Fault Injection for Dependability Validation: A Methodology and Some Applications

JEAN ARLAT, MEMBER, IEEE, MARTINE AGUERA, LOUIS AMAT, YVES CROUZET, JEAN-CHARLES FABRE, JEAN-CLAUDE LAPRIE, MEMBER, IEEE, ELIANE MARTINS, AND DAVID POWELL

Abstract—**This paper addresses the problem of the dependability validation of fault-tolerant computing systems and more specifically the validation of the fault-tolerance mechanisms. The presented approach is based on the use of fault-injection at the physical level on a hardware/software prototype of the considered system. The place of this approach in a validation directed design process, as well as its place with respect to related works on fault-injection, is clearly identified. The major requirements and problems related to the development and application of a validation methodology based on fault injection are presented and discussed. Emphasis is put on the definition, analysis, and use of the experimental dependability measures that can be obtained. The proposed methodology has been implemented through the realization of a general pin-level fault injection tool (MESSALINE) and its usefulness is demonstrated by the application of MESSALINE to the experimental validation of two systems: 1) a subsystem of a centralized computerized inerlocking system for railway control applications and 2) a distributed system corresponding to the current implementation of the dependable communication system of the ESPRIT Delta-4 Project.**

Index Terms—**Coverage evaluation, experimental dependability validation, fault/error injection, fault simulation, fault-tolerant computing systems.**

I. Introduction

ONE of the major problems related to the development of fault-tolerant computer systems concerns their validation and can be summarized by the following question: *How can one obtain confidence in the system's ability to deliver a proper service*? [1].

As it is not possible to wait and get statistical data from field operation of the system or from equivalent systems, it is necessary that preliminary studies of the behavior of the system in presence of anticipated faults be carried out during all phases of the development process. Accordingly, the fault injection method that is addressed by this paper constitutes an invaluable approach in the validation process: fault injection can be seen as a technique for testing a fault-tolerant system with respect to a class of inputs specific to such a system, i.e., the faults. Nevertheless, it should be noted that the presented experimental fault injection methodology complements rather than competes with other validation approaches (such as modeling, simulation, etc.) and its application is aimed at increasing the confidence in the validation process.

This paper constitutes an elaboration of the work presented in [2]; in particular, the application section is extended to provide a larger set of results. The paper is organized as follows. Section II presents the place of fault injection in the development process with a special emphasis on the abstraction level of the model of the system on which fault injection is applied and on the characterization of the goals of fault injection: validation and design-aid. Section III gives a statistical characterization of experimental evaluation based on fault injection and shows how experimental evaluation can be used to obtain dependability measures. Section IV presents the main features of a general tool—MESSALINE—for the application of fault injection at the pin level on a hardware/software prototype of the system. Section V gives some results of the application of MESSALINE to the experimental validation of two systems: 1) a subsystem of a computerized interlocking system for railway control applications and 2) a distributed system corresponding to the current implementation of the dependable communication system of the ESPRIT Delta-4 Project. Some concluding remarks are given in Section VI.

II. Fault Injection in the Development Process

Fault injection, i.e., the deliberate introduction of faults into a system—the target system—is applicable every time fault and/or error notions are concerned in the development process.

When fault injection is to be considered on a target system, the input domain corresponds to a set of faults $\boldsymbol{F}$ and a set of activations $\boldsymbol{A}$ that specifies the domain used to functionally exercise the system and the output domain corresponds to a set of readouts $\boldsymbol{R}$ and a set of derived measures $\boldsymbol{M}$. Together, the $\boldsymbol{FARM}$ sets constitute the major attributes that can be used to fully characterize fault injection. In this section, we focus on the impact of the levels of abstraction and of the goals of fault injection on these attributes.

Manuscript receipt April 1, 1989; revised October 4, 1989. Recommended by R. K. Iyer. This work was supported in part by SNCF (French Railways) under Contract 0540300013, the Commission of European Community (ESPRIT) under Contract P818/P2252 Delta-4), and by the Conseil Régional de Midi-Pyrénées.

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 8932225.

A. Levels of Abstraction of Fault Injection

1) Identification of the Levels of Abstraction: Various levels of abstraction may be identified for the application of fault injection, depending on the type of model used for the target system. For this presentation we use the classification identified in [3] that distinguishes three major types of models:

- *axiomatic* models, examples of which are the analytical models used to model the structure and the dependability and/or performance behavior of the system such as reliability block diagrams, fault trees, Markov graphs or stochastic Petri nets, etc.;
- *empirical* models, corresponding to models that incorporate more complex or detailed behavioral and/or structural descriptions that require specific cases analysis (i.e., a simulation approach) to process them;[1]
- *physical* models, corresponding to prototypes actually implementing the hardware and/or software features of the developed system.

In the sequel, we emphasize the impact of these three types of models on the ***FARM*** attributes.

2) The F Set: In axiomatic models, the ***F*** set is described by stochastic processes whose parameters are characterized by probabilistic distributions. In practice, for sake of simplicity in the processing of the model, the considered processes are generally restricted to Poisson processes. The ***F*** set may cover both hardware and software faults, but most of the work has been carried out dealing mainly with a specific class of hardware faults: the physical faults.

Empirical models enable one to relax some of the assumptions and restrictions associated with axiomatic models. In particular, they enable one to use more realistic distributions for the parameters of the ***F*** set. Examples of the use of such models correspond to the fault simulation methods at component, gate, circuit or system level and may be deterministic (see, e.g., [4]–[7]) or of the Monte Carlo type (see, e.g., [8], [9]). The freedom in the description level favors the use of a hierarchical approach in the description of the system. In particular, in the case of hardware, basic fault models at the gate level (stuck-at) can be considered as well as more subtle fault cases at the transistor level (stuck-open). Although most of the work in this area is devoted to physical-fault simulation, the method presented in [10] to facilitate the mutation of instrumented programs is an example of application of software-fault simulation.[2] The emergence of rapid prototyping techniques may also provide a basis to facilitate such studies.

[1] Accordingly, it should be noted that it is the processing of the models that is empirical rather than their construction.

[2] It is worth noting that the validity of software-fault injection could be questionable since, on the contrary to hardware, the injected faults are readily distinct from the actual faults. However, initiators of the mutation testing approach contend that, based on the assumption that simple faults are more difficult to identify than complex and subtle ones (the so-called coupling effect [10]), it is expected that significant experiments can be obtained when using an ***F*** set elaborated from simple fault models.

Physical models (prototypes) are needed to validate the actual system implementation. Three types of target systems can be distinguished at this level: i) software-only prototypes, ii) hardware-only prototypes, or iii) hardware-and-software prototypes.

The number of studies concerning software-only prototypes is rather limited and the mutant programs are obtained from an ***F*** set made up of simple alterations (e.g., removal of a statement, permutation of truncation and roundoff, etc.) in the source code of programs of small size: typically a few hundred statements (see, e.g., [11]–[13]).

The second and third form of target systems, correspond to the application of physical fault-injection methods and the ***F*** set is mainly based on physical faults. A large number of studies have been published on physical fault-injection were the target system can be i) a complete fault tolerant system (e.g., [14]–[18]), ii) a more specific hardware-implemented fault tolerance mechanism (e.g., [19], [20]), or iii) standard IC's (e.g., [21], [22]). In all these studies, except for [22] that uses heavy-ion radiation, the faults are injected on the pins of the IC's.

With the exception of the simulation-based approaches presented in [5] and [6], most of the fault injection studies have been developed on an ad hoc basis for a specific target system. The goal of our work was mainly to provide a more methodological approach to physical fault injection and to facilitate its application to several target systems.

3) The A Set: In axiomatic models, the ***A*** set can also—as for the ***F*** set—be described by stochastic processes. However, due to the dynamic range (several orders of magnitude) of the time-related parameters of the ***A*** set and of the parameters of the ***F*** set, it can be considered that the processes of the ***A*** set stabilize faster than those of the ***F*** set and the description of the ***A*** set is often omitted.

In the case of empirical models, the ***A*** set describes the behavior of the system in a form where elementary parameters can be more appropriately identified and assigned.

In the case of software-only and hardware-only prototypes, the ***A*** set consists of a set of test data patterns aimed at exercising the injected faults. In the case of hardware-and-software prototypes, depending on the type of use of the target system under investigation (general purpose or real time process control) the determination of the ***A*** set may differ significantly. For general purpose systems, a solution consists of using a representative program. In the case of a real time process control system, the fault injection is seldom carried out on the application site and an environment simulator has thus to be provided.

4) The R and M Sets: In the case of axiomatic models, the ***M*** set may correspond to dependability measures such as reliability, MTFF, etc., which can be directly computed from an analytical processing of the model.

For both empirical and physical models, the measures in ***M*** can be obtained only experimentally from a series of fault injection case studies. A *fault injection test se-*

quence is made up of a set of elementary injections; each elementary injection is called an *experiment*.

For each experiment, a fault f is selected in $\boldsymbol{F}$ and an activation trajectory a is described in $\boldsymbol{A}$. The reactions of the system are observed and form a readout r that fully characterizes the outcome of the experiment. An experiment is thus characterized by the triple $< f, a, r >$, where the readouts for each experiment r form a global set of readouts $\boldsymbol{R}$ for the test sequence and can be used to elaborate a measure in $\boldsymbol{M}$.

5) Relation Between the Models: In the case of fault tolerant systems, the axiomatic models should provide a means to account for the behavior of the fault tolerance mechanisms in response to the fault [23]. Although fault and performance parameters can be obtained from existing statistical data concerning the components of the system, the characterization of system behavior and assignation of values to coverage and execution parameters of the fault tolerance mechanisms is a much more difficult task due to the fact that, in the current state-of-the-art, these data are not *a priori* available and are specific to the system under investigation. Accordingly, empirical and/or physical models must be used to gather experimental data to clearly affirm or deny the hypotheses in selecting values for the parameters of the axiomatic model. This complementary use will be addressed in more detail in Section III.

The empirical models enable: i) a hierarchical approach, ii) the application at different stages of the development process and on different objects (hardware and software), and iii) arbitrary definition of the $\boldsymbol{F}$, $\boldsymbol{A}$, and $\boldsymbol{R}$ sets. However, for efficiency reasons, these models are in practice often specialized to describe only a part of the system under investigation: an object and/or a short period analysis. On the other hand, physical prototypes enable the actual implementation of the complete system, including hardware and application software interactions, to be tested.

An example of relation between the empirical and physical models is provided when dealing with the problem of the injection of internal faults of IC's in physical fault injection. Indeed, such a problem could be addressed by a two-level experimental strategy where i) a fault simulator is used to determine the output error patterns due to the internal IC fault, and ii) these patterns are then physically injected on the pins of the IC. The work on model extraction presented in [7] and the validation environment proposed in [24] are significant examples of multilevel approaches integrating axiomatic and various levels of empirical models.

B. Goals of Fault Injection

This section details first the major goals for the application of fault injection in the design process and then presents their influence on the ***FARM*** attributes.

1) Characterization of the Goals: Two complementary major goals can be identified as: validation and design aid.

a) Validation: In the validation process, the role of fault injection is related to the concept of coverage, i.e., the validation of the validation and can be synthesized by the following question: *how can one obtain confidence in the methods and mechanisms used to obtain the confidence in the system*? [1]. Thus, fault injection can be viewed as a means for testing these methods and mechanisms with respect to the inputs they have been designed to cope with: the faults.

In practice, the two main issues concern:

- the validation of the verification procedures (e.g., test sets) used to reveal faults during all the phases of the development process;
- the validation of the fault tolerance mechanisms (detection, recovery, etc.) aimed at achieving the dependability of the system in the operational phase.

Furthermore, fault injection participates in both aspects of the validation process, fault removal based on verification and fault forecasting based on evaluation [1]:

- with respect to fault removal, the analysis is essentially qualitative and is aimed at checking the adequacy of the verification procedures and of the fault tolerance mechanisms with respect to the considered fault assumptions;
- in the case of fault forecasting, fault injection allows a rating of the efficiency of the test procedures (test coverage) or the coverage and execution parameters of the fault tolerance mechanisms (coverage factor [23], fault dormancy and error latency [25], etc.)

b) Design Aid: Fault injection can be applied at various steps in the development process and in particular, the (negative) results of fault injection can be used to initiate feedback loops to improve the test procedures and the fault tolerance mechanisms. As such, fault injection can be considered as a design aid in their development, provided that it is applied in the early phases, i.e., on axiomatic or empirical models.

Another significant goal of fault injection as a design aid concerns the establishment of fault dictionaries that can be used to develop diagnosis procedures [26], [27].

In the remainder of this section, we identify the major differences induced on the ***FARM*** sets by the fault removal and fault forecasting goals.

2) The F Set: For fault removal, the $\boldsymbol{F}$ set is usually made up of a small number of specific faults that are identified according to design specification.

Furthermore, it is practically required that the experiments carried out be reproducible, i.e., the same fault produces the same readouts. In particular, this enables the adequacy of possible design changes to be checked; in practice, this favors experiments based on the injection of permanent faults.

In the case of fault forecasting, the main concern is that the $\boldsymbol{F}$ set corresponds to a representative statistical distribution among the possible faults and that a large number of faults be injected to guarantee a good confidence level in the measure.

3) The A Set: With respect to fault removal, two cases

have to be distinguished, depending on the fact that the tested procedure or mechanism is intended to:

- both activate the fault and make the error observable, in which case the ***A*** set is implicitly made up of the tested procedure;
- only process errors, in which case, it is more efficient that the ***A*** set be a test program maximizing the number of activated faults.

In the case of fault forecasting, it is necessary that the ***A*** set correspond to a simulation of the operational field activity of the system under test.

*4) The **R** and **M** Sets:* In the case of fault removal, the ***M*** set is essentially represented by a binary variable that characterizes the fact that a specified conjunction of predicates on the behavior of the system under test is true or not; it is expected that the tested predicates be true for all experiments. However, in order to provide the opportunity to improve the design when the predicates are not true, it is in practice useful to record for such experiments a wider set of data in the readout r for the purpose of diagnosis. It is worth noting that in order to be able to check for the possible design modifications, it is mandatory to associate with each readout r the corresponding (f, a) pair.

In the case of fault forecasting, the ***M*** set corresponds to probabilistic or statistical measures on:

- the occurrence of states characterized by predicate combinations,
- the duration into and between states.

The selection of the predicates and of the combinations to account for is usually based on an *a priori* model of system behavior; in practice, such a model may be modified by the measures actually observed. Besides the classical probabilistic dependability measures, examples of such measures are i) reliability estimates based on the estimation of the number of software faults [28], and ii) coverage ratios [15], [13] and timing distributions [29], [30] of the fault tolerance mechanisms of the target system. These measures are computed from the ***FAR*** sets resulting from all the experiments.

As shown above, the derivation of measures in the case of fault forecasting is more difficult than in the case of fault removal. Thus, we pay further attention to this problem in the next section.

III. Experimental Fault Forecasting Based on Fault Injection

This section covers the major aspects related to the fault forecasting aim of experimental fault injection introduced in Section II. First, the complementary use of experimental fault injection with axiomatic model-based fault forecasting is motivated by a brief discussion on reliability evaluation. The second subsection provides a statistical characterization of experimental fault-injection. The third subsection gives an example of use of the experimental fault forecasting measures to calibrate the coverage and time execution parameters incorporated in axiomatic dependability evaluation models and defines the associated statistical estimators. Finally, the fourth and last subsection briefly addresses the problem of the accuracy of experimental fault forecasting.

A. Reliability Evaluation and Experimental Fault Injection

1) Reliability of a Non-Fault-Tolerant (NFT) System: Due to the probabilistic dependencies in the fault/error/failure sequence, the reliability of an NFT system can be expressed as:

$$\begin{aligned} R &= 1 - P\{\text{Failure}\} \\ &= 1 - (P\{\text{Failure} \mid \text{fault}\} \cdot P\{\text{fault}\}) \\ &= 1 - (P\{\text{Failure} \mid \text{error}\} \cdot P\{\text{error} \mid \text{fault}\} \\ &\quad \cdot P\{\text{fault}\}). \end{aligned} \tag{1}$$

It has to be noted that for a strictly NFT system, $P\{\text{Failure} \mid \text{error}\}$ is equal to one and thus $P\{\text{Failure}\}$ is equal to $P\{\text{error} \mid \text{fault}\} \cdot P\{\text{fault}\}$. Thus $P\{\text{Failure}\}$ is essentially related to $P\{\text{fault}\}$ and the level of activity of the system which directly influences $P\{\text{error} \mid \text{fault}\}$. However, in practice, the analysis is even more complex due to the unintentional redundancies which cause $P\{\text{Failure} \mid \text{error}\}$ to be not quite equal to one: any system is to some extent naturally robust as we do not know how to build a strictly nonredundant system.

2) Reliability of a Fault-Tolerant (FT) System: As sketched by Fig. 1, the FT mechanisms can be viewed as a shield intended to prevent errors from leading to system failure. It is well known that the coverage of this shield has a prominent impact on the reliability measure [23]. In practice, depending on the types of FT mechanisms used, it may be useful to decompose the concept of coverage into several terms such as: the error detection coverage, the error recovery coverage, the fault isolation coverage [27], etc.

Relation (1) also applies of course in the case of an FT system and in particular, $P\{\text{Failure} \mid \text{fault}\}$ characterizes a lack of coverage of the shield, i.e., the failure of the fault tolerance (FT) mechanisms in properly handling the consequence of a fault. However, the values of the probabilities may be changed from the values observed in an NFT system:

- it is likely that $P\{\text{fault}\}$ will be higher due to the hardware and software overhead associated with the FT mechanisms;
- also $P\{\text{error} \mid \text{fault}\}$ might be increased as a result of the increase in the activity of the system (e.g., test programs, diagnosis, etc.);
- however, it is hoped that $P\{\text{Failure} \mid \text{error}\}$ will be significantly reduced by the application of the FT mechanisms (masking, detection, retry, etc.).

Although only probabilities of occurrence of specific events have been considered so far, another major issue is related to the time domain. In particular, two intervals identified on Fig. 1 require special attention [25]:

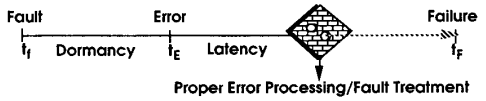


Fig. 1. Fault/error/failure sequence for an FT system.

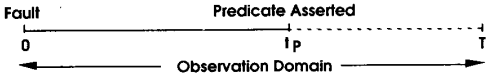


Fig. 2. Characterization of an experiment.

- the time interval between the occurrence of a fault and its activation as an error, i.e., the fault dormancy;
- the time interval between an error and its first perception by the FT mechanisms, i.e., the error detection latency.

All fault tolerance mechanisms are based on assumptions and are thus designed to deal with a specific set of fault/errors. Accordingly, their coverage and execution time parameters are strongly dependent on the error types they are faced with. As it is most likely that complex and widespread error patterns may result from the accumulation of faults, it is thus mandatory that the dormancy and the latency be decreased; this would limit the risk that the FT mechanisms be faced with errors they were not designed to deal with and thus fail to process them. As a corollary, for a specified observation time interval, a reduction in dormancy results in an increase in $P\{\text{error} \mid \text{fault}\}$ that has to be compensated by a significant reduction in the $P\{\text{Failure} \mid \text{error}\}$ probability.

Furthermore, the time interval between the occurrence of a fault and its complete recovery is of prime interest first, with respect to the interruption of proper service delivery [1] and also with respect to the problem of near-coincident faults [31].

In the remainder of this section, we show how the results of fault injection experiments can help in deriving useful statistical data to estimate the probabilities of (1) and the associated time intervals. In order to do so, we first provide in the next subsection a formal framework for experimental fault forecasting.

B. Characterization of Experimental Fault Forecasting

The readouts obtained during an experiment can be used to determine the observed state of the target system. The set of such observed states $\boldsymbol{S}$ can be defined in terms of combinations of a set of predicates $\boldsymbol{P}$ on the experiment readouts of the $\boldsymbol{R}$ set, e.g., $\{\text{fault\_activated}\}$, $\{\text{fault\_activated} \cap \text{parity\_error\_signaled}\}$, etc.

In order to simplify the presentation, we consider a single predicate P of the $\boldsymbol{P}$ set, e.g., for a self-checking system which must signal an error when a fault is present: $P = \{\text{error\_signaled}\}$, thus leading to two observable states $S_0 \Rightarrow P$ false and $S_1 \Rightarrow P$ true. The measures of interest in this case are the ratio of the number of observations of S_1 with respect to the total number of experiments and the distribution of the interval between the instant of fault injection and the observation of S_1, i.e., the instant at which P is asserted.

Fig. 2 illustrates this concept. $\boldsymbol{T} = [0, T]$ denotes the observation domain from the instant of fault injection; t_p designates the instant at which P is asserted, i.e., the fault is covered. Let T_p denote the associated random variable and $C(t)$ the cumulative distribution of T_P. $C(t)$ fully characterizes the coverage with respect to predicate P.

1) Statistical Characterization of a Test Sequence: In a first approximation, a test sequence can be considered as a series of Bernouilli trials. Let n denote the number of independent fault injection experiments and $N(t)$ the total number of assertions of P (coverages) observed in a time interval $[0, t]$. It follows that $N(t)$ is distributed according to the binomial distribution:

$$P\{N(t) = k; n\} = \binom{n}{k} [C(t)]^k [1 - C(t)]^{n-k}. \quad (2)$$

Accordingly, the mathematical expectation of the number of coverages is written as:

$$E[N(t)] = nC(t) \quad (3)$$

whence:

$$C(t) = \frac{E[N(t)]}{n}. \quad (4)$$

A statistical estimator of $C(t)$ is thus:

$$\hat{C}(t) = \frac{N(t)}{n}. \quad (5)$$

2) Practical Issues: Two practical issues of the specification of the fault injection test sequence have a significant impact on the derivation of statistical measures in the $\boldsymbol{M}$ set:

- $C(t)$ may be defective since all the faults cannot be properly covered, thus:

$$\max\{\text{coverage}\} = C(\infty) = \lim_{t \to \infty} C(t) \leq 1 \quad (6)$$

- the observation domain $\boldsymbol{T}$ is bounded by a time-out (T) and the readouts obtained from the experiments form a set of so-called *Type I* (or time) censored data [32].

Concerning the first issue, it is worth noting that—as in [33]—we are considering here the cumulative distribution function of the coverage instead of the density function that is used in other related studies (e.g., see [17]). In particular, the influence of the first issue is illustrated by the possible bias induced by the fact that the injected fault is not activated, i.e., the fault dormancy is infinite. Indeed, as previously defined, $[1 - C(\infty)]$ would incorporate the quantity $(1 - P\{\text{error} \mid \text{fault}\})$ from relation (1).[3] It is thus important that the experimentation envi-

[3]It has to be noted that such an incorporation would be totally acceptable to rate the efficiency of a test program that is part of the FT mechanisms, since the role of the test program is both to activate the fault and to make it observable.

ronment provide a means to discriminate the injected faults that were activated from those that were not activated. Such a possibility is indeed required to perform a direct measurement of the fault dormancy. An interesting approach to the problem of time measurement has been provided in [29] where an indirect measurement method of fault dormancy is proposed. However, the obtained estimation is restricted to the dormancy of the faults whose consequences were properly detected by the FT mechanisms; faults leading to system crashes could not be accounted for.

When combined, these two issues result in a total uncertainty for the $(n - N(T))$ experiments for which no coverage is observed. Indeed, two possible outcomes can be considered: i) the coverage time is delayed and would have been observed in the interval $]T, \infty[$; ii) no finite coverage time can be observed since $C(t)$ is defective. It follows that statistics may be obtained only from the actually observed data. Thus, the measurement of the coverage distribution is pessimistically biased, since only $C(T) < C(\infty)$ can be obtained. It follows that T has to be carefully determined according to the timing characteristics. The next subsection gives an example of use of the experimental readouts accounting for these practical issues.

C. Derivation and Use of Statistical Measures

Experimental readouts can be used to obtain statistical measures that may be useful to derive values to calibrate the coverage and time execution parameters incorporated in axiomatic dependability evaluation models [34], [33]. Such reference to a high level model is required to account for the fault occurrence process and thus to obtain dependability measures.

1) Coverage Modeling: As shown in [33], a rigorous approximation of the coverage requires that the effect of a second fault occurring during the processing/treatment of the first one be accounted for in the analysis; this leads to:

$$C' \approx C(1 - \lambda' E[T'_P]) \tag{7}$$

where C is the asymptotic coverage factor,[4] λ' is the second fault occurrence rate, and T'_P is the random variable characterizing the coverage time provided that it is finite. The distribution of T'_P can be easily related to the distribution of T_P, by considering that:

$$\begin{aligned} C(t) &= P\{T_P \le t\} = P\{P \text{ asserted in } [0, t]\} \\ &= P\{P \text{ asserted in } [0, t] \mid P \text{ asserted in } [0, \infty[\} \\ &\quad \cdot P\{P \text{ asserted in } [0, \infty[\} \\ &= P\{T_P \le t \mid P \text{ asserted in } [0, \infty[\} P\{T_P \le \infty\} \\ &= P\{T'_P \le t\} C(\infty). \end{aligned} \tag{8}$$

[4]Although the notion of "instantaneous" coverage has been used by other authors (see, e.g., [33]), we prefer the term "asymptotic" which better relates to the coverage distribution $C(t)$.

2) Estimation of the Parameters of the Coverage Model: This section shows how the results of an experimental fault-injection test sequence can be used for the estimation of the *coverage* and *execution time* parameters of FT mechanisms. Relation (7) shows that the two major measures of interest are the asymptotic coverage and the average coverage time. Another typical measure of interest that may be useful to complement these measures is the probability density function of the coverage.

a) Asymptotic Coverage: The asymptotic coverage is defined by $C(\infty) = P\{T_P \le \infty\}$. From (5), it follows that $C(\infty)$ is (conservatively) estimated by the statistical estimator:

$$\hat{C}(T) = \frac{N(T)}{n}. \tag{9}$$

b) Average Coverage Time: From (8) and in accordance with [33] the distribution function of T'_P is thus:

$$C'(t) = P\{T'_P \le t\} = \frac{C(t)}{C(\infty)}$$

and can be estimated for the observation domain $[0, T]$ by the statistical estimator:

$$\hat{C}'(t) = \frac{\hat{C}(t)}{\hat{C}(T)} = \frac{N(t)}{N(T)}. \tag{10}$$

The mathematical expectation of T'_P can be written as:

$$E[T'_P] = \int_0^\infty t \, dC'(t) = \frac{E[T_P]}{C(\infty)}$$

and can be estimated by:

$$\hat{E}[T'_P] = \frac{1}{N(T)} \sum_{i=1}^{N(T)} t_{Pi} = \frac{1}{\hat{C}(T)} \frac{1}{n} \sum_{i=1}^{N(T)} t_{Pi} \tag{11}$$

where t_{Pi} is the coverage time actually observed for experiment *#i*.

All the previous discussion implicitly addressed the estimation of the coverage parameters of FT mechanisms characterized by property P with respect to the presence of a fault. Let us use the subscript f to identify the parameters previously defined (e.g., P_f, $C_f(t)$, T_{Pf}, $\hat{C}_f(t)$, etc.). Analogous parameters can be derived for coverage related to the occurrence of an error by replacing C_f and $\hat{C}_f$ in the expressions above, by C_e and $\hat{C}_e$, respectively, where for example:

$$C_e(\infty) = P\{P_e \text{ asserted}\} = P\{\text{error covered} \mid \text{error}\}$$

is estimated by:

$$\hat{C}_e(T) = \frac{\hat{C}_f(T)}{\hat{E}_f(T)} \tag{12}$$

and $\hat{E}_f(T)$ is an estimate of $P\{\text{error} \mid \text{fault}\}$ from (1).

c) Probability Density Function: The probability density function of T_P is defined by: $c(t)\,dt = P\{t < T_P \le t + dt\}$, i.e., $c(t) = (d/dt)C(t)$. Let the observation domain $[0, T]$ be decomposed into k intervals of duration

$t(i) = t_i - t_{i-1}$, such that $T = \Sigma_{i=1}^{k} t(i)$; $c(t)$ can thus be estimated by a discrete probability mass function corresponding to a histogram defined by:

$$\hat{c}(t_i) = \frac{N(t_i) - N(t_{i-1})}{n}. \quad (13)$$

In particular, it can be easily verified that: $\hat{C}(T) = \Sigma_{i=1}^{k} \hat{c}(t_i)$.

D. Accuracy of Experimental Fault Forecasting

Besides the problems of i) the choice of the right measures and readouts (***M*** and ***R*** sets), and of ii) the length of each experiment (***T*** domain), which are inherent to the type of FT mechanism to be tested, two other more general attributes of the experimental environment significantly influence the accuracy of the experimental measures: the distribution of the input domain and the size of the test sequence.

1) Distribution of the Input Domain: Although the ***F*** set is of prominent importance in a fault injection test sequence, it should be noted that it is also necessary to provide a proper selection and variation of the ***A*** set since it determines in relation with ***F*** the creation of errors and thus the responses collected in the ***R*** set. As not all elements of the ***F*** and ***A*** sets can be fully tested in practice, it follows that adequate restrictions should be associated with the derived measures (e.g., see [35]: "$\cdots$ *x percent of type y errors will be detected for workload of type z*").

Furthermore, as an exhaustive coverage of the selected input domain is not always feasible, two distinct approaches can be considered to overcome this problem:

- *a priori* statistical sampling of the input domain according to the field profile,
- *a posteriori* weighting of clustered readouts and measures according to the field profile.

Although the second approach enables more flexible processing of the readouts of the experiments, the *a priori* sampling enables measures to be computed on-line which may useful to monitor the test sequence.

2) Size of the Test Sequence: The statistical model presented for the estimation of coverage enables some motivated judgement to be made on the accuracy of the experimental results obtained in a fault injection test sequence. Depending on the type of use of the model, two forms of actions can be carried out:

- determination of the confidence interval associated with an experimental measurement for a specified confidence level, e.g., see [36];
- acceptance or rejection test with a specified confidence level of a hypothesis concerning an *a priori* value to be matched by a coverage parameter, e.g., see [37].

IV. Implementation of the Methodology: The Messaline Experimental Fault Injection Tool

This section first presents the basic design guidelines concerning the implementation of the ***FARM*** attributes introduced in the previous sections to specify a fault injection test sequence. The current architecture and features of the validation tool obtained from these guidelines are then briefly described.

A. Design Guidelines

The main guidelines for the implementation of the fault injection methodology were to design a flexible experimental validation tool based on physical fault-injection capable of adapting easily to various target systems and to different measures. The physical level for fault injection was chosen in order to enable i) the injection of actual faults, and ii) the test of the implementation of FT mechanisms, which departs from and complements the recent simulation approaches presented in [6], [38], [7]. Furthermore, in opposition to some recent works on physical fault injection based on heavy-ion radiation [22], we emphasise the use of fault injection at pin level so as to be able to provide reproducible results.

One important issue concerns the validity of the faults injected: *are the injected faults representative of the actual faults?* In particular a major problem related to pin-level fault injection concerns the injection of faults which are representative of internal fauls of the IC's. The activation of internal faults will lead to errors that may affect more than one pin; thus, in order to provide a partial solution to this problem, pin-level fault injection must be able to inject temporary patterns on several pins.

The accurate simulation of the faulty behavior would require to perform an *a priori* simulation of the faulted IC to determine the right patterns to inject and—what is practically unrealistic—to establish a tight synchronization of the injection with the activity of the system to apply these patterns under timing conditions which accurately represent the error production process consecutive to fault activation. We think that a partial solution but doable alternative is to apply random patterns whose characterization (e.g., timing parameters, number of pins affected) can also be derived from such simulation studies.

Our approach was thus to have the ability i) of injecting multiple temporary faults, and ii) of investigating the application of different techniques for fault injection and the injection of non-stuck-at fault models at pin level.

So that the definition of a measure in ***M*** be flexible, the selection of the ***FAR*** parameters must be easily programmable. As an example, the definition of clusters of the form $< a_1, \boldsymbol{F}, \boldsymbol{R} >$, $< a_2, \boldsymbol{F}, \boldsymbol{R} >$, $\cdots$ may be useful to compare the influence of various activation modes (a_1, a_2, $\cdots$) on experiments characterized by the same ***F*** and ***R*** sets.

The activation modes for the target system are in fact closely related to the type of activation required for the FT mechanisms to be validated. In practice, such a determination depends heavily on the application domain of the target system: the activation may be ensured by a specific test program or through a specific environment simulator that exercizes/samples the inputs/outputs of the target system. The main concern with respect to the ***A*** set is

restricted to the provision of adequate communication means to initialize the provided activation modules. It is worth noting that such a task would be significantly facilitated by giving special attention to controllability issues during the design of the target system.

We describe next in more detail the design guidelines concerning the ***F*** and ***R*** sets.

*1) The **F** Set:* A pin on which an injection element is connected will be called an *injection point*. For each experiment, the injected fault f actually corresponds to a set of elementary faults $\{f_1, f_2, \cdots, f_m\}$, where m denotes the multiplicity of the fault. Every elementary fault f_i at an injection point is characterized by the following attributes:

- the location of the fault, that designates one failed element in the target system (e.g., IC location and pin number),
- the nature (model) of the fault,
- the time of application of the fault with respect to the beginning of the experiment,
- the duration of application of the fault.

Concerning the nature of the fault, it is necessary to distinguish i) the logical or electrical characteristics (stuck-at-0, stuck-at-1, short, open, etc.) that define the type of the fault, and ii) the temporal characteristics of the fault (permanent, transient, or intermittent fault). In order to fully characterize a nonpermanent fault, it is mandatory to specify the average period of occurrence and/or duration as well as the distributions.

*2) The **R** Set:*

For each experiment, the readout r corresponds to a set of elementary readouts $\{r_1, r_2, \cdots, r_n\}$ that correspond to specific observations of the reactions of the target system to the fault injected. The basic types of r_i that can be made on the target system can be decomposed into three types:

- binary status readouts,
- counter readouts,
- extensive data acquisition readouts.

A binary readout consists of a predicate characterizing the yes or no response to the occurrence of one event. Counter readouts correspond essentially to time measurements that complement the binary readouts by indicating the time of occurrence of the monitored event during an experiment; they may also correspond to measurements of the interval between two events or conversely, the number of events in a given interval. The third type of readouts allows retrieval of extensive data on the evolution of target system behavior in presence of the fault; this facility is essential for *post mortem* analysis of system behavior.

B. The MESSALINE Architecture [39], [2]

The current architecture of MESSALINE and the basic testbed configuration when connected to the target system (a physical hardware/software prototype) are depicted in Fig. 3.

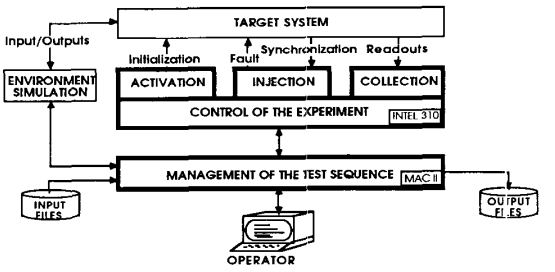


Fig. 3. General architecture of MESSALINE.

MESSALINE is made up of four modules:

- a fault *injection* module to enable one element of the ***F*** set to be injected,
- a target system *activation* module to ensure the proper initialization of the target system according to the elements of the ***A*** set,
- a readout *collection* module to collect the elements of the ***R*** set,
- a test sequence *management* module to elaborate the ***M*** set by means of a proper management of the ***FAR*** sets.

The first three modules are hardware modules implemented on an Intel 310 microcomputer which constitutes the backbone of the architecture that controls each experiment. The software management module that was initially implemented on the 310 system has been recently transferred to a MacIntosh II system to enable more operator interface flexibility (programming environment, result analysis, etc.). The main features of these modules are detailed in the following subsections.

1) The Injection Module: In the current configuration, the injection module enables injection on up to 32 injection points by means of injection elements supporting two different fault injection techniques:

- *forcing*, where the fault is directly applied by means of multipin probes on IC(s) pin(s) and associated equipotential line(s),
- *insertion*, for which, one or more IC(s) is(are) removed from the target system and inserted on a specific box where solid-state switches ensure the proper isolation of the IC(s) from the system.

The table of Fig. 4 lists the faults that can be injected. Logical bridging corresponds to logical AND or OR functions of two adjacent pins, while physical bridging corresponds to a direct short between two adjacent pins.

It is worth noting that for each of these faults, the timing characteristics (duration and frequency of occurrence) are user-programmable and it is possible to control the injection elements by signals originating from the target system.

A specific device is associated with each injection element that enables the occurrence of an error (activation of a fault) to be identified at the point of injection (forcing: current sensing, insertion: comparison of signals on both sides of the switch). This novel capability is used to resolve whether injected faults are actually activated and to perform a direct measurement of the fault dormancy.

	Forcing	Insertion
Stuck-at-0	•	•
Stuck-at-1	•	•
Stuck-at-External Value	•	•
Logical Bridging (previous pin)		•
Logical Bridging (next pin)		•
Physical Bridging	•	
Intermediate Voltage Level	•	
Inversion (effective error)		•
Open		•

Fig. 4. Nature of injectable faults.

In the current implementation (based on the use of 2N170-type bipolar transistors) the maximum frequency of the signal to be faulted is 10 Mhz. For signals of higher frequency parasitic mutations induced by capacitive effects impede proper control of the injection; for example, the parasitic capacitance of the insertion elements prevents from injecting open-type faults. Speed-related restrictions also apply to the synchronization features. The fault activation monitoring devices at each injection point provide proper indication for error signals longer than 300 ns for insertion and 1 μs for forcing.

It is worth noting that the implemented injection module thus encompasses the features of other related efforts (e.g., [17], [20], [18]).

2) The Activation and Collection Modules: The activation and collection modules are based on standard interface boards supporting parallel input/output lines and serial lines. In the current configuration, 24 parallel outputs are used to control the target system and the environment simulator (essentially in the initialization phase of the experiment), 48 parallel inputs are devoted to the collection of readouts (error signals, outputs of the system, etc.). Two RSC232 serial lines may be used for communication with the environment simulator, for activation of the target system and for readout collection.

A dedicated board was developed to increase the collection features of MESSALINE to support event counting (eight edge-triggered interruption lines with a maximum of 65 536 events per signal) and time measurement (eight level-sensitive inputs controlled by a 16-bit counter with a resolution varying from 1 μs to 65 ms). In order to facilitate the adaptation to various target systems, the active edge of each interrupt line and the active level of each time measurement input are switchable.

3) The Management Module: This module is responsible for the automatic and parametrable generation of a test sequence, for the run time control of its execution and for result archiving for post-test analysis. The management is based on three specific description files:

- the *board description* file, that details the interconnection of the integrated circuits (IC) on a board;
- the *IC library* file, which defines the size (number of pins) of an IC as well as the type of each pin (input, output, ground, etc.);
- the *injection device description* file, that contains the description of the relationship between each injection element and each pin of the injection devices, e.g., pin i of device j is connected to the injection element k that uses the forcing technique. Several injection devices can be used during a single experiment which enables simultaneous injection on several IC's.

The fault injection test sequence is generated by means of command files in which the ***FARM*** sets are described. Such a description and the files previously described are used to generate an exhaustive or random selection of the elements of the $\boldsymbol{F}$ set. A more detailed description of the form of the command files can be found in [2].

V. Examples of Experimental Results

A. Characterization of the Experiments

Two different types of target systems have been tested with MESSALINE:

- a centralized system, consisting of a prototype of a computerized interlocking system (called PAI) for railway control applications designed by SNCF (French National Railways) [40], [34];
- a distributed system, corresponding to the current implementation of the dependable communication system of the ESPRIT Delta-4 Project, called the Multicast Communication System (MCS) [41].

The complete results are presented in [42] and [43], respectively.

These studies are interesting since they provide complementary application domains for the tool MESSALINE as shown in Fig. 5.

In both studies, the type of target system and the validation goal had a significant impact on the choice of the test sequence attributes. In particular, the type of architecture had a major impact on the difficulty to set up a reliable testbed and on the repeatability of the experiments.

In the case of the PAI study, the fact that the FT mechanism to be validated was a self-test program based on the stuck-at fault model led to the injection of permanent faults. In order to comply with the design assumptions and to enable an exhaustive test, only single pin faults were injected and to prevent damage to the IC's, the insertion technique was applied. The choice of the $\boldsymbol{A}$ set was implicit. We focused on the analysis of the diagnostic message delivered by the test program and on the available hardware error detection signals. As the target system is activated by a test program and not by the execution of the application software, only event occurrence and message collection readouts were performed since, in this case, timing measurements would not be meaningful to the operational behavior; thus, only asymptotic coverage measures have been derived (see (9) in Section III). The information provided by the fault activation monitoring device was used to partition the deficiencies of the test program between the activation and observation aspects. The time-out value characterized by the $\boldsymbol{T}$ domain is mainly a consequence of the duration of the test program. The major difference between the time-out value and the duration of an experiment is due to the fact that each test has been carried out twice: first with fault injection and then without fault injection to check the adequate recov-

	Railway PAI	Delta-4 MCS
The Target System:		
Architecture	Centralized	Distributed
F° Mechanisms Tested	Self-Test Program	Self-Checking Hardw. & Protocol Resiliency
The Main Validation Goal	Verification	Evaluation and Verification
the *F* Set:		
Injected Faults	Permanent Single	Intermittent Multiple
Selection of the Faults	Exhaustive	Random Sampling
Injection Technique	Insertion	Forcing
The *A* Set	Test Program	Several Workloads
The *R* Set:	Software Diagnosis Message (168 bytes)	Hardware Detection Signals (occurr. & timing)
	Hardware Detection Signals (occurrences)	Observed Traffic (≈ 100 messages)
Fault Monitoring Device	Activ./Observ. of Test Program	Error Occurrence & Fault Dormancy
The *M* Set	Asymptotic Coverage	Coverage Distribution
The *T* Domain	3 s	110 s
Experiment Duration	10 s (max)	5 mn (average)

Fig. 5. Major characteristics of the reported fault injection test sequences.

ery of the system from the injected fault; such a procedure identified a problem for only 5 percent of the experiments which enabled the corresponding results to be discarded from the analysis.

In the case of the Delta-4 test sequence, the target system was distributed and it was thus not possible in practice to carry out repeatable experiments and we had to rely on statistical evaluation rather than on deterministic verification. In order to tackle with the most frequent fault types, we injected intermittent faults on multiple pins of a single IC. In order to cope with the large number of parameters characterizing the ***F*** set, we used random sampling for the determination of the faults to be injected. The temporary nature of the faults injected allowed use of the forcing technique. The workload was varied to study its impact on the faulty behavior of the target system. Hardware detection signals as well as messages exchanged in the observed traffic constituted the ***R*** set and were respectively analyzed to rate the efficiency of the Delta-4 self-checking communication hardware and the resilience of the atomic multicast protocol executed on this hardware. The information provided by the fault activation monitoring device was used to eliminate from the statistics those experiments where the fault was not activated and also to measure dormancy. The collection of occurrence and timing readouts enabled the derivation of empirical coverage distributions (see (5) and (13) in Section III) for the hardware detection signals. The much larger time-out value is imposed by the length of the activation modes considered. The significant extension of the duration of an experiment is due to the experiment setup and by the possible execution of the automatic recovery/restart procedures in case of failure of the testbed.

The next subsections further detail the experiments with a special emphasis on the results obtained. Although most of the results are specific to the systems tested and cannot always be generalized to other systems, they nevertheless exemplify the type of analysis that can be carried out with MESSALINE.

B. PAI Validation

1) The Target System: The PAI architecture is characterized by an active-redundancy duplex structure made up of two self-testing 68 000-based processing units (CPU's) and monitored by a two-out-of-two voter implemented in fail-safe hybrid technology.

The target system of the experiments reported here is a single CPU running a sample self-test program: a complete testbed environment (including the duplex structure, the application program and the environment simulation module) was not available. In particular, the lack of the application program did not allow proper use of all the hardware redundancies provided to ensure error detection. The main goal of the experiments was thus to test the self-test program that had been designed on the permanent stuck-at fault hypothesis.

2) Definition of the Experiments: The main characteristics of the experiments are summarized in the table of Fig. 5; we only focus here on the features that fully describe the attributes of the experiments.

*a) The **F** Set:* The test sequence is defined by the injection of all single pin permanent faults characterized by the following set:

- stuck-at-0, stuck-at-1 and open circuit faults on functional (input) pins;
- open circuit faults only, on the power supply pins.

*b) The **A** Set:* The activation corresponds to the execution of a test program made up of several modules each tailored to test the functions of the CPU (input, output, processor, RAM, PROM, etc.).

*c) The **R** Set:* Two types of readouts have been collected for each experiment:

- *binary* readouts corresponding to:
  - —information concerning activation of the injected fault,
  - —hardware error detection signals that constitute the hardware diagnosis,
- *messages* delivered by the procedures integrated into the test program that specify the software diagnosis.

Four error detection signals ensure the hardware diagnosis: DA (double addressing), DTACK (lack of acknowledge from memory and peripheral circuits), RAM (RAM addressing check), PROM (PROM addressing check).

For the analysis, the messages delivered by the software diagnosis are decomposed into four classes: i) *complete diagnosis*: the message properly identifies the test module that has detected the error, ii) *muteness*: no message is delivered, iii) *garbage*: the message consists of a sequence of random symbols, and iv) *no detection*: the message is identical to the nominal message obtained in

the absence of faults. The first three classes correspond to cases where the message differs from the nominal message and we thus consider that these classes contribute to the software diagnosis, although the message is not always sufficient to identify the fault.

d) The M Set: Although the main concern of the experiments was directed to qualitative analysis, the study also enabled estimates of the asymptotic coverage to be obtained for the hardware and software diagnoses.

3) Result Analysis: The fault injection test sequence consisted of more than 6000 experiments on 144 IC's ranging from 14 to 28 pins. The study generated a 600 kbyte database storing, for each experiment:

- the location tested on the board, the IC at this location, the number of the injected pin, the type of the fault;
- the indication of error occurrence (fault activation) at the injected pin, the readouts of the error detection signals, the message delivered by the software diagnosis.

The test sequence enabled some abnormal behaviors to be identified. The complete database has been delivered to SNCF for a detailed analysis to help improve the design of the system and the diagnosis procedures.

Furthermore, a simple statistical treatment of the data base was carried out, which provided a basis to perform a more refined analysis of the results.

a) Detailed Results: Fig. 6 presents an example of the results obtained for all the memory chips of the CPU. The results show a large difference among the tested IC's and that the IC's can be grouped in pairs. This results from the fact that the data path is 16 bits wide and thus the IC's of each pair correspond to the same address space. In particular, the differences in the detection and muteness columns for the software diagnosis are due to the distinct use of the IC pair by the test program (e.g., for the PROM IC-pairs: ZL6 & ZN6 are simply tested, while ZJ7 & ZJ8 store the test program).

Fig. 6 enables the efficiencies of the hardware diagnosis error detection mechanisms to be compared and gives the distribution among the various classes of the software diagnosis.

For the hardware diagnosis, it is worth noting that more than one signal can be activated during a single experiment and thus the percentages of the Error Detection Signal columns do not sum up to the values of the Global Coverage column. The DTACK and RAM signals provide most of the diagnosis.

Although a software diagnosis was available in more than 90 percent of the experiments, a complete diagnosis was obtained in less than one third of the cases.

Fig. 7 shows the statistical results obtained for the complete CPU. The percentages in **bold** characters correspond to the nominal coverages. The figures in *italics* correspond to the upper and lower confidence limits for a 95 percent confidence level [36]. The coverages for the complete CPU are significantly less than the coverages for the memory circuits, but the general tendency for the distribution of the coverages among the detection signals and the classes of software diagnosis is almost the same. These

Location	IC	Number of Experiments	Hardware Diagnosis: Global Coverage	Error Detection Signals: DA	DTACK	RAM	PROM	Software Diagnosis: Total	Form of the Message: Complete	Muteness	Garbage
Random Access Memory											
ZE7	TC5565	77	23.38%	0.00%	23.38%	20.78%	0.00%	98.70%	74.03%	23.38%	1.30%
ZE8	TC5565	77	24.68%	0.00%	24.68%	20.78%	0.00%	97.40%	72.73%	23.38%	1.30%
ZF7	TC5565	77	58.44%	0.00%	58.44%	57.14%	0.00%	94.81%	28.57%	55.84%	10.39%
ZF8	TC5565	77	59.74%	0.00%	58.44%	54.55%	3.90%	96.10%	22.08%	61.04%	12.99%
Read Only Memory											
ZL6	2716	66	42.42%	0.00%	42.42%	42.42%	0.00%	68.18%	19.70%	45.45%	3.03%
ZN6	2716	66	46.97%	0.00%	43.94%	42.42%	7.58%	78.79%	28.79%	43.94%	6.06%
ZJ7	27128	76	92.11%	0.00%	90.79%	85.53%	5.26%	97.37%	5.26%	86.84%	5.26%
ZJ8	27128	76	92.11%	0.00%	92.11%	82.89%	1.32%	97.37%	3.95%	90.79%	2.63%
Global Statistics		592	55.24%	0.00%	54.56%	51.01%	2.20%	91.72%	32.26%	54.05%	5.41%

Fig. 6. Detail of the results obtained for the memory circuits.

Complete CPU Coverage Statistics									
5760 Exper.	Hardware Diagnosis: Globally	DA	DTACK	RAM	PROM	Total	Software Diagnosis: Complete	Muteness	Garbage
Upper CL	*21.99%*	*0.18%*	*20.31%*	*13.78%*	*2.06%*	*52.38%*	*30.89%*	*20.70%*	*1.83%*
Nominal	**21.11%**	**0.10%**	**19.46%**	**13.06%**	**1.77%**	**51.32%**	**29.91%**	**19.84%**	**1.56%**
Lower CL	*20.21%*	*0.03%*	*18.59%*	*12.32%*	*1.48%*	*50.21%*	*28.90%*	*18.97%*	*1.29%*

Fig. 7. Statistical results for the CPU.

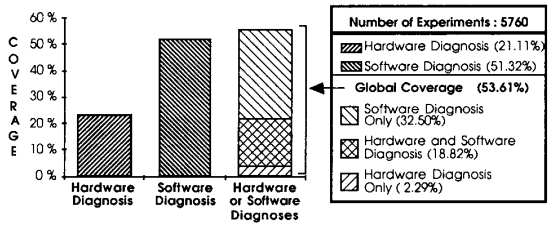


Fig. 8. Distribution among the hardware and software diagnosis.

results suggest that, in regard to the value of the coverage, the contribution of the DA and PROM mechanisms is minor.

An analysis of the influence of the nature (stuck-at-0, stuck-at-1, open) of the faults was also performed which did not reveal any significant impact on the hardware and software diagnoses.

b) Comparison of Hardware and Software Diagnoses: Fig. 8 focuses on the respective distribution among the hardware diagnosis and the software diagnosis. The software diagnosis is much more efficient than the hardware diagnosis and the contribution of the hardware diagnosis alone to the global coverage is very limited (2.29 percent). Also, the global coverage level should be improved; indeed, such a level is not sufficient i) to neglect the risk of accumulation of latent faults in the paired CPU's that would defeat the duplex structure, or even more, ii) to allow a degradation in simplex mode. Furthermore, the coverage values obtained should be regarded as upper bounds of the actual ones since only permanent faults were injected.

c) Analysis of the Lack of Diagnosis: Fig. 9 identifies the causes for the 46.39 percent portion of lack of diagnosis by using the information delivered by the fault activation monitoring devices.

Indeed, as a result of the restriction of the fault injection to the functional pins actually used, the fact that an injected fault remains inactivated characterizes an activation deficiency (10.90 percent) of the test program supporting the software diagnosis.

On the other hand, the lack of diagnosis when the injected fault is effectively activated and produces an error,

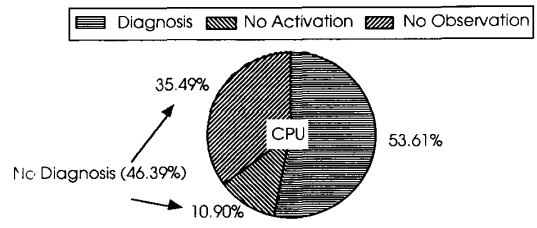


Fig. 9. Identification of the causes for lack of diagnosis.

corresponds to an observation deficiency (35.49 percent) of the software diagnosis mechanisms integrated into the test program.

These results show that the coverage problem is both an activation and an observation problem. It is worth noting that the lack of diagnosis, when the injected fault is actually activated, characterizes also a coverage deficiency of the error detection mechanisms of the hardware diagnosis.

C. Delta-4 MCS Validation

1) The Target System: The Delta-4 multicast communication system (MCS) provides generalized multicast services that are built on top of an atomic multicast protocol (AMP). In the tested version, the AMP is implemented as an extension of the medium access control (MAC) sublayer for token-ring Local Area Networks based on the 8802.5 standard.

MCS is implemented in communication servers, called the Network Attachment Controllers (NAC's), that connect the stations to the Delta-4 network. The NAC's are assumed to exhibit the fail-silent property (a kind of halt-on-failure behavior enforced through self-checking hardware).

The fault injection experiments were carried out in order to cope with the validation of two levels of coverage:

- *local coverage* of the concurrent testing capabilities of the NAC's,
- *distributed coverage* provided by the defensive properties of the AMP protocol entities with respect to injected faults.

Upon detection of an error, the self-checking mechanisms of the NAC control a relay that provokes its extraction from the ring. The NAC extraction mechanisms and the AMP properties are the basic building blocks that support the fault tolerance features of the Delta-4 architecture [41]. The definitions of the set of tested properties of the AMP are given below:

PA. Unanimity	Any message delivered to a correct recipient is delivered to all other correct recipients.
PB. Nontriviality	Every message positively confirmed to a correct sender, is delivered to all correct recipients.
	Every message delivered to a correct recipient is positively confirmed to its sender, if the sender is correct.
PC. Order	If a pair of messages are delivered to a pair of correct recipients, then they are delivered in the same order to both recipients.

2) Definition of the Experiments: The testbed configuration is shown in Fig. 10. For the experiments, the target system was composed of four stations each implemented by a Bull SPS-7 computer (target system host) interconnected through a token ring NAC component to the physical medium (target network). Faults were injected into a single NAC; the four stations were thus partitioned into two sets: the injected station (station S1) and the three "correct" (noninjected) stations (stations S2, S3, and S4).

The experiment supervisor provides the run time control of the experiment in each station and the analysis of the data collected by each one, in order to verify the protocol properties; it is implemented by a BULL SPS9 computer. The testbed network (an Ethernet) ensures the communication between the experiment supervisor and the target system hosts.

The test sequence manager offers the operator interface necessary to start the experiments. It is responsible for the run-time control of the test sequence and for archiving data for further analysis. The test sequence manager is connected to the experiment supervisor and the experiment controller through serial lines. Early experiments with the testbed revealed the possibility of crashes of all the stations and resulted in the implementation of specific hardware control lines between the experiment controller and all the stations so as to be able to reset them.

The fault injection test sequence is characterized by the following attributes.

a) The F Set: For each IC, the injected pins, the nature and the timing characteristics of the injected faults were selected randomly according to the order and form shown below:

1°) Multiplicity (MX): Faults are injected on several (1, 2, or 3) pins of an IC, with a frequency of 50, 30, and 20 percent, respectively.

2°) Location: Selection of MX pins among the injectable pins, with a uniform distribution.

3°) Nature: Stuck-at-0 and stuck-at-1 faults, each with equal probability of occurrence.

4°) Timing Characteristics: As the faults injected are essentially intermittent faults, their temporal parameters are composed of three values:

- —*lead time* (from start of experiment to first pulse): uniformly distributed in the interval: 1 to 40 seconds;
- —*width:* uniformly distributed between 2 μs and 1 ms;
- —*period:* logarithmically distributed between 10 μs and 30 ms, with a duty cycle ≤ 50 percent.

In the lack of sound available data, most of the parameters were selected according to a uniform distribution

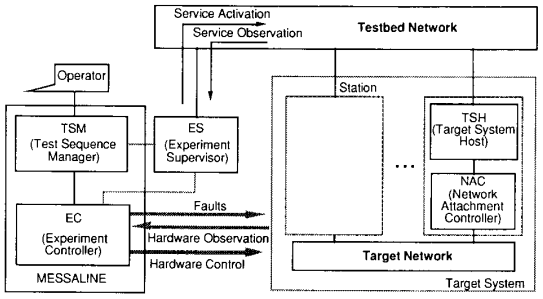


Fig. 10. Testbed configuration.

among range values selected quite arbitrarily. However, the limitation to a multiplicity order of 3 is to some extent substantiated by the results presented in [22] for a microprocessor. These results show that more than 80 percent of the internal single-event upsets led to error patterns on up to 3 pins. The logarithmic distribution for the period is intended to obtain a significant number of short period intermittent faults while maintaining a wide selection range.

*b) The **A** Set:* The activation is characterized by the application of 2 types of traffic flows: *observed traffic* flow with respect to which AMP properties are tested, and *background traffic* flow to provide more realistic activation of the target system.

In order to provide a representative activation set, five activation modes have been considered for the stations; Fig. 11 shows the Transmitter/Receiver allocations of the stations for each mode with respect to observed traffic and background traffic flows.

*c) The **R** Set:* Three types of readouts are collected for each experiment:

- *binary* readouts: activation of the injected faults, status of the ring insertion relays for each NAC (*Ri* is assumed to be true if the NAC of station *i* is inserted into the ring), AMP properties derived from the analysis of the messages;
- *timing* readouts: activation of the injected fault, extraction of the stations;
- *message* readouts: number of messages exchanged for both traffic flows and number of messages positively or negatively confirmed for the observed traffic.

*d) The **M** Set:* The measures considered for the analysis presented here consists of two types of measures: *predicates* and *time distributions*.

In order to analyze the results of the fault injection experiments, with respect to the local and distributed coverages we introduce the two following predicates:

- the *local coverage predicate X* that characterizes the NAC self-checking mechanisms:

$$X = E \cdot \overline{R1}$$

where E indicates that the injected fault is activated as an error; X is true if the NAC of the injected station is extracted when a fault is activated (the expected behavior in presence of faults).

	Observed traffic				Background traffic			
Mode	S1	S2	S3	S4	S1	S2	S3	S4
1	TR	TR	R	R	TR	TR	TR	TR
2	TR	TR	TR	R	TR	TR	TR	TR
3	-	TR	TR	R	TR	TR	TR	TR
4	-	TR	TR	R	-	TR	TR	TR
5	X	TR	TR	R	X	TR	TR	TR

T : Transmitter present R : Receiver present
X : Extracted

Fig. 11. Transmitter/receiver allocations per mode.

- the *distributed coverage predicate K* that characterizes the defensive properties of the protocol at the MAC layer:

$$K = P \cdot C = (PA \cdot PB \cdot PC) \cdot (R2 \cdot R3 \cdot R4)$$

i.e., K is true whenever all the AMP properties (predicate P) are verified and the confinement of the fault/error (predicate C) is ensured (all the "correct" stations remain inserted in the ring after an experiment).

In order to complement the analysis, two types of time distributions were also accounted for:

- the *fault dormancy:* the time interval between the injection of a *fault* and its activation as an *error* at the point(s) of injection; if T_D denotes this random variable, and T_E and T_F the error and fault times respectively, then:

$$T_D = T_E - T_F.$$

- the *error latency:* the time interval between an error occurrence and the extraction of the NAC of the injected station; if T_L denotes this random variable, and T_X the extraction time, then:

$$T_L = T_X - T_E.$$

3) Result Analysis: Although only 20 IC's out of the 101 that compose the NAC were actually injected, the use of the forcing technique allowed 60 percent of the pins of the NAC IC's to be subjected to injected faults. The results concern a total of 3000 fault injection experiments (20 IC's * 5 modes * 30 experiments per mode). The test sequence for a single IC was fully automated which enabled the experiments to be carried out overnight.

The experiments resulted in a primary database of more than 2000 kbytes. Detailed analysis of the influence of the tested IC location and of the influence of the activation modes was carried out and reported in [43]. In this section, we focus essentially on the presentation of summarized results concerning the predicate combinations analysis, the dormancy and the latency measures. As the fifth activation mode is specific (faults are injected into an active but noninserted NAC), the reported statistics concern the first four modes only, i.e., 2400 experiments.

a) Predicate Combinations Analysis: Fig. 12 characterizes the predicate combinations used for the analysis.

Fig. 13 shows the statistics obtained for the possible combinations of the K and X predicates. The table gives the number of occurrences and frequency of each predicate combination.

$K \cdot X$:	This case corresponds to the ideal case when the error results in: - the permanence of the connection of the non-faulted stations and the verification of the properties of the protocol, - the extraction of the injected station.
$K \cdot \overline{X}$:	In this case the injected fault does not alter the observed behavior of the system (the protocol properties are verified and the non injected stations remain connected); this case may correspond to: - an activated fault creating an error at the point of injected but not propagated , - a propagated error which remains latent inside the NAC, - an error propagated outside the NAC, but tolerated by the defensive mechanisms of the protocol entities.
$\overline{K} \cdot X$:	This case corresponds to a failure of the AMP that may result from: - either, an implementation deficiency in the protocol to handle the extraction of a station, - or, a much too high extraction latency of the injected station resulting in the alteration of the AMP properties
$\overline{K} \cdot \overline{X}$:	This case characterizes a major failure of both the extraction mechanisms and the AMP properties.

Fig. 12. Characterization of the predicate combinations.

#Experiments: 2400	Errors	$K \cdot X$	$K \cdot \overline{X}$	$\overline{K} \cdot X$	$\overline{K} \cdot \overline{X}$
Number of Occurrences	2198	1483	523	188	4
95% Upper CL	*92.42%*	*69.03%*	*25.26%*	*9.53%*	*0.34%*
Nominal Frequency	91.58%	67.47%	23.79%	8.55%	0.18%
95% Lower CL	*90.54%*	*65.74%*	*22.27%*	*7.56%*	*0.02%*

Fig. 13. Predicate combinations.

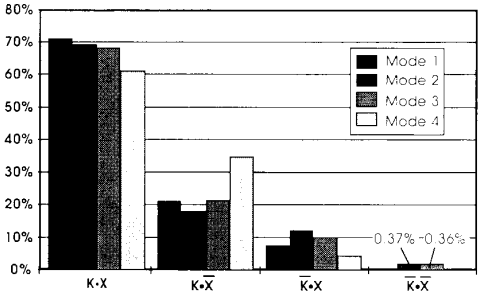


Fig. 14. Distribution among predicate combinations.

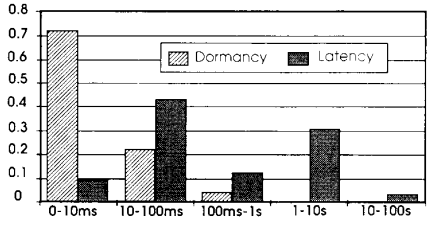


Fig. 15. Dormancy and latency distributions.

For the number of occurrences, the first column (labeled Errors) gives the number of injected faults that were actually activated. The four subsequent columns indicate the number of cases when the predicate combinations were observed provided an error was injected.

It is important to note that the frequencies for the Errors column are computed with respect to the number of experiments (2400): more than 90 percent of the faults are activated; this figure corresponds to the statistical estimator $\hat{E}_f(T)$ (see (12) in Section III). For the predicate combinations, the nominal frequency corresponds to the ratio of the number of occurrences of the combination over the number of significant experiments, i.e., the number of errors (2198). These values are estimates of the $\hat{C}_e(T)$ type [see (12)]. The top and bottom figures in italics indicate in each case the upper and lower confidence limits for a 95 percent confidence level.

These results show that an apparent distributed coverage of 91.26 percent ($K \cdot X$ and $K \cdot \overline{X}$) is achieved. A large proportion of it ($\approx 1/4$) is due to the $K \cdot \overline{X}$ combination explained in Fig. 13, the relative contribution of the local coverage being limited to about $3/4$. On the other hand, from the 76.02 percent figure of the local coverage ($K \cdot X$ and $\overline{K} \cdot X$) about 11 percent still result in the failure of the AMP properties (case $\overline{K} \cdot X$). Further analysis of this combination will be presented later.

Fig. 14 visualizes the impact of the activation modes on the distribution among the predicate combinations. This shows that the reduction in the activation of the injected station (in particular in mode 4) significantly reduces the local coverage efficiency (predicate X).

b) Timing Analysis: Fig. 15 plots the histograms characterizing the dormancy and latency empirical probability density functions. Although the time-out was set to 110 s, all readouts fell below 100 s; accordingly, we decomposed the time scale into five logarithmic time intervals. The dormancy plot shows that about 95 percent of the faults are activated in less than 100 ms and that if a fault is not activated within 1 s there is a high probability that it will never be activated. According to latency, the histogram depicts a bimodal distribution that identifies (at least) two distinct behaviors with respect to the injected faults; such an observation is of significant interest with respect to modeling when selecting the number and types of coverage parameters.

c) Analysis of the Influence of Station Extraction on Distributed Coverage: This analysis is intended to discriminate among the two possible causes—AMP Design/Implementation Problem versus Large Extraction Latency—for the occurrence of the $\overline{K} \cdot X$ predicate combination (Fig. 12). Specific experiments were carried out to characterize AMP behavior upon extraction of one station. It should be noted that, due to the specificity of these experiments, in all cases the predicate X is true (each experiment is in reality a forced station extraction). Fig. 16 compares the results obtained with those presented in the table of Fig. 13 for the fault injection experiments.

It appears that for the 1600 specific experiments carried out, one or more of the "correct" (noninjected) stations were extracted in 21 cases (1.31 percent). This figure clearly identifies a potential design/implementation problem. However, the 10 percent difference with respect to the fault injection experiments shows that this is not the only cause for the observation of the $\overline{K} \cdot X$ predicate combination. Fig. 17 plots the relative proportion of the percentages of the two predicate combinations where the predicate X is true as a function of the extraction time.

These results show that the percentage of $K \cdot X$-type extractions is about 3 times greater than the $\overline{K} \cdot X$-type for extractions times in the range of 10–100 ms and that

	# Extrac. (X)	$\overline{K}$·X (Occ.)	$\overline{K}$·X (Freq.)
Specific Experiments	1600	21	1.31 %
Fault Injection Exp.	1671	188	11.25 %

Fig. 16. Comparison of specific and fault injection experiments.

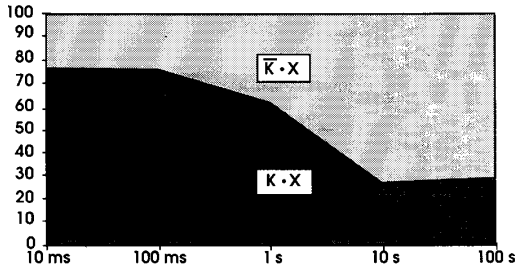


Fig. 17. Proportion of $\overline{K} \cdot X$ and $K \cdot X$ combinations with respect to extraction time.

this proportion is reversed for the 10–100 s interval. This clearly reveals the influence of the time efficiency of the local coverage on the distributed coverage and substantiates a well known general result: the later the errors are caught by the detection mechanisms the more difficult it is to prevent the occurrence of a failure.

VI. Conclusion

This paper proposed a novel framework for the characterization of the major attributes of fault injection in a validation-directed design process (fault, activation, readout, and measure sets). In particular, the implications and interactions concerning the abstraction level of the target system and the goal of the fault injection have been presented in detail.

The problems related to the derivation of significant experimental measures (time censoring of the experiments, defective time distribution, validity of the injected faults, size of the test sequence) have also been addressed.

The considerations above show that it is readily possible to implement a methodology for experimental validation by physical fault injection provided that a general tool is available. Furthermore, the features of the injection tool (MESSALINE) that has been developed at LAAS and the experimental results presented show that pin-level fault injection constitutes a significant contribution to increase the confidence in the validation process.

From a practical point of view, despite the limitations of the methodology (late application in the design process, incompatibility with stringent dependability objectives) and the difficulty expressed for its application (selection of the faults to be injected, synchronization of the injection, parasitic mutations induced by the physical interference between the target system and the experimental tool, effort to set up a reliable testbed), the two examples presented definitely confirm the ability of the methodology i) to point out design/implementation deficiencies after the systems had been designed and validated, and ii) to obtain measures to help characterize the behavior of the systems in the presence of faults (e.g., impacts of the workload, of the latency).

Since physical fault injection will always be needed to test the actual implementation of a fault-tolerant system, much effort is still needed to improve the application of the methodology. Besides the problems already mentioned, we would like to mention the difficulty of comparing the results obtained in different experimental studies. Strong interrelationships exist between axiomatic modeling, fault simulation, and physical fault injection (e.g., estimation of analytical parameters, determination of pin level error patterns by internal fault simulation); they should be further investigated as an attempt to minimize the problems encountered in the application of physical fault injection.

Further work on fault-injection is currently being carried out at LAAS in two directions:

- the continuation of the validation of the Delta-4 network, in particular with the use of network attachment controllers with improved self-checking;
- the investigation of the application of fault-injection in the early phase of the design process and during the development process in order to complement the physical fault injection approach implemented by MESSALINE; "rapid prototyping" and simulation approaches appear to be promising directions.

Acknowledgment

The authors would like to thank P. Traverse (now with Aérospatiale-Toulouse) for his valuable contribution to the early design of the MESSALINE tool. Useful comments from A. Costes on previous reports on this work are also gratefully acknowledged.

References

[1] J.-C. Laprie, "Dependable computing and fault-tolerance: Concepts and terminology," in *Proc. FTCS-15*, Ann Arbor, MI, June 1985, pp. 2–11.

[2] J. Arlat, Y. Crouzet, and J.-C. Laprie, "Fault-injection for dependability validation of fault-tolerant computing systems," in *Proc. FTCS-19*, Chicago, IL, June 1989, pp. 348–355.

[3] W. C. Carter and J. Abraham, "Design and evaluation tools for fault-tolerant systems," in *Proc. AIAA Computers in Aerospace Conf.*, 1987, pp. 70–77.

[4] J. R. Armstrong, J. G. Tront, and K. W. Li, "Modeling and simulation of the effects of internal transient upsets on microprocessors," *Trans. Soc. Comput. Simulation*, vol. 2, no. 1, pp. 73–93, 1985.

[5] J.-P. Gérardin, "Aid to the design of reliable and safe systems: The tool DEFI," *Electronique Industrielle*, no. 116, pp. 58–63, Nov. 1986 (in French).

[6] Z. Segall, D. Vrsalovic, D. Siewiorek, D. Yaskin, J. Kownacki, J. Barton, D. Rancey, A. Robinson, and T. Lin, "FIAT—Fault injection based automated testing environment," in *Proc. FTCS-18*, Tokyo, Japan, June 1988, pp. 102–107.

[7] G. S. Choi, R. K. Iyer, and V. Carreno, "A fault behavior model for an avionic microprocessor: A case study," in *Proc. 1st Int. Working Conf. Dependable Computing for Critical Applications*, Santa Barbara, CA, Aug. 1989, pp. 71–77.

[8] D. Powell, "A hierarchical approach to distribution computing system dependability evaluation," *J. Syst. Software*, vol. 1, no. 2, pp. 183–198, 1986.

[9] M. M. Alidrisi, "A simulation approach for computing system reliability," *Microelectron. Rel.*, vol. 27, no. 3, pp. 463–467, 1987.

[10] A. T. Acree, "On mutation," Ph.D. dissertation, Georgia Inst. Technol., Atlanta, ICS Rep. 80/12, Aug. 1980.

[11] R. A. DeMillo, R. J. Lipton, and F. G. Sayward, "Hints on test data selection: Help for the practicing programmer," *Computer*, pp. 34–41, Apr. 1978.
[12] W. E. Howden, "Weak mutation testing and completeness of test sets," *IEEE Trans. Software Eng.*, vol. SE-8, pp. 371–379, July 1982.
[13] A. Mahmood, D. M. Andrews, and E. J. McCluskey, "Executable assertions and flight software," in *Proc. 6th AIAA/IEEE Digital Avionics Systems Conf.*, Baltimore, MD, Dec. 1984, pp. 346–351.
[14] Raytheon Co., "Brassboard fault-tolerant spaceborne computer (BFTSC)," Final Rep., Dec. 1978.
[15] Y. Crouzet and B. Decouty, "Measurements of fault detection mechanisms efficiency: Results," in *Proc. FTCS-12*, Santa Monica, CA, June 1982, pp. 373–376.
[16] F. Morillon, "Physical fault simulation," in *Proc. Int. Symp. EUROCON'82*, Copenhagen, Denmark, June 1982, pp. 489–492.
[17] J. H. Lala, "Fault detection, isolation and reconfiguration in FTMP: Methods and experimental results," in *Proc. AIAA/IEEE Digitals Avionics Systems Conf.*, Nov. 1983, pp. 21.3.1–21.3.9.
[18] A. Damm, "Experimental evaluation of error-detection and self-checking coverage of components of a distributed real-time system," Doctorate thesis, Tech. Univ. Vienna, Oct. 1988.
[19] M. E. Schmid, R. L. Trapp, A. E. Davidoff, and G. Masson, "Upset exposure by means of abstraction verification," in *Proc. FTCS-12*, Santa Monica, CA, June 1982, pp. 237–244.
[20] M. A. Schuette, J. P. Shen, D. P. Siewiorek, and Y. X. Zhu, "Experimental evaluation of two concurrent error detection schemes," in *Proc. FTCS-16*, Vienna, Austria, July 1986, pp. 138–143.
[21] M. L. Côrtes, S. D. Millman, H. A. Goosen, and E. J. McCluskey, " Techniques for injecting non stuck-at faults," Stanford Univ., CRC Tech. Rep. 87-21, Mar. 1987.
[22] U. Gunneflo, J. Karlsson, and J. Torin, "Evaluation of error detection schemes using fault injection by heavy-ion radiation," in *Proc. FTCS-19*, Chicago, IL, June 1989, pp. 340–347.
[23] W. G. Bouricius, W. C. Carter, and P. R. Schneider, "Reliability modeling techniques for self-repairing computer systems," in *Proc. 24th ACM Nat. Conf.*, 1969, pp. 295–309.
[24] M. K. Joseph and J. Bannister, "Coverage estimation and validation," Aerospace Corp. Rep., Aug. 1988.
[25] A. Avizienis and J.-C. Laprie, "Dependable computing: From concepts to design diversity," *Proc. IEEE*, pp. 629–638, May 1986.
[26] W. N. Toy, "Fault-tolerant design of local ESS processors," *Proc. IEEE*, vol. 66, pp. 1126–1145, Oct. 1978.
[27] D. C. Bossen and M. Y. Hsiao, "Model for transient and permanent error-detection and fault-isolation coverage," *IBM J. Res. Develop.*, vol. 26, no. 1, pp. 67–77, Jan. 1982.
[28] J. W. Duran and J. Wiorkowski, "Capture-recapture sampling for estimating software error content," *IEEE Trans. Software Eng.*, vol. SE-7, no. 1, pp. 147–148, Jan. 1981.
[29] K. G. Shin and Y. H. Lee, "Measurement and application of fault latency," *IEEE Trans. Comput.*, vol. C-35, pp. 370–375, Apr. 1986.
[30] R. Chillarege and R. K. Iyer, "Measurement-based analysis of error latency," *IEEE Trans. Comput.*, vol. C-36, pp. 529–537, May 1987.
[31] J. McGough, "Effects of near-coincident faults in multiprocessor systems," in *Proc. AIAA/IEEE Digitals Avionics Systems Conf.*, Nov. 1983, pp. 16.6.1–16.6.7.
[32] J. F. Lawless, *Statistical Models and Methods for Lifetime Data*. New York: Wiley, 1982.
[33] J. B. Dugan and K. S. Trivedi, "Coverage modeling for dependability analysis of fault-tolerant systems," *IEEE Trans. Comput.*, vol. 38, pp. 775–787, June 1989.
[34] J. Arlat and J.-C. Laprie, "On the dependability evaluation of high safety systems," in *Proc. FTCS-15*, Ann Arbor, MI, June 1985, pp. 318–323.
[35] W. C. Carter, "Hardware and software dependability evaluation: System dependability," in *Proc. IFIP 11th World Computer Congress*, San Francisco, CA, Aug. 1989, p. 118.
[36] V. D. Agrawal, "Sampling techniques for determining fault coverage in LSI circuits," *J. Digital Syst.*, vol. 5, no. 3, pp. 189–202, 1981.
[37] N. R. Mann, R. E. Schaffer, and N. D. Singpurwalla, *Methods for Statistical Analysis of Reliability and Life Data*. New York: Wiley, 1974.
[38] R. Chillarege and N. S. Bowen, "Understanding large system failures—A fault injection experiment," in *Proc. FTCS-19*, Chicago, IL, June 1989, pp. 356–363.
[39] F. Traverse, "On the dependability validation by mutation—Design of an error injection tool," Doctor-Engineer thesis, INP Toulouse, Dec. 1983 (in French).
[40] M. Sevestre, "Validation of a microprocessor-based railway safety system," in *Proc. 4th Int. Conf. Rel. and Maint.*, Perros-Guirec and Trégastel, France, May 1984, pp. 586–589 (in French).
[41] D. Powell, P. Verissimo, G. Bonn, F. Waeselynck, and D. Seaton, "The Delta-4 approach to dependability in open distributed computing systems," in *Proc. FTCS-18*, Tokyo, Japan, June 1988, pp. 246–251.
[42] J. Arlat, Y. Crouzet, and L. Amat, "Dependability validation by means of fault-injection: Application of MESSALINE for the validation of the processing unit of a computerized interlocking system," LAAS Res. Rep. 87-108, Mar. 1987 (in French).
[43] M. Aguera, J. Arlat, Y. Crouzet, J.-C. Fabre, E. Martins, and D. Powell, "Results of fault-injection into an MCS network attachment controller with limited self-checking," LAAS Res. Rep. 89-071, Mar. 1989.

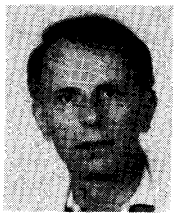
Jean Arlat (M'80) was born in Toulouse, France, in 1953. He received the Certified Engineer degree from the National Institute of Applied Sciences of Toulouse (INSAT) in 1976 and the Doctor-Engineer degree from the National Polytechnic Institute of Toulouse (INPT) in 1979.

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. During 1979–1980 he spent a postdoctoral year in the Department of Computer Science of the University of California at Los Angeles, 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 IEEE Computer Society and of the Fault-Tolerant and Simulation Technical Committees. 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).

Martine Aguera was born in Carcassonne, France, in 1957. She received the "Diplôme d'Enseignement Supérieur de Technologie" (DEST) in automatic control from the Centre National des Arts et Metiers (CNAM), Toulouse, France, in 1987.

She gained experience about microprocessor architectures in the company CEIS Espace, France, with the realization of the French ground station of the SARSAT System (Search and Rescue Aided by SATellite) in 1980–1983. She is now working in Information Processing and Instrumentation Support Service at LAAS-CNRS. In this context, she is responsible for the implementation of the Delta-4 Network Testbed at LAAS.



Louis Amat was born in Toulouse, France, in 1939. He obtained the Speciality Doctorate degree in 1965 from the University of Toulouse.

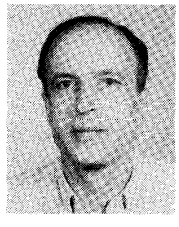
During 1965–1968, he worked in the Electronical Laboratory of the Thomson-Houston French company. He is now working in the Information Processing and Instrumentation Support Service at LAAS-CNRS. In this context, he is responsible for the design and the realization of the MESSALINE fault-injector.

Yves Crouzet was born in Toulouse, France, in 1952. He received the Engineer degree from the Higher National School of Electronics, Electrical Engineering, Computer Science and Hydraulics, Toulouse, in 1975, and the Engineering Doctorate from the National Polytechnic Institute, Toulouse, in 1978.

He is currently "Chargé de Recherche" at the National Center for Scientific Research (CNRS). Since 1975 he has been a member of the Dependable Computing and Fault-Tolerance group at LAAS-CNRS. During 1975–1982 he worked on the design and realization of self-checking VLSI circuits. Since 1982 his research interests have concerned the experimental validation of dependable systems by fault injection.

Jean-Charles Fabre was born in Toulouse, France, in 1957. He received the Master's degree and Diploma for further studies in computer science in 1980, and the Doctorate in 1982, both from the University of Toulouse.

From 1980 to 1983 he was first at LCR (Central Research Laboratory of the Thomson Company) and then at INRIA working in the field of Distributed Computing Systems Architectures, especially with the Chorus project at INRIA. He became a permanent researcher (Chargé de Recherche) of INRIA in January 1984 and joined the Dependable Computing and Fault-Tolerance group at LAAS-CNRS. He is currently involved in the LAAS-INRIA Saturne project and the European Esprit Delta-4 project. His current interest concerns distributed algorithms, fault and intrusion-tolerance in distributed systems, and implementation validation by fault-injection. He was a consultant with the Chorus Systèmes Company on the Colombus project of the European Space Agency and has also served as an expert for a project of the European Esprit program.



Jean-Claude Laprie (M'83) received the Certified Engineer degree from the Higher National School for Aeronautical Constructions, Toulouse, France, in 1968, and the Doctor of 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.

Eliane Martins was born in Rio de Janeiro, Brazil, in 1955. She received the B.S. and M.S. degrees in computer science from the Rio de Janeiro Federal University (UFRJ) in 1976 and 1982, respectively.

She is now a Ph.D. student at the Ecole Nationale Supérieure de l'Aéronautique et de l'Espace (ENSAE) in Toulouse, France. Her research interests include methodologies and tools for protocol and distributed systems testing and the experimental validation of the hardware and software fault-tolerant systems. She is currently working with the Dependable Computing and Fault Tolerance group of the LAAS-CNRS, and her dissertation concerns the experimental validation of an open distributed dependable architecture using the fault injection approach. She worked in the design and implementation of software for mainframe, mini-, and microcomputers for the Brazilian Research Space Institute.

David Powell was born in Greenwich, England, in 1951 and received the Bachelor of Science degree in electronic engineering from the University of Southampton, England, in 1972.

He has been at LAAS-CNRS since 1972 and is a member of the Dependable Computing and Fault-Tolerance group. He obtained the Speciality and State Doctorates in 1975 and 1981, respectively. His current research work in the Dependable Computing and Fault-Tolerance group at LAAS concerns fault-tolerance and security in distributed computing systems. He has written over 30 papers for international and national journals and conferences and hold a patent for a fault and damage-tolerant network for data transmission. He has managed several national and European research constracts and carried out consultancy work with several aerospace, telecommunication and data processing companies in France. He is currently Directeur de Recherche at the National Center for Scientific Research (CNRS) and the Scientific Director of the European Esprit Delta-4 project.