theta_I - Q(\bar{a}/3,c)]]$

Figure 12b. NVP Markov Model for Reliability Growth

The aggregated model in stable reliability can be obtained from figure 12a by aggregating the non-failed states or from figure 12b by making $\omega_X = 0$ (equivalent to removing states 1 - 7 and the corresponding transitions).

For case RB,

$$r_{\mathrm{NVP}}(t) = \sum_{i=1}^{8} \alpha_i \cdot \exp(-\beta_i t)$$

$$\sum_{i=1}^{8} \alpha_i = 1,\ \alpha_i \geq 0. \tag{18}$$

$$h_{\mathrm{NVP}}(0) = \pi_V \cdot k_R \cdot h_R(\infty) + \pi_V \cdot (6/\gamma_V) \cdot [k_I \cdot h_I(\infty)]^2, \tag{19}$$

$$h_{\mathrm{NVP}}(\infty) = \pi_V \cdot h_R(\infty) + \pi_V \cdot (6/\gamma_V) \cdot [h_I(\infty)]^2. \tag{20}$$

Since γ_V is generally several orders of magnitude higher than the failure intensity associated with *independent* faults, (19) & (20) suggest that, as for RB (see (12) & (13)), *independent* faults in the versions have a lower impact on NVP reliability than *related* faults.

When the influence of *independent* faults can be neglected, a simpler expression for NVP failure intensity is:

$$h_{\mathrm{NVP}}(t) = \pi_V \cdot h_R(\pi_V \cdot t). \tag{21}$$

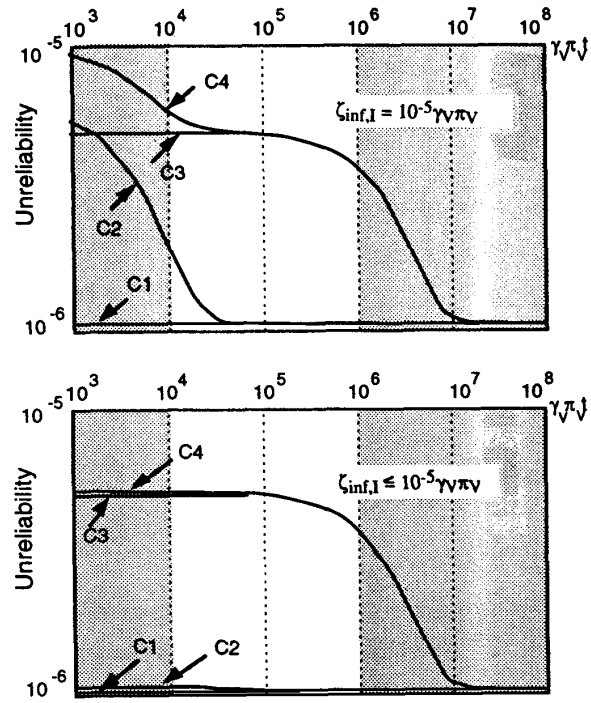
4.2. *Sensitivity Analysis: Numerical Examples*

The reliability growth of the different versions is assumed to be the same for all the versions and of the same order of magnitude as the primary of a RB.

Figure 13 shows unreliability curves for NVP as a function of the mean number of NVP executions without failure: $\gamma_V \pi_V t$. As for figure 9, the non-shaded regions correspond to the range of time variation that is important in practice. To illustrate the impact of *independent* faults, two values of $\zeta_{\mathrm{inf},I}$ are considered, as shown in figure 13.

Figure 13 shows that if the reliability growth of the components of an NVP architecture is not taken into account, too-high estimations of global reliability are obtained once again (when the failure rates for stable reliability correspond to their minimum values). As for RB, reliability is more sensitive to the reliability growth of *related* faults. Even though *independent* faults have more impact on NVP reliability than on RB reliability, the conclusions are the same: *independent* faults have no important influence when the corresponding failure rates are small. The impact is felt only for high *independent* failure rates, *ie*, when $6\lambda_I^2/(2\lambda_I+\gamma_V) \approx 6\lambda_I^2/\gamma V \sim \lambda_R$.

The rate of evolution of NVP reliability towards the asymptotic behavior depends on reliability growth during the operational life. This is depicted in figure 14 which plots two sets ($S1$ & $S2$) of NVP reliability curves corresponding to two values for $h_R(0)$ (different values for k_R are used for each set). Depending on the removal rate of the *related* faults and on the failure intensity at the beginning of the operational phase, various behaviors of NVP can be observed.



$k_I = 10$, $\omega_I = 0.95$
$k_R = 5$, $\omega_R = 0.1$
$\gamma_D = 10\gamma_V$
$\zeta_{\mathrm{inf},R} = \zeta_{\mathrm{inf},I}/1000$
$u = 100/(\gamma_V \pi_V)$

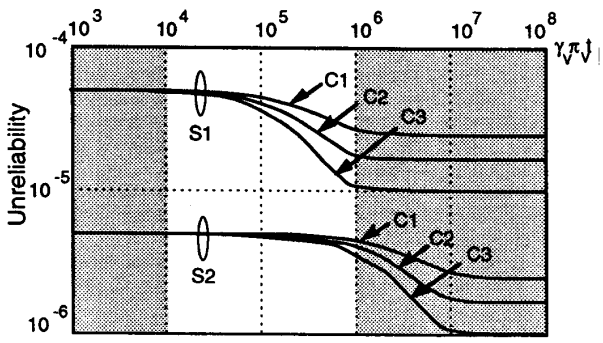
Curve	I	R
C1	→	→
C2	↗	→
C3	→	↗
C4	↗	↗

Figure 13. Unreliability Curves for NVP

5. RB & NVP COMPARISON

Comparison of the failure intensity expressions for RB, (12) & (13), and NVP, (19) & (20), shows that, for NVP, removal of *independent* faults has more impact on reliability than for RB since the versions are run in parallel. Nevertheless, as the influence of *independent* faults is perceived only at the beginning of the operational phase when high failure rates are observed, their impact can be neglected. Then they should be compared by considering only faults leading to common-mode failures: *related* faults and *independent* faults in the acceptance test for RB and *related* faults and faults in the decider for NVP.

Comparison of failure intensity expressions for RB (15) and NVP (21) suggests that NVP is better than RB because the impact of *related* faults between the versions and the decider is likely to be greater for RB than for NVP — due mainly to the fact that the acceptance test is specific to each application

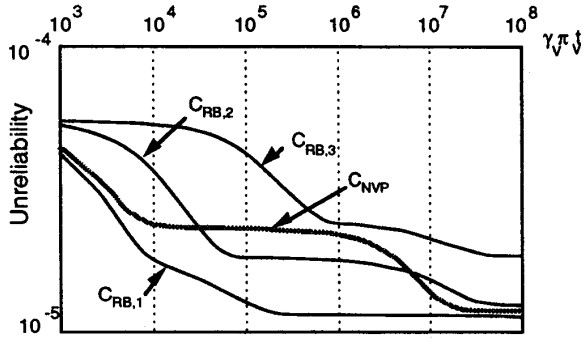


$S1$: $h_R(0) = 10^{-4}\gamma_V\pi_V$
$S2$: $h_R(0) = 10^{-5}\gamma_V\pi_V$

Other parameters have the same values as in figure 13.

Curve	k_R
$C1$	5
$C2$	3
$C3$	2

Figure 14. NVP Reliability Variation with respect to Removal of *Related* Faults.



$\gamma_V\pi_V \cdot t = \gamma_P\pi_P \cdot t$; $u = 100/(\gamma_V\pi_V) = 100/\gamma_P\pi_P$

C_{NVP}
$k_I=10$, $\omega_I=0.95$
$k_R=2$, $\omega_R=0.5$
$h_I(0)=10^{-4}\gamma_V\pi_V$
$h_R(0)=2\cdot10^{-7}\gamma_V\pi_V$
$\gamma_D=10\gamma_V$

$C_{RB,1}$, $C_{RB,2}$, $C_{RB,3}$		
$k_P=10$,	$\omega_P=0.95$,	$h_P(0)=2.5\cdot10^{-4}\gamma_V\pi_V$
$k_S=3$,	$\omega_S=0.5$,	$h_S(0)=1.0\cdot10^{-5}\gamma_V\pi_V$
$h_T(0)=9\cdot10^{-7}\gamma_V\pi_V$		$h_R(0)=5\cdot10^{-7}\gamma_P\pi_P$
$\gamma_T=10\gamma_P$		$\gamma_S=\gamma_P$

Curves	k_T	k_R	ω_T	ω_R
$C_{RB,1}$	5	5	.004	.001
$C_{RB,2}$	3	5	.5	.006
$C_{RB,3}$	3	3	.5	.006

Figure 15. Unreliability Curves for RB & NVP

whereas the decider is generic to a large extent. When considering reliability growth, however, the rate of evolution of the failure intensity function depends on many factors; for instance, the maintenance activities could be different for RB and NVP. As a result, depending on reliability growth and on the values of the reliability growth factors observed, the reliability of RB could be better or worse than that of NVP.

Figure 15 shows this phenomenon; it plots unreliability curves with respect to the mean number of executions without failure: $C_{RB,1}$, $C_{RB,2}$, $C_{RB,3}$, for RB, and C_{NVP} for NVP. For comparison purposes, it is assumed that at the beginning of the operational phase:

- the values of the failure intensities associated with the activation of *independent* faults in the primary P and *independent* faults in one version are the same: $h_P(0) = h_I(0)$.
- because the failure intensity associated with *related* faults in RB is anticipated to be higher than the failure intensity associated with *related* faults in NVP: [$h_R(0)$ for RB] > [$h_R(0)$ for NVP].
- the failure intensity associated with activation of *independent* faults in the acceptance test for RB is higher than the failure intensity associated with the activation of *related* faults: $h_T(0)$ > [$h_R(0)$ for RB] > [$h_R(0)$ for NVP].
- [t_e for RB] = [t_e for NVP]: $t_e = 1/(\gamma_P\pi_P) = 1/(\gamma_V\pi_V)$. t_e is the execution time without failure.
- to compare RB & NVP and to illustrate the impact of reliability growth due to maintenance activities, fixed values are attributed to [k_R & ω_R for NVP] while various reliability curves are plotted for RB with various values of k_R, ω_R, k_T, ωT.

The numerical values of the model parameters are given in figure 15. Even though at the beginning of the operational phase, RB reliability is worse than that of NVP since the failure intensity of RB is higher than that of NVP, this can be reversed due to maintenance activities (compare $C_{RB,2}$ to C_{NVP}).

However, the comparison above is only partial. Additional features have to be taken into account, such as the fact that, for RB, service delivery is suspended during error recovery, *ie*, when the secondary is invoked.

ACKNOWLEDGMENT

We are pleased to thank the editors of this special issue and an anonymous referee; their pertinent comments greatly helped in improving the final version of the paper. The work reported in this paper was partially supported by the ESPRIT Basic Research Action on Predictably Dependable Computing Systems (Actions no. 3092 & 6362).

REFERENCES

[1] J. Arlat, K. Kanoun, J. C. Laprie, "Dependability evaluation of software fault tolerance", *Proc. 18th IEEE Int'l Symp. Fault-Tolerant Computing*, 1988 Jun, pp 142–147; FTCS-18, Tokyo, Japan. Also in *IEEE Trans. Computers*, vol 39, 1990 Apr, pp 504–513.

[2] H. Ascher, H. Feingold, *Repairable Systems Reliability: Modeling, Inference, Misconceptions and Their Causes*, Lecture Notes in Statistics, vol 7, 1984.

[3] C. Béounes *et al*, "SURF-2: A program for dependability evaluation of complex hardware and software systems", *Proc. 23rd IEEE Int'l Symp. Fault Tolerant Computing* (FTCS-23), 1993 Jun; pp 668-673 Toulouse, France.

[4] P. G. Bishop, F. D. Pullen, "Error-masking: A source of failure dependency in multi-version programs", *Dependable Computing for Critical Applications*, 1991, pp 53–74; Springer-Verlag. Also in *Proc. 1st Int'l Working Conf. Dependable Computing in Critical Applications*, 1989 Aug, pp 25–32.

[5] A. Bobbio, K. Trivedi, "An aggregation technique for the transient analysis of stiff Markov chains", *IEEE Trans. Computers*, vol C-35, 1986 Sep, pp 803–814.

[6] L. Chen, A. Avizienis, "N-version programming: A fault-tolerance approach to reliability of software operation", *Proc. 8th IEEE Int'l Symp. Fault Tolerant Computing* (FTCS-8), 1978 Jun, pp 3–9.

[7] D. R. Cox, H. D. Miller, *The Theory of Stochastic Processes*, 1968; Methuen.

[8] A. Csenki, "Recovery block reliability analysis with failure clustering", *Dependable Computing for Critical Applications*, 1991, pp 75–104; Springer-Verlag. Also in *Proc. 1st Int'l Working Conf. Dependable Computing in Critical Applications*, 1989 Aug, pp 33–42.

[9] D. E. Eckhardt, L. D. Lee, "A theoretical basis for the analysis of multi-version software subject to coincident errors", *IEEE Trans. Software Engineering*, vol SE-11, 1985 Dec, pp 1511–1517.

[10] B. V. Gnedenko, Y. K. Belyayev, A. D. Solovyev, *Mathematical Methods of Reliability Theory*, 1969; New Academic Press.

[11] A. Grnarov, J. Arlat, A. Avizienis, "On the performance of software fault-tolerance strategies", *Proc. 10th IEEE Int'l Symp. Fault Tolerant Computing* (FTCS10), 1980 Oct, pp 251–253.

[12] H. Hecht, "Fault-tolerant software", *IEEE Trans. Reliability*, vol R-28, 1979 Aug, pp 227–232.

[13] K. Kanoun, T. Sabourin, "Software dependability of a telephone switching system", *Proc. 17th IEEE Int'l Symp. Fault Tolerant Computing* (FTCS-17), 1987 Jun, pp 236–241.

[14] K. Kanoun, M. Bastos Martini, J. Moreira De Souza, "A method for software reliability analysis and prediction — Application to the TROPICO-R switching system", *IEEE Trans. Software Engineering*, vol 17, 1991 Apr, pp 334–344.

[15] K. Kanoun, M. Kaâniche, J. C. Laprie, S. Metge, "SoRel: A tool for software reliability growth analysis and prediction from statistical failure data", *Proc. 23th IEEE Int'l Symp. Fault Tolerant Computing* (FTCS-23); pp 656-659 see [3].

[16] J. C. Laprie, "Dependability evaluation of software systems in operation", *IEEE Trans. Software Engineering*, vol SE-10, 1984 Nov, pp 701–714.

[17] J. C. Laprie, J. Arlat, C. Beounes, K. Kanoun, "Definition and analysis of hardware- and software-fault-tolerant architectures", *IEEE Computer*, vol 23, 1990 Jul, pp 39–51.

[18] J. C. Laprie, K. Kanoun, C. Béounes, M. Kaâniche, "The KAT (Knowledge-Action-Transformation) approach to the modeling and evaluation of reliability and availability growth", *IEEE Trans. Software Engineering*, vol 17, 1991 Apr, pp 370–382. Also in *Proc. 20th IEEE Int'l Symp. Fault Tolerant Computing* (FTCS-20), 1990 Jun, pp 364–371.

[19] J. C. Laprie, K. Kanoun "X-Ware reliability and availability modeling", *IEEE Trans. Software Engineering*, vol 18, 1992 Feb, pp 130–147.

[20] J. C. Laprie, "Dependability: Basic concepts and terminology", *Dependable Computing and Fault-Tolerant Systems*, vol 5, (J.C. Laprie, Ed), 1992; Springer Verlag.

[21] Y. Levendel, "Defects and reliability analysis of large software systems: Field experience", *Proc. 19th IEEE Int'l Symp. Fault Tolerant Computing* (FTCS-19), 1989 Jun, pp 238–244.

[22] B. Littlewood, "Software reliability model for modular program structure", *IEEE Trans. Reliability*, vol R-28, 1979 Aug, pp 241–246.

[23] B. Littlewood, D. R. Miller, "A conceptual model of multi-version software", *Proc. 17th IEEE Int'l Symp. Fault Tolerant Computing* (FTCS-17), 1987 Jul, pp 150–155.

[24] A. Marsan, G. Balbo, G. Conte, "A class of generalized stochastic petri nets for the performance analysis of multiprocessor systems", *ACM Trans. Computers*, vol 2, 1984 May, pp 93–122.

[25] D. R. Miller, "Exponential order statistic models of software reliability growth", *IEEE Trans. Software Engineering*, vol SE-12, 1986 Jan, pp 12–24.

[26] J. D. Musa, K. Okumoto, "A logarithmic Poisson execution time model for software reliability measurement", *Proc. Compsac'84*, 1984, pp 230–238; Chicago.

[27] B. Randell, "System structure for software fault tolerance", *IEEE Trans. Software Engineering*, vol SE-1, 1975 Jun, pp 220–232.

[28] F. Saglietti, W. Ehrenberger, "Software diversity — Some considerations about its benefits and its limitations", *Proc. SAFECOMP'86*, 1986 Oct, pp 27–34; Sarlat, France.

[29] R. K. Scott, J. W. Gault, D. F. McAllister, "Fault-tolerant software reliability modeling", *IEEE Trans. Software Engineering*, vol SE-13, 1987 May, pp 582–592.

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

[31] S. Yamada, S. Osaki, "Software reliability growth modeling: Models and assumptions", *IEEE Trans. Software Engineering*, vol SE-11, 1985 Dec, pp 1431–1437.

AUTHORS

Dr. Jean Arlat; LAAS-CNRS; 7, Avenue du Colonel Roche; 31077 Toulouse Cedex, FRANCE.

Jean Arlat was born in Toulouse, France, in 1953. He received the Engineer degree from the National Institute of Applied Sciences of Toulouse in 1976, and the Doctor in Engineering and the Doctor es-Sciences from the National Polytechnic Institute of Toulouse in 1979 & 1990. He is "Charge de Recherche" of CNRS. He joined the research group on Fault Tolerance and Dependable Computing at LAAS-CNRS in 1976. His research interests focus mainly on the evaluation of hardware-and-software fault-tolerant systems including both analytic modeling and experimental fault injection approaches, subjects on which he (co)authored more than 30 papers. From 1992 Apr to Sep, he has been teaching as an Invited Visiting Professor at the Dept. of Computer Science of the Tokyo Institute of Technology in the framework of the Toshiba Endowed Chair in Intellectual Information Systems. He also contributed to various National and European research contracts and acted as a consultant to several organizations in France. He is the Vice-Chair'n of the IEEE Computer Society Technical Committee on Fault-Tolerant Computing since 1992 Jan. He was member of the program committees of several international conferences and workshops, and acts as a referee for several international conferences and journals. He is a member of AFCET and IEEE.

Dr. Christian Béounes; LAAS-CNRS; 7, Avenue du Colonel Roche; 31077 Toulouse Cedex, FRANCE.

Christian Béounes was "Chargé de Recherche" of INRIA, the French National Institute for Computing and Automatic Control Research. He joined LAAS in 1974 as a member of the "Fault Tolerance and Dependable Computing" group. His research interests included stochastic Petri nets modeling, and dependability evaluation of computer systems. He received the Certified Engineer degree from the National Institute of Applied Sciences, Toulouse, France in 1973 and the Doctor in Engineering in Automatic Control from the University of Toulouse in 1977. Christian passed away on 1993 April 23, after several months of a painful battle against illness. He lives on in our memory, both for his professional and human qualities.

Dr. Mohamed Kaâniche; LAAS-CNRS; 7, Avenue du Colonel Roche; 31077 Toulouse Cedex, FRANCE.

Mohamed Kaâniche was born in Sfax, Tunisia in 1963. He received the Certified Engineer degree from the National School of Civil Aviation of Toulouse in 1987, and the Doctorate in Computer Science and Automation from the University of Toulouse in 1992. He is Chargé de Recherche at CNRS since 1991. He joined LAAS in 1988 as a member of the group "Fault Tolerance and Dependable Computing". His research interests focus on dependability growth modeling of computing systems, and on software reliability evaluation.

Dr. Karama Kanoun; LAAS-CNRS; 7, Avenue du Colonel Roche; 31077 Toulouse Cedex, FRANCE.

Karama Kanoun is "Chargé de Recherche au CNRS" and a member of the research group on Fault Tolerance and Dependable Computing at LAAS. She holds the Engineer degree from ENAC (Toulouse, 1977), the Doctor in Engineering and the Doctor es-Sciences from the National Polytechnic Institute of Toulouse in 1980 and 1989. Her major research interests include hardware & software dependability evaluation (considering stable reliability as well reliability growth). She has been major investigator or contributor to several research contracts, and consultant to organizations in France (Aérospatiale, CEIS-Espace, CNES, Renault-Automation, SYSECA, SAGEM) and to the International Union of Telecommunications. She is a referee for several international conferences and journals. She was a member of the program committees of several (inter)national conferences and workshops. She has written over 40 papers for (inter)national journals and conferences. She is a member of the working group of the European Workshop on Industrial Computer Systems (EWICS): "Technical Committee 7 - Reliability, Safety and Security", a member of the AFCET working group on Dependability of Computing Systems, and a member of the IEEE Technical Committee on Software Engineering.

Dr. Jean-Claude Laprie; LAAS-CNRS; 7, Avenue du Colonel Roche; 31077 Toulouse Cedex, FRANCE.

Jean-Claude Laprie received the Certified Engineer degree from the Higher National School for Aeronautical Constructions, Toulouse, France, in 1968; he received the Doctor in Engineering in Automatic Control and the Doctor es-Sciences in Computer Science from the University of Toulouse, in 1971 and 1975. He is "Directeur de Recherche" of CNRS, the French National Organization of Scientific Research. He joined LAAS-CNRS in 1968, where he has directed the research group on Fault Tolerance and Dependable Computing since 1975. His research has focused on dependable computing since 1973, and especially on fault tolerance and on dependability evaluation, subjects on which he has (co)authored more than 100 papers, as well as coauthored or edited several books; he is the Principal Investigator of several contracts in these areas. From 1985 Jan to Aug, he was an Invited Visiting Professor at the UCLA Department of Computer Science, Los Angeles, USA. He has acted as a consultant and an expert in dependable computing in France and abroad for government agencies as well as industrial organizations. He is the Director of LIS, the recently founded Laboratory for Dependability Engineering (a joint laboratory between Matra Marconi Space, Technicatome and LAAS). Besides organizing or serving on program committees for numerous conferences and workshops, he was Chair'n of the IEEE Computer Society Technical Committee on Fault Tolerant Computing in 1984 & 1985; he has been Chair'n of the IFIP Working Group 10.4 on Dependable Computing and Fault Tolerance since 1986. He founded in 1987 the AFCET (French Association for Science and Technology of Information) Group on Computing Systems Dependability, which he chaired until 1991, and he is on the Board of Directors of AFCET. He is co-editor of the Springer Verlag series on Dependable Computing and Fault Tolerant Systems. He is a member of ACM, AFCET, and IEEE Computer Society.

Manuscript TR92-302 received 1992 May 11; revised 1992 December 1.

IEEE Log Number 07620

◄TR►