Dependability Modeling and Evaluation of Software Fault-Tolerant Systems

JEAN ARLAT, MEMBER, IEEE, KARAMA KANOUN AND JEAN-CLAUDE LAPRIE, MEMBER, IEEE

Abstract—**The paper provides dependability modeling and evaluation (encompassing reliability and safety issues) of the two major fault tolerance software approaches: recovery blocks (RB) and N-version programming (NVP). The study is based on the detailed analysis of software fault-tolerance architectures able to tolerate a single fault (RB: two alternates and an acceptance test, NVP: three versions and a decider).**

Index Terms—**Dependability evaluation, dependability modeling, software design diversity, software fault tolerance.**

I. Introduction

A NUMBER of papers devoted to the dependability analysis of software fault tolerance approaches have appeared in the literature, for which two major goals can be identified: 1) modeling and evaluation of the dependability measures [7], [10], [14], [15], [18], [24], [28], [29], 2) detailed analysis of the dependencies in diversified software [6], [11], [22], [27].

This paper is an elaboration on the work presented in [2] and belongs to the first class and analyzes the two most documented approaches to software fault tolerance: RB [26] and NVP [8]. The major extensions to published work concern: 1) the definition of a unified modeling framework based on the identification of the possible types of faults through the analysis of the software production process [18], 2) the evaluation of both reliability and safety measures, and 3) the consideration of two specific characteristics of the architectures that have received little treatment up to now: the discarding of a failed version, for NVP, and the nesting of the blocks, for RB.

Two classes of faults are considered: *independent faults* and *related faults* [3]. Related faults result either from a fault in the common specification, or from dependencies in the separate designs and implementations. Two types of related faults may be distinguished: 1) among several variants (alternates for RB or versions for NVP) and 2) among one or several variants and the decider (the acceptance test of the RB or the voting algorithm of NVP). Related faults manifest under the form of *similar errors*, whereas we shall assume that independent faults cause *distinct errors*.

Since the faults considered are design faults that are introduced in the software, either during its specification or during its implementation, we shall start the analysis of each approach by relating the various types of faults to the production process [18].

Manuscript received July 23, 1989; revised November 20, 1989. This work was carried out in the framework of the Hermes European Space Shuttle Project and of the ESPRIT Project "Predictably Dependable Computing Systems."

The authors are with the Laboratoire d'Automatique et d'Analyse des Systèmes du Centre National de la Recherche Scientifique (LAAS-CNRS), Toulouse, France.

IEEE Log Number 8933892.

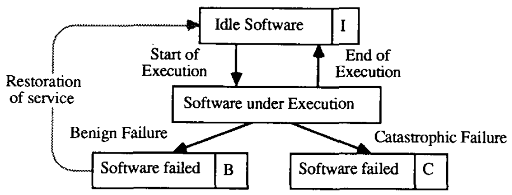


Fig. 1. General behavior model.

When a failure occurs, the detection of the inability to deliver acceptable results may be an important consideration, in the sense that an undetected failure may have, and generally has, catastrophic consequences. Although the notion of safety strongly depends on the considered application, in practice, the detection of the inability to deliver proper service is a prerequisite to initiate the specific safety procedures. A detected failure (no acceptable result is identified by the decider and no output result is delivered) will thus be termed as a benign failure, whereas an undetected failure (an erroneous result is delivered) will be termed as a catastrophic failure.

As usual, we shall consider reliability as a measure of the time to failure and safety as a measure of the time to catastrophic failure.

Software faults can manifest only when it is executed. We shall thus consider the execution process and the fault manifestation process.

The general behavior model is given in Fig. 1. Transition from B to I stands only for safety, in which case it is assumed that it is possible to restore service delivery by means of procedures carried out at an upper level, i.e., supplying input data different from those having led to benign failure. State class C is absorbing for safety whereas both state classes B and C are absorbing for reliability.

We shall assume that the behavior of the systems under consideration can be modeled as a Markov chain; for a discussion of this assumption, see, e.g., [9], [18], [21]. The execution process will be modeled through execution rates and the fault manifestation process will be modeled through probabilities conditioned on the execution of the various components of the software: the variants and the decider. The transition rates outputting from the nonabsorbing states are of the form

$$\lambda_{ij} = p_{ij} \cdot \lambda_i \text{ with } \sum_j p_{ij} = 1 \qquad (1)$$

where i designates a nonabsorbing state, λ_i is the rate associated to the tasks executed in state i, and p_{ij} represents the probability of the transition from state i to state j of the model.

When the nonabsorbing states (nonfailed states for reliability, nonfailed and benign failure states for safety) constitute an irreducible set [12] (i.e., the graph associated with the nonabsorbing states is strongly connected), it is shown in [25] that the absorption process is asymptotically a homogeneous Poisson process (HPP), whose failure rate Γ is given by

$$\Gamma = \sum_{\substack{\text{paths from} \\ \text{initial state } (I) \text{ to} \\ \text{absorbing states}}} \frac{\prod (\text{transition rates of the considered path})}{\prod_{\substack{\text{states in path} \\ (I \text{ excepted})}} \left\{ \sum (\text{output rates of the considered state}) \right\}}. \quad (2)$$

The rate of convergence of the absorption process towards the asymptotic HPP is directly related to the execution rates; it is thus reached very rapidly (say, after three executions). We shall adopt this approach in the following whenever possible, and we shall denote as equivalent rate, the rate of the asymptotic HPP. Γ_R will denote the *equivalent failure rate* for reliability and Γ_S is the *equivalent catastrophic failure rate* for safety.

Using relations 1) and 2), it can be easily verified that the equivalent failure rates can be expressed simply using: 1) the departure rate σ from state I of Fig. 1 and 2) the probability of failure of the software obtained from the embedded discrete chain. Let Q_R (resp., Q_S) be the probability of failure (resp., catastrophic failure), thus,

$$\Gamma_R = \sigma Q_R, \quad \Gamma_S = \sigma Q_S. \quad (3)$$

Accordingly, reliability ($R(t)$) and safety ($S(t)$) are given by

$$R(t) = \exp(-\Gamma_R t) \quad S(t) = \exp(-\Gamma_S t). \quad (4)$$

As Q_R and Q_S are evaluated directly from the discrete Markov chain, in the sequel we focus essentially in the presentation of the discrete Markov chains describing the fault manifestation process of the fault-tolerant softwares.

Finally, it is worth noting that we focus on the fault-tolerant software itself, i.e., the underlying mechanisms are not considered: 1) recovery point establishment and restoration for RB, and 2) synchronization of the versions, cross-check points establishment for NVP.

The sequel of the paper is organized into four sections. Sections II and III present, respectively, the analyses of RB and NVP: for each approach a detailed model based on the production process of the fault-tolerant software is first established and then it is simplified through the assumptions that only a single fault type may manifest during execution of the fault-tolerant software and that no error compensation may take place within the software. Section IV introduces some elements for RB and NVP comparison. Section V analyzes the nested RB's.

II. Recovery Blocks

Fig. 2(a) shows the production process of an RB with two alternates [a primary (P) and a secondary (S)], and one acceptance test (AT). During the diversified designs and implementations of P, S, and AT, *independent faults* may be created. However, due to dependencies, some *related faults* between P and S or between P, S, and the AT may be introduced. Faults committed during common specification (path $1 \to 2$, $1 \to 3$, $1 \to 2 \to 3$) are likely to be related faults and, as such, the cause of *similar errors*. Faults created during the implementation can also lead to related faults between P, S, and AT (channels a, b, c); all these faults are summarized in Fig. 2(b). It is worth noting that the probabilities listed could be obtained from controlled experiments such as the one reported in [1].

For deriving the fault manifestation model, a question immediately arises: what types of faults are considered as possibly manifesting as the consequence of their activation? This leads to consider successively the following assumptions:

A1) only a single fault type (either independent or related) may manifest during the execution of an alternate and the AT and no error *compensation* may take place within an alternate and the AT during an execution, i.e., an error is either detected and processed or leads to catastrophic failure.

A2) only a single fault type may manifest during the execution of the whole RB and no error *compensation* may take place within the RB.

The detailed model will be based on assumption A1, which enables some singular behaviors of the decider to be characterized.

Assumption A2 will serve as a basis for the simplified model.

A. Detailed RB Model

Fig. 3 describes the M1 model based on the notation of Fig. 2(b). P, TP, S, and TS form the Software under Execution class from Fig. 1, respectively: execution of P, execution of AT after P, execution of S, execution of AT after S.

Different states are considered for TP to account for the various types of faults that may be activated in P:

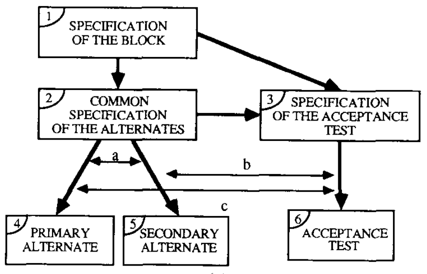
TP1) no fault activated [p_P],

TP2) activation of an independent fault [q_P],

TP3) activation of a related fault between P and S [q_{PS}],

TP4) activation of a related fault between P and the AT [q_{PT}].

The partition leads to a subsequent decomposition of states S and TS. It is assumed that no fault can be activated in AT after activation of an independent fault in P (unity transition from state TP2): these faults are considered as consisting essentially of related faults and, as such, are accounted for in probability q_{PT} leading to state TP4. Activation of a related fault between P and S (state TP3) corresponds to a detected failure and leads through $S3$ and TS3 to state B. The activation of a related fault between P and AT (state TP4) corresponds to a catastrophic failure and leads to state C.


1 SPECIFICATION OF THE BLOCK
2 COMMON SPECIFICATION OF THE ALTERNATES
3 SPECIFICATION OF THE ACCEPTANCE TEST
a
b
c
4 PRIMARY ALTERNATE
5 SECONDARY ALTERNATE
6 ACCEPTANCE TEST

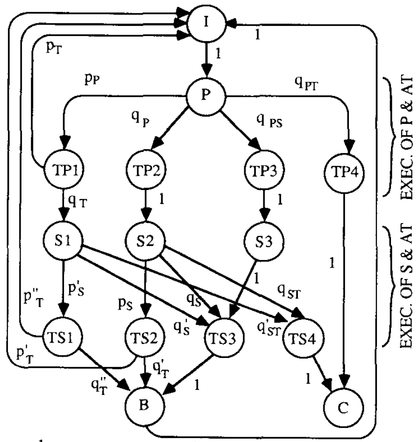
(a)

Path where fault(s) is (are) created or dependency channel(s)	**Fault type(s)**	**Probability of activation**
1 → 2 or (a)	Related fault in P and S	q_{PS}
1 → 3, (c) or 1 → 2 → 3	Related fault in P and AT (or P, S and AT)	q_{PT}*
(b)	Related fault in S and AT	q_{ST}
2 → 4 or 2 → 5	Independent fault in P or S	q_P or q_S
3 → 6	Independent fault in AT	q_T

* Since the activation of a related fault between P and AT leads to RB failure, no further decomposition with respect to the faults of S is necessary.

(b)

Fig. 2. RB analysis. (a) Fault sources in production process. (b) Fault types and notation.



$p_P = 1 - q_P - q_{PT} - q_{PS}$ $p_S = 1 - q_S - q_{ST}$ $p'_S = 1 - q'_S - q'_{ST}$
$p_T = 1 - q_T$ $p'_T = 1 - q'_T$ $p''_T = 1 - q''_T$

Fig. 3. M1—detailed RB model.

Due to the fact that S is executed only when an independent fault has been activated either in P or in AT, conditional probabilities have been introduced in the model; in particular

q_S = Prob {activation of an independent fault in $S|S$ is executed after activation of a fault in P}

q'_S = Prob {activation of an independent fault in $S|S$ is executed after activation of a fault in AT}.

The same differences in the conditions apply for q_{ST} and q'_{ST} and also for the probabilities of activation of an independent fault in the AT following the execution of S: q'_T and q''_T.

The path $\pi = \{P, TP1, S1, TS1, I\}$ corresponds to an error compensation identifying a singular behavior of the AT: the AT *rejects* an acceptable result provided by P and subsequently *accepts* the result given by S.

It is worth noting that M1 can be reduced when considering

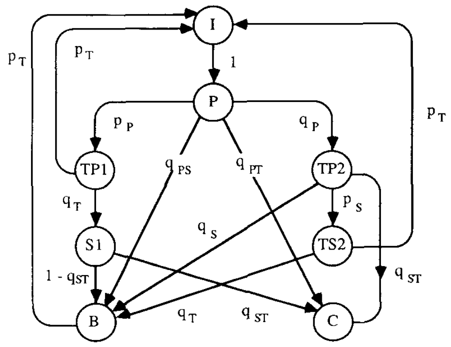


Fig. 4. Model M′1.

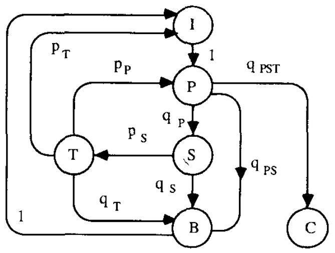


Fig. 5. M2—simplified RB model.

that

— $q_T \approx q'_T$, $q_S \approx q'_S$, $q_{ST} \approx q'_{ST}$: the probabilities of activation of a fault in S (or AT) following the activation of an independent fault either in P or in AT are equivalent, since in any case their execution is a consequence of the application of error-prone input data,

— $p''_T \ll 1$: error compensation (path π) is unlikely to occur,

— each state belonging to the Software under Execution class with an outgoing transition equal to 1 can be merged with the next state,

M1 can thus be reduced to model M′1 of Fig. 4.

B. Simplified RB Model

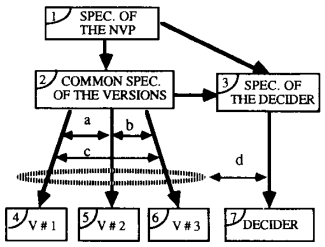
In this case, since assumption A2 applies, a single fault type can be activated in the whole RB; thus, transitions from $S1$ and $S2$ to TS4 of model M1 (resp., $S1$ to C for M′1) must be deleted. This is equivalent to make $q_{ST} = 0$ and to merge the related faults between S and AT with the related faults between P and AT; it follows that q_{PT} becomes q_{PST}. The corresponding model (M2) is given in Fig. 5.

C. Processing of the Models

Assuming that $p_P \approx 1 - q_P$ and $p_S \approx 1 - q_S$, we obtain for models M1 and M′1:

$$\text{for reliability: } \Gamma_R = \sigma\{q_{PS} + q_{PT} + q_T + q_P[q_{ST} + q_S(1 - q_T)]\} \quad (5)$$

$$\text{for safety: } \Gamma_S = \sigma\{q_T q_{ST} + q_{PT} + q_P q_{ST}(1 - q_T)\}. \quad (6)$$



(a)

Paths where fault(s) is(are) created or dependency channel(s)	Fault type(s)	Probability of activation
$1 \to 2$	Related fault in the 3 versions	q_{3V}
(a), (b) or (c)	Related fault in 2 versions	q_{2V}
$1 \to 2 \to 3$, $1 \to 3$ or (d)	Related fault in versions and decider	q_{VD}
$2 \to 4$, $2 \to 5$ or $2 \to 6$	Independent fault in a version,	q_{IV}
$3 \to 7$	Independent fault in the decider	q_D

(b)

Fig. 6. NVP analysis. (a) Fault sources in production process. (b) Major fault types and notation.

For model M2, we obtain

$$\Gamma_R = \sigma\{q_{PS} + q_{PST} + q_T + q_P q_S(1 - q_T)\} \tag{7}$$

$$\Gamma_S = \sigma q_{PST}. \tag{8}$$

D. Comparison to a Nonfault-Tolerant Software

The comparison to a nonfault-tolerant software leads us to consider a software with no internal fault detection mechanisms whose failure rate is equal to the sum of the elementary failure rates of an alternate:

$$\Gamma'_R = \sigma\{q_P + q_{PS} + q_{PT}\} \tag{9}$$

where q_{PT} must be replaced by q_{PST} when considering assumption A2.

Comparison is presented for reliability only, since the notion of safety as defined here does not apply to a software with no internal detection mechanisms. Let define r as $r = \Gamma_R/\Gamma'_R$; the RB provides a reliability improvement if $r < 1$. This leads to

$$\text{For M}'1\text{:}\; q_T < q_P(1 - q_P - q_{ST})/(1 - q_P q_S) \tag{10}$$

$$\text{For M2:}\; q_T < q_P(1 - q_S)/(1 - q_P q_S). \tag{11}$$

Since the AT is usually less complex than P or S and assuming that complexity and probability of failure are related, we have $q_T \ll q_P$, which enables relations (10) and (11) to be verified. However, the quantification of the improvement must be studied for each specific case.

III. N-Version Programming

The potential sources of faults in the production process of an NVP software with three versions and one decider are shown on Fig. 6(a).

As the versions correspond to operational software of good quality, it can be assumed that they are of equivalent reliability, and thus:

A3) The probability of fault activation is the same for the three versions.[1]

This leads to the following notation:

q_{IV} = Prob {activation of an independent fault in one version}

q_{2V} = Prob {activation of a related fault between two specific versions}

q_{3V} = Prob {activation of a related fault between the three versions}.

Two other probabilities are defined in order to account for the faults of the decider:

q_D = Prob {activation of an independent fault in the decider}

q_{VD} = Prob {activation of a related fault between the three versions and the decider}.

The probabilities concerning the versions could be evaluated from controlled experiments such as [1] and [16]. However, these experiments do not account for the analysis of the faults in the decider. The presented models and decider-associated probabilities enable the performance of various voters under failure conditions such as the ones theoretically investigated in [23] to be accounted for and may constitute a framework for conducting more comprehensive and more adapted experiments. Fig. 6(b) summarizes this notation and relates the considered types of faults with the production process of Fig. 6(a).

Further notation will be introduced when required; in particular, let q_V denote the probability of activation of a fault in any version, thus from assumption A3 we have

$$q_V = q_{3V} + 2q_{2V} + q_{VD} + q_{IV}. \tag{12}$$

An important characteristic to account for is related to the fact that besides *error processing* procedures (majority vote based on cross-checks [8], selection of the median result [4], or other voters identified in [23], etc.), the decider implements or not specific *fault treatment* mechanisms to make a disagreeing version passive. Accordingly, the following assumptions will be considered successively.

A4) No fault treatment is carried out after error processing: should a version disagree with the result selected by the decider, the version is kept in the NVP architecture and supplied with the new input data.[2]

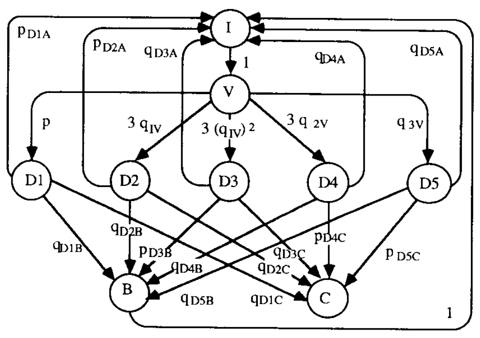
A5) Fault treatment is carried out: it consists in the identification of a disagreeing version and its elimination from the NVP architecture.

A. NVP Model without Fault Treatment

In this case, the major specification of the decision algorithm is only to provide an acceptable output result when the versions provide at least two acceptable results.

[1]This assumption is used only to simplify the notation and does not alter the significance of the results obtained; the generalization to the case where the characteristics of the versions are distinguished can be easily deduced.

[2]This applies when faults exhibit a *soft* behavior [13], [20], i.e., when it is likely that the fault will not recur in next execution.



$p = 1 - 3q_{IV} - 3(q_{IV})^2 - 3q_{2V} - q_{3V}$, $p_{DiA} = 1 - q_{DiB} - q_{DiC}$, $i = 1,.2$
$p_{D3B} = 1 - q_{D3C} - q_{D3A}$, $p_{DiC} = 1 - q_{DiA} - q_{DiB}$, $i = 4,.5$

Fig. 7. M3—Detailed NVP model without fault treatment.

Due to the fact that, 1) the versions are executed in parallel and 2) the decision of acceptance of the current execution and selection of the "best" result is made on a relative basis, the dependability analysis of NVP requires that the interactions between the faults in the versions and the faults in the decider, as well as their consequences, be precisely identified. Thus, as for RB, we consider the following assumptions:

A6) Only a single fault type may manifest during the execution of the versions.

A7) Only a single fault type may manifest during the whole NVP software execution (versions and decider) and no compensation may take place between the errors of the versions and of the decider.

1) Detailed NVP Model: The behavior of NVP when considering assumption A6 is described by model M3 shown in Fig. 7.

State V is the state when the versions are executed. States Di, correspond to the execution of the decider. Based on A3 and on the impact of the evaluation of acceptable, distinct or similar erroneous results on dependability, five cases are distinguished:

D1) no fault activation [p]; the versions provide three acceptable results,

D2) activation of an independent fault in 1 version [$3q_{IV}(1-q_V)^2 \approx 3q_{IV}$]; the versions provide two acceptable results,

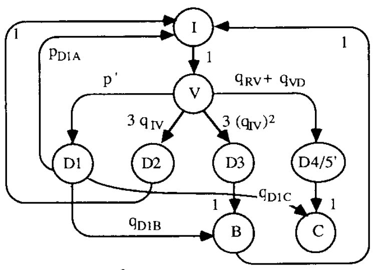
D3) activation of independent faults in two or three versions [$3(q_{IV})^2(1-q_V)+(q_{IV})^3 \approx 3(q_{IV})^2$]; the versions give three distinct results,

D4) activation of related faults in two versions [$3q_{2V}$]; the versions provide two similar erroneous results,

D5) activation of related faults in the three versions [q_{3V}]; the versions provide three similar erroneous results.

From these states, the nominal (fault-free) behavior [p_{DiA}] resulting from the execution of the decider, leads to a transition from

- $D1$ & $D2$ to I, since the decider evaluates three or two acceptable results,
- $D3$ to B, since the decider evaluates three distinct results,
- $D4$ & $D5$ to C: the decider evaluates two or three similar erroneous results.



$p' = 1 - 3q_{IV} - 3q_{IV}^2 - q_{RV} - q_{VD}$, $p_{D1A} = 1 - q_{D1B} - q_{D1C}$

Fig. 8. M4—Simplified NVP model without fault treatment.

Considering decider faults, leads to the following singular events:

- *error compensation:* the decider delivers an acceptable result when evaluating, at least two distinct results (state $D3$),[3] two (state $D4$) or three (state $D5$) similar erroneous results, which leads to state I; the associated probabilities are denoted q_{DiA},
- *rejection of an execution* although at least two similar results are provided by the versions (states $D1$, $D2$, $D4$, and $D5$),[4] which leads to state B: the associated probabilities are denoted q_{DiB},
- *delivery of an erroneous output result* when evaluating, either at least two acceptable, or at least two distinct erroneous results; (states $D1$, $D2$, $D4$, and $D5$)[5] leading to state C, the associated probabilities are denoted q_{DiC}.

As the decision made by the decider is essentially relative, its efficiency depends rather on the *similar/distinct* than on the *acceptable/erroneous* aspects of the results to be evaluated; thus, the following assumptions can be considered in practice to simplify model M3:

A8) The decider is not able to discriminate similar acceptable results from similar erroneous results, thus: $q_{D1B} \approx q_{D5B}$ and $q_{D2B} \approx q_{D4B}$.

A9) The decider has the same nominal behavior (it provides a common output result) when evaluating either two (majority) or three similar results; accordingly:

$p_{D1A} \approx p_{D2A}$, $q_{D1B} \approx q_{D2B}$, and thus $q_{D1C} \approx q_{D2C}$,
$p_{D4A} \approx p_{D5A}$, $q_{D4B} \approx q_{D5B}$, and thus $q_{D4C} \approx q_{D5C}$.

2) Simplified NVP Model: The corresponding model (M4) can be directly derived from the analysis of the NVP production process [Fig. 6(a)] and is shown on Fig. 8.

States $D1$, $D2$, and $D3$ are equivalent to related states of M3. State $D4/5'$ corresponds to the activation of related faults either 1) among the versions (merging of states $D4$ and $D5$ from M3 [$q_{RV} = 3q_{2V} + q_{3V}$]), or 2) between the three versions and the decider [q_{VD}] (Fig. 7).

In this case, q_{VD} includes all the interactions between the faults of the versions and of the decider and thus, the impact

[3]This would take place, for example, in the case of a median-based decision when the erroneous results are placed on each side of the acceptable result.

[4]The decider is too "tight;" this results in a reliability penalty, in the case when the similar results correspond to acceptable results.

[5]This case does not correspond to the case when the decider is evaluating at least two similar erroneous results. The singularities correspond here to the cases—hopefully rare!—when the decider outvotes acceptable results or when the decider is too "loose."

of the activation of an independent fault in the decider is considered only for state $D1$ with probability $q_D = q_{D1B} + q_{D1C}$. For states $D2$, $D3$, and $D4/5'$, the description is limited to the nominal (fault-free) behavior of the decider.

3) Processing of the Models:

For model M3,

$$\Gamma_R = \sigma\{(p + 3q_{IV})(q_{D1B} + q_{D1C}) + q_{RV}(p_{D4C} + q_{D1B}) + 3(q_{IV})^2(p_{D3B} + q_{D3C})\} \quad (13)$$

$$\Gamma_S = \sigma\{(p + 3q_{IV})q_{D1C} + q_{RV}p_{D4C} + 3(q_{IV})^2 q_{D3C}\}. \quad (14)$$

For model M4, we have

$$\Gamma_R = \sigma\{p' q_D + q_{RV} + q_{VD} + 3(q_{IV})^2\} \quad (15)$$

$$\Gamma_S = \sigma\{p' q_{D1C} + q_{RV} + q_{VD}\} \quad (16)$$

for which the expressions below are close pessimistic approximations:

$$\Gamma_R = \sigma\{q_D + q_{RV} + q_{VD} + 3(q_{IV})^2\} \quad (15')$$

$$\Gamma_S = \sigma\{q_{D1C} + q_{RV} + q_{VD}\}. \quad (16')$$

It is worth noting that the same expressions can be obtained from M3, 1) when there is no compensation [$q_{D3A} = q_{D4A} = 0$] and 2) noting that expression [$3q_{IV}q_{D1C} + 3(q_{IV})^2 q_{D3C}$] obtained in (13) and (14) can be identified to the probability of activation of a related fault in the versions and in the decider [q_{VD}] from (15) and (16). Indeed, (13) and (14) write as

$$\Gamma_R = \sigma\{(p + 3q_{IV} + q_{RV})q_{D1B} + pq_{D1C} + q_{RV}p_{D4C} + q_{VD} + 3(q_{IV})^2 p_{D3B}\} \quad (13')$$

$$\Gamma_S = \sigma\{pq_{D1C} + q_{RV}p_{D3C} + q_{VD}\} \quad (14')$$

for which it can be verified that (15′) and (16′) constitute valid approximations. Accordingly, further analyses will be carried out considering essentially these approximate expressions.

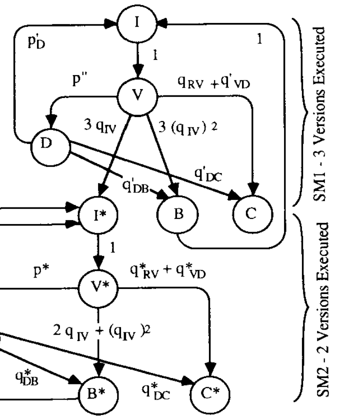
4) Comparison to a Nonfault-Tolerant Software: From (12) the failure rate of the nonfault-tolerant software corresponding to the selection of any version is expressed as

$$\Gamma'_R = \sigma q_V = \sigma\{q_{3V} + 2q_{2V} + q_{VD} + q_{IV}\} = \sigma\{q'_{RV} + q_{VD} + q_{IV}\} \quad (17)$$

where q'_{RV} corresponds to the probability of activation of the faults in the selected version that could be mapped to related faults in the other two versions in the NVP software.

The ratio $r = \Gamma_R/\Gamma'_R$ is then

$$r = \frac{q_D + q_{RV} + q_{VD} + 3(q_{IV})^2}{q'_{RV} + q_{VD} + q_{IV}}. \quad (18)$$



$p'' = 1 - 3\,q_{IV} - 3\,(q_{IV})^2 - q_{RV} - q'_{VD}$, $p'_D = 1 - q'_{DB} - q'_{DC} = 1 - q'_D$

$p^* = 1 - 2\,q_{IV} - (q_{IV})^2 - q^*_{RV} - q^*_{VD}$, $p^*_D = 1 - q^*_{DB} - q^*_{DC} = 1 - q^*_D$

Fig. 9. M5—NVP model with fault treatment.

For the comparison, we introduce the ratio i identifying the proportion of independent faults in the selected version:

$$q_{IV} = iq_V = i(q'_{RV} + q_{VD} + q_{IV}). \quad (19)$$

It follows that

$$r = \frac{q_D}{q_V} + \frac{q_{RV} + q_{VD}}{q_V} + 3(i)^2 q_V. \quad (20)$$

As $q_{RV} = q_{3V} + 3q_{2V}$ and $q'_{RV} = q_{3V} + 2q_{2V}$, we have $q_{RV} \approx q'_{RV}$, when related faults in three versions dominate (specification), and $q_{RV} \approx (3/2)q'_{RV}$, when related faults in two versions dominate (implementations). Thus, r is comprised into domain determined by the lower (r') and upper (r'') bounds:

$$r' = q_D/q_V + (1 - i) + 3(i)^2 q_V \quad (21)$$

$$r'' = q_D/q_V + (3/2)(1 - i) + 3(i)^2 q_V. \quad (22)$$

These expressions enable us to quantify the following qualitative (and intuitive) results:

- the decider must be far more reliable than the versions,
- if related faults dominate ($i \approx 0$) no improvement has to be expected, which confirms the results obtained in a large number of previous studies, e.g., see [17] and [22].

B. NVP Model with Fault Treatment

In this case (since assumption A5 applies), a supplementary specification of the decider is to correctly diagnose a disagreeing version when two versions provide acceptable results.

1) Description of the Model: The corresponding model (M5) is shown on Fig. 9. Submodel SM1 is equivalent to model M4, the only differences concern

- the elimination of the states Di with an output transition equal to 1 by merging them with the next state,
- the modification of the probabilities of activation of a fault in the decider to account for the change in its specification, 1) change of q_{VD} into q'_{VD}[6] and 2) change of q_{D1i} into q'_{Di} ($i \in \{B, C\}$),

[6]In particular, q'_{VD} includes the risk of failure of the diagnosis of the decider: discarding a version providing an acceptable result.

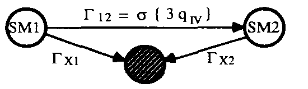


Fig. 10. M6—Model for reliability and safety analysis.

- when an independent fault has been activated in a single version, this version is discarded and thus SM2 is entered.

Submodel SM2 has the same structure as SM1; however, as only two versions are used, the probabilities of fault activation in the versions and in the decider are modified in accordance; in particular, $q^*_{RV} = q_{2V}$.

2) Processing of the Model: As in model M5 the non-failed states do not constitute an irreducible Markov chain, it is not possible to obtain equivalent failure rates. However, equivalent failure rates may be derived for each submodel in isolation:

for SM1:

$$\Gamma_{R1} = \sigma\{q'_D + q_{RV} + q'_{VD} + 3(q_{IV})^2\} \tag{23}$$

$$\Gamma_{S1} = \sigma\{q'_{DC} + q_{RV} + q'_{VD}\} \tag{24}$$

for SM2:

$$\Gamma_{R2} = \sigma\{q^*_D + q^*_{RV} + q^*_{VD} + 2q_{IV} + (q_{IV})^2\} \tag{25}$$

$$\Gamma_{S2} = \sigma\{q^*_{DC} + q^*_{RV} + q^*_{VD}\}. \tag{26}$$

Reliability and safety of the model can then be analyzed by processing model M6 of Fig. 10, where $X = R$ for reliability and $X = S$ for safety.

Reliability and safety and the associated mean times to failure express as

$$\begin{aligned} R(t) &= \exp[-(\Gamma_{12} + \Gamma_{R1})t] + \frac{\Gamma_{12}}{\Gamma_{12} + \Gamma_{R1} - \Gamma_{R2}} \\ &\quad \cdot \{\exp[-\Gamma_{R2}t] - \exp[-(\Gamma_{12} + \Gamma_{R1})t]\} \\ S(t) &= \exp[-(\Gamma_{12} + \Gamma_{S1})t] + \frac{\Gamma_{12}}{\Gamma_{12} + \Gamma_{S1} - \Gamma_{S2}} \\ &\quad \cdot \{\exp[-\Gamma_{S2}t] - \exp[-(\Gamma_{12} + \Gamma_{S1})t]\} \end{aligned} \tag{27}$$

$$\begin{aligned} \text{MTTF}_R &= \frac{\Gamma_{12}}{\Gamma_{12} + \Gamma_{R1}} \left(\frac{1}{\Gamma_{R2}} - \frac{1}{\Gamma_{R1}}\right) + \frac{1}{\Gamma_{R1}} \\ \text{MTTF}_S &= \frac{\Gamma_{12}}{\Gamma_{12} + \Gamma_{S1}} \left(\frac{1}{\Gamma_{S2}} - \frac{1}{\Gamma_{S1}}\right) + \frac{1}{\Gamma_{S1}}. \end{aligned} \tag{28}$$

Analysis of these expressions requires that the rates be precisely evaluated which is a rather difficult task due to the uncertainty in the values of the probabilities from which they are derived. Nevertheless, interesting results can be obtained with the following assumptions:

A10) The decider has the same behavior when evaluating either three versions (SM1) or two versions (SM2), i.e., $q'_{Di} \approx q^*_{Di}$ and $q'_{VD} \approx q^*_{VD}$.

A11) Due to the intrinsic simplicity of the algorithm involved, the behavior of the decider is not significantly altered when its specifications are modified from assumption A4 (model M4) to assumption A5 (model M5), i.e., $q_{D1i} \approx q'_{Di}$ and $q_{VD} \approx q'_{VD}$.

According to A10, it can be verified from relations (23) to (26), that

$$\Gamma_{R1} = \sigma\{q'_D + q_{3V} + 3q_{2V} + q'_{VD} + 3(q_{IV})^2\} \tag{29}$$

$$\Gamma_{S1} = \sigma\{q'_{DC} + q_{3V} + 3q_{2V} + q'_{VD}\} \tag{30}$$

$$\Gamma_{R2} = \sigma\{q'_D + q_{2V} + q'_{VD} + 2q_{IV} + (q_{IV})^2\} \tag{31}$$

$$\Gamma_{S2} = \sigma\{q'_{DC} + q_{2V} + q'_{VD}\}. \tag{32}$$

These relations show that $\Gamma_{S1} > \Gamma_{S2}$ and that

- $\Gamma_{R1} < \Gamma_{R2}$, when independent faults in the versions dominate,
- $\Gamma_{R1} > \Gamma_{R2}$, when related faults among the versions dominate.

Expressions (28) show that the decision to discard the disagreeing version improves MTTF_S, which is not always the case for MTTF_R.[7] Analogous conclusions can be derived from expressions (27) for $S(t)$ and $R(t)$.

More generally, the expressions confirm a general system reliability result, i.e., it is better to use two versions than three versions, when emphasis is put on safety rather than on reliability. Furthermore, in the case of reliability, the impact of related faults is clearly indicated by the fact that no improvement has to be expected when using three versions instead of two if related faults among the versions dominate significantly over independent faults [22].

IV. RB and NVP Comparison

In order to be homogeneous, the comparison is carried out only when

- assumptions A2 and A7 hold (a single fault type activation and no error compensation within the whole software),
- no specific fault treatment is considered for NVP.

For each architecture, specific notations have been used. However, similar expressions can be derived for Γ_R and Γ_S, based on the following notation:

q_I = Prob{independent failure of one variant|execution}, thus: $q_{I,\text{RB}} = q_P \approx q_S$ and $q_{I,\text{NVP}} = q_{IV}$

q_{CM} = Prob{common-mode failure|execution}, thus: $q_{\text{CM,RB}} = q_{PST} = q_{PS} + q_T$ and $q_{\text{CM,NVP}} = q_{VD} + q_{RV} + q_D$.

Accordingly, relations (7) with $q_T \ll 1$ and (15′) become, respectively,

$$\text{for RB:} \quad \Gamma_R = \sigma\{q_{\text{CM,RB}} + (q_{I,\text{RB}})^2\} \tag{33}$$

$$\text{for NVP:} \quad \Gamma_R = \sigma\{q_{\text{CM,NVP}} + 3(q_{I,\text{NVP}})^2\}. \tag{34}$$

Although the form of these expressions would suggest that RB is better than NVP, it has to be noted that the influence of the various terms may be different. Indeed, if the probabilities of activation of 1) an independent fault in one variant [q_P (or q_S) and q_{IV}] and 2) of related faults between variants [q_{PS}

[7]Indeed, $1/\Gamma_{R1}$ (resp., $1/\Gamma_{S1}$) can be interpreted as the MTTF_R (resp., MTTF_S) of the NVP software when no fault treatment is carried out (model M4).

and q_{RV}], are of the same order of magnitude, however, the probabilities of activation of 1) an independent fault in the decider [q_T and q_D] and 2) of related faults between the variants and the decider [q_{PST} and q_{VD}] are likely to be greater for RB than for NVP; this is mainly due to the fact that the AT is specific to each application, whereas the decider in NVP is generic to a large extent.

It follows that independent failures for the variants have more impact for the NVP, whereas the impact of the decider is lower for this architecture. However, a precise knowledge of these probabilities would be needed in order to perform a more detailed comparison.

Considering safety analysis, we have obtained

$$\text{for RB: } \Gamma_S = \sigma q_{PST} \tag{35}$$

$$\text{for NVP: } \Gamma_S = \sigma\{q_{RV} + q_{VD} + q_{DIC}\} \tag{36}$$

Related faults among variants have no influence for RB, but are of prime importance for NVP. This is a consequence of the fact that for RB an absolute decision is taken for each alternate against the specification and that for NVP the decision is made on a relative basis among the results provided by the versions.

Due to the very nature of the NVP decider, q_{DIC} may be made very low and as $q_{VD} \ll q_{RV}$, q_{RV} has to be compared with q_{PST}.

Finally it is worth noting that only partial conclusions can be drawn from this analysis. Additional features need to be taken into account, such as the fact that, for RB, service delivery is suspended during error recovery, i.e., when the secondary is invoked.

V. Nested Recovery Blocks

This section provides a preliminary extension of the analysis of the RB architecture to account for the specific case of nested RB's. Nested RB's are very interesting and give rise to stimulating discussions, but have received little treatment from the modeling point of view. We simply illustrate here, how a simple case of nested RB's can be handled with the modeling and evaluation approach presented. Let us consider for example that *P* is itself a RB; *P* will be called the *nested block* (NB).

The production process of a RB [Fig. 2(a)] is still the same, but box 4 becomes *specification of the NB*, and it is decomposed into an equivalent production process as for the original RB (Fig. 11).

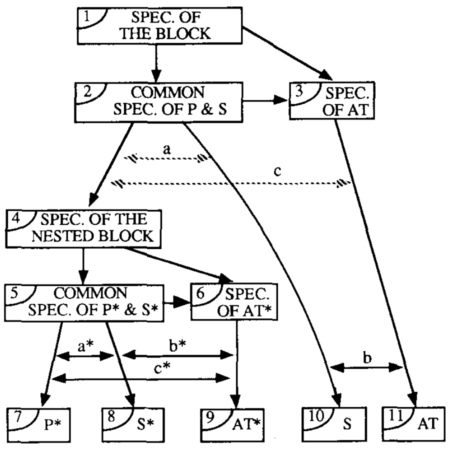


Fig. 11. Nested RB.

It can be seen that related faults due to the specification still remain; on the contrary related faults introduced, during separate implementations (channels *a*, *b*, *c*) are moved to the lower level, i.e., between the alternates composing the NB and the associated AT, *S*, and the original AT.

Independent faults in *S* are not changed either, but independent faults in *P* are split into related and independent faults in the NB.

It follows that the common-mode failure probability (q_{CM}) must be split into *specification* and *implementation* common-mode failure probabilities denoted (q_{CMS}) and (q_{CMI}), respectively. q_{CMS} is the same for the original and the nested RB's, but the probability of common-mode failure due to implementation faults has to be distinguished for the original (q_{CMI}) and the nested (q^*_{CMI}) RB.

The new equivalent failure rate is deduced directly from (33) as

$$\Gamma^*_R = \sigma\{q_{CMS} + q^*_{CMI} + [q'_{CM} + (q'_I)^2]q_I\} \tag{37}$$

where q'_{CM} and q'_I denote, respectively, the probabilities of common-mode and independent failure of the nested block.

The important question is: how does Γ^*_R compare to Γ_R? For this concern, it is worth noting that NB's correspond to 1) a step in the decomposition of the complexity of the software and 2) an increase in the redundancy, and as such it can be expected that the reliability be improved as shown in [7] where the reliability of an RB with two and more alternates is evaluated.

Formula (37) can be easily generalized to the cases of 1) successive NB's, 2) NB's in both *P* and *S*, and 3) more than one NB in each alternate.

VI. Conclusion

The paper presented a detailed *reliability and safety analysis* of the two major software fault-tolerance approaches: RB and NVP.

The methodology used for *modeling* is based on 1) the identification of the possible types of faults introduced during the specification and the implementation, 2) the analysis of the behavior following fault activation.

An important comment concerns the *fault assumption* used in the modeling. The most significant issue for evaluation of diversified software in operation concerns the types of errors (distinct or similar) that result from the activation of faults. We considered a direct mapping of these errors with two distinct fault classes: independent and related faults. Such a mapping is pessimistic as it enables us to incorporate only positive correlation in the manifestation of related faults. As recently evidenced in [6], another form of correlation (negative correlation) does exist among related faults which has a beneficial consequence on the execution of multivariant software in forcing the delivery of distinct errors. However, although they could be traced to (negatively correlated) related faults, faults leading to distinct errors are not distinguishable at the execution level from independent faults and thus they both can be merged into a single category: the independent faults. A dual discussion applies also to similar errors since in some—very rare (e.g., see [3])—cases they also could be traced to independent faults. It is worth noting that the detailed analysis of the relationship between classes of errors and faults would result in a further increase in the—already large—number of parameters of the models. In addition, owing to the prominent influence of the deciders in the failure process of a fault-tolerant software, such an analysis should not be limited to examining the intervariant correlations, but should cover the positive and negative correlations between the variants and the deciders, as well.

The main outcome of the *evaluation* carried out concerns the derivation of analytical results enabling us 1) to identify the conditions of improvement, when compared to a nonfault-tolerant software, that could result from the use of RB (the acceptance test has to be more reliable than the alternates) and NVP (related faults among the versions and the decider have to be minimized) and 2) to reveal the most critical types of related faults. In particular, for safety, the related faults between the variants have a significant impact for NVP, whereas only related faults between the alternates and the acceptance test have to be considered for RB.

The study of the *nested RB's* showed that 1) the proposed analysis approach can be applied to such realistic software structures and 2) when an alternate is itself an RB, the results are analogous to the case of the addition of a third alternate. The reliability analysis showed that an improvement has to be expected, but that this improvement would be very low.

The specific study of the *discarding of a failed version in NVP* showed that this strategy is always worthwhile for safety, whereas, for reliability, it is all the more beneficial as independent faults dominate.

References

[1] T. Anderson, P. A. Barrett, D. Halliwell, and M. R. Moulding, "Software-fault tolerance: An evaluation," *IEEE Trans. Software Eng.*, vol. SE-11, no. 12, pp. 1502-1510, Dec. 1985.

[2] J. Arlat, K. Kanoun, and J.-C. Laprie, "Dependability evaluation of software fault-tolerance," in *Proc. FTCS-18*, June 1988, pp. 142-147.

[3] A. Avizienis and J. P. J. Kelly, "Fault-tolerance by design diversity: Concepts and experiments," *IEEE Comput. Mag.*, pp. 67-80, Aug. 1984.

[4] A. Avizienis, P. Gunninberg, J. P. J. Kelly, R. T. Lyu, L. Strigini, P. J. Traverse, K. S. Tso, and U. Voges, "Software fault-tolerance by design diversity—DEDIX: A tool for experiments," in *Proc. SAFECOMP'85*, Como, Italy, Oct. 1985, pp. 173-178.

[5] A. Avizienis and J. C. Laprie, "Dependable computing: From concepts to design diversity," *Proc. IEEE*, vol. 74, no. 5, pp. 629-638, May 1986.

[6] P. G. Bishop and F. D. Pullen, "Error masking: A source of failure dependency in multiversion programs," in *Proc. 1st Int. Working Conf. Dependable Comput. Critical Appl.*, Santa Barbara, CA, Aug. 1989, pp. 25-32.

[7] S. D. Cha, "A recovery block model and its analysis," in *Proc. SAFECOMP'86*, Sarlat, France, Oct. 1986, pp. 21-26.

[8] L. Chen and A. Avizienis, "N-Version programming: A fault-tolerance approach to reliability of software operation," in *Proc. FTCS8*, Toulouse, France, June 1978, pp. 3-9.

[9] RC. Cheung, "A user-oriented software reliability model," *IEEE Trans. Software Eng.*, vol. SE-6, no. 2, pp. 118-125, Mar. 1985.

[10] A. Csenki, "Recovery block reliability analysis with failure clustering," in *Proc. 1st Int. Working Conf. Dependable Comput. Critical Appl.*, Santa Barbara, CA, Aug. 1989, pp. 33-42.

[11] D. E. Eckhardt and L. D. Lee, "A theoretical basis for the analysis of multiversion software subject to coincident errors," *IEEE Trans. Software Eng.*, vol. SE-11, no. 12, pp. 1511-1517, Dec. 1985.

[12] W. Feller, *An Introduction to Probability Theory and its Application, Vol. I.* New York: Wiley, 1968.

[13] J. N. Gray, "Why do computers stop and what can be done about it?," in *Proc. 5th SRDSDS*, Los Angeles, CA, Jan. 1986, pp. 3-12.

[14] A. Grnarov, J. Arlat, and A. Avizienis, "On the performance of software fault-tolerance strategies," in *Proc. FTCS10*, Kyoto, Japan, Oct. 1980, pp. 251-253.

[15] H. Hecht, "Fault tolerant software," *IEEE Trans. Reliability*, vol. R-28, no. 3, pp. 227-232, Aug. 1979.

[16] J. C. Knight, N. G. Leveson, and L. D. St. Jean, "A large scale experiment in N-version programming," in *Proc. FTCS15*, Ann Arbor, MI, July 1985, pp. 135-139.

[17] J. C. Knight and N. G. Leveson, "An empirical study of failure probabilities in multi-version software," in *Proc. FTCS16*, Vienna, Austria, July 1986, pp. 165-170.

[18] J.-C. Laprie, "Dependability evaluation of software systems in operation," *IEEE Trans. Software Eng.*, vol. SE-10, no. 6, pp. 701-714, Nov. 1984.

[19] J.-C. Laprie, "Dependability computing and fault-tolerance: Concepts and terminology," in *Proc. FTCS15*, Ann Arbor, MI, July 1985, pp. 2-11.

[20] J.-C. Laprie, J. Arlat, C. Beounes, K. Kanoun, and C. Hourtolle, "Hardware- and software-fault tolerance: Definition and analysis of architectural solutions," in *Proc. FTCS17*, Pittsburgh, PA, July 1987, pp. 116-121.

[21] B. Littlewood, "Software reliability model for modular program structure," *IEEE Trans. Reliability*, vol. R-28, no. 3, pp. 241-246, Aug. 1985.

[22] B. Littlewood and D. R. Miller, "A conceptual model of multi-version software," in *Proc. FTCS17*, Pittsburgh, PA, July 1987, pp. 150-155.

[23] P. R. Lorczak, A. K. Caglayan, and D. E. Eckhardt, "A theoretical investigation of generalized voters for redundant systems," in *Proc. FTCS-19*, Chicago, IL, June 1989, pp. 444-451.

[24] M. Mulazzani, "Reliability versus safety," in *Proc. SAFECOMP'85*, Como, Italy, Oct. 1985, pp. 141-1146.

[25] A. Pagès and M. Gondran, *Fiabilité des systèmes.* France: Eyrolles, 1980.

[26] B. Randell, "System structure for software fault tolerance," *IEEE Trans. Software Eng.*, vol. SE-1, no. 2, pp. 220-232, June 1975.

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

[28] R. K. Scott, J. W. Gault, and D. F. McAllister, "Fault-tolerant software reliability modeling," *IEEE Trans. Software Eng.*, vol. SE-13, pp. 582-592.

[29] K. S. Tso, A. Avizienis, and J. P. J. Kelly, "Error recovery in multiversion software," in *Proc. SAFECOMP'86*, Sarlat, France, Oct. 1986, pp. 35-41.

Jean Arlat (M'80) was born in Toulouse, France, in 1953. He received the Certified Engineer degree from the National Institute of Applied Sciences of Toulouse (INSAT) in 1976 and the Doctor-Engineer degree from the National Polytechnic Institute of Toulouse (INPT) in 1979.

He is Chargé de Recherche with the National Center of Scientific Research (CNRS) and member of the group Dependable Computing and Fault Tolerance of the Laboratory for Automatics and Systems Analysis (LAAS) of the CNRS that he first joined in 1976. From September 1979 to August 80, he spent a postdoctoral year in the Computer Science Department of the University of California, Los Angeles, (UCLA), working on the dependability and performance evaluation of software fault tolerance architectures. Since 1980 he has been a Research Staff Member at LAAS-CNRS. His research interests focus on the evaluation of hardware-and-software fault-tolerant systems including both analytical modeling and experimental fault injection approaches.

Dr. Arlat is a member of the Fault-Tolerant Computing and Simulation Technical Committees of the IEEE Computer Society. He is also a member of the AFCET Working Group Dependability of Computing Systems and of the French branch of the Computer Measurement Group (CMGF).

Karama Kanoun received the Certified Engineer degree from National School of Civil Aviation, Toulouse, France in 1977, the Doctor-Engineer degree and the Doctor-ès-Science degree from the National Polytechnic Institute of Toulouse in 1980 and 1989, respectively.

She is currently Chargée de Recherche at CNRS. She joined LAAS in 1977 as a member of the Fault-Tolerance and Dependable Computing group. Her current research interests include modeling and evaluation of computer system dependability considering hardware as well as software aspects. She has conducted several research contracts and she has been a consultant for some French companies and for the International Union of Telecommunications.

Dr. Kanoun is a member of the working group of the European Workshop on Industrial Computer Systems (EWICS): "Technical Committee 7-Reliability, Safety and Security" and a member of the AFCET working group Dependability of Computing Systems.

Jean-Claude Laprie (M'90) received the Certified Engineer degree from the Higher National School for Aeronautical Constructions, Toulouse, France, in 1968, the Doctor in Engineering degree in automatic control, and the Doctor ès-Sciences degree in Computer Science from the University of Toulouse, in 1971 and 1975, respectively.

He is currently Directeur de Recherche of CNRS, the National Organization of Scientific Research. He joined LAAS 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 authored and coauthored more than 50 papers; he is the Principal Investigator of several contracts in these areas of interest. From January to August 1985, he was an Invited Visiting Professor at the UCLA Department of Computer Science, Los Angeles. He has also acted as a consultant and as an expert in the area of dependable computing in France and abroad for government agencies as well as industrial organizations.

Dr. Laprie served in 1978 as the General Chairman of the 8th International Symposium on Fault-Tolerant Computing, and on program committees for numerous conferences and workshops. He was the Chairman of the IEEE Computer Society's Technical Committee on Fault-Tolerant Computing in 1984 and 1985. He is a founding member of the IFIP Working Group on Reliable Computing and Fault Tolerance, which he is presently chairing. He is the founding Chairman of the AFCET (French Association for Economics and Techniques of Cybernetics) Group on Computing Systems Dependability. He is co-editor of the Springer Verlag series on Dependable Computing and Fault Tolerant Systems. He is a member of the Association for Computing Machinery and AFCET.