Evaluation of Safety-Oriented Two-Version Architectures

Juan A. Carrasco and Joan Figueras
Departament d'Enginyeria Electrònica
Universitat Politècnica de Catalunya
Diagonal 647, plta. 9
08028 Barcelona, Spain

Annie Kuntzmann
CISI Ingenierie
France

Except for formatting details, this version matches exactly the version published with the same title and author in *Journal of Systems and Software,* vol. 14, no. 3, 1991, pp. 155–162

Abstract

A Markov model taking into account physical and design faults for a two-version architecture oriented to safety-related applications is developed. Only a probabilistic knowledge of the initial state of the versions in relation to the presence of design faults is assumed. The model can be split into two submodels accounting separately for physical and design faults, and a closed form expression for the unsafety of the system is obtained. The parameter estimation problem is discussed and a method to predict the probability distribution of the number of related design faults at the beginning of the operational life of the system is proposed. The method uses a pool model to process fault-occurrence data collected during a "face-to-face" debugging of the two versions. It has by nature a limited capability for proving version diversity, but it is shown that the limit is of the order of the diversity reported by recent experiments on real software. Finally, the impact of version correction during operation is shown to be negligible for critical applications.

1 Introduction

One of the most critical problems faced in the production of fault-tolerant systems is how to monitor architecture design together with the development process in order to meet the dependability and performance specifications ensuring cost effectiveness. Evaluation methods encompassing physical (hardware) faults and design faults introduced during the specification and development processes might help solve this problem. The need for such a combined evaluation has recently been pointed out [10]. In critical applications, the complexity of current software and hardware designs makes it no longer possible to cope with design faults using only a fault-avoidance approach, and fault-tolerance techniques should be considered. Design diversity [1] is a suggested approach to provide design fault-tolerance.

Design diversity is applied differently in safety-oriented and reliability-oriented applications. Safety-oriented applications do not require complete fault-tolerance but merely error detection. For applications of this type, design diversity is implemented using two versions and a comparison monitor, which takes the system to a safe failure state when the outputs of the two versions differ. For reliability-oriented applications, design faults have to be tolerated. In applications of this type, three or more versions are used in conjuction with a majority voter to decide which of the results provided by the versions are to be issued. The efficiency of design diversity depends critically on the extent to which the versions fail independently. Experimental results recently carried out on multiversion software seem to indicate that even though the versions do not fail independently [7], multiversion systems improve significantly single-version systems in both safety and reliability-oriented applications [8]. In our opinion, since hardware design is similar in many aspects to software development, a parallel promise stands in relation to hardware design faults.

Some work has been done toward the combined modeling of physical and design faults [3, 10, 12], but, to the best of our knowledge, no evaluation of multiversion systems incorporating both types of faults has been carried out. In this paper, a Markov model to evaluate the unsafety of a two-version system with version correction during operation is developed and solved. This architecture can match the requirements of most safety-oriented applications at a reasonable development cost.

2 Taxonomy

2.1 Architecture

Figure 1 shows the architecture under evaluation. Two independently designed computation channels in synchronous operation share a given set of inputs and produce separate outputs that are compared by a totally self-checking (TSC) comparison monitor. The output of the system is taken from the first channel and is considered valid as long as no failure indication is given by the monitor. The comparison monitor can be implemented using well-known techniques [9]. We would like to point out that no assumption is made regarding the particular implementation of the channels.

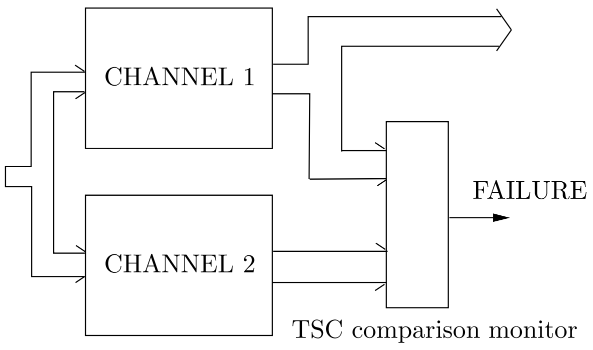

Figure 1: Architecture of a two-version system.

They could be special-purpose VLSI systems or programmed systems. The only requirement is that correct outputs from both channels should be equal.

2.2 Fault Model

The faults considered for the development of the model can be classified as follows:

* physical faults
  - in channels
    + unrelated
    + related
  - in the monitor
    + benign
    + latent
* design faults (in channels)
  + unrelated
  + related

Physical faults are permanent or temporary malfunctions of the hardware resulting from physical degradation processes or external disturbances. They can affect the channels or the monitor. Physical faults affecting only one channel are called unrelated faults; physical faults affecting both channels are called related faults. Related physical faults produce erroneous, but usually not identical, outputs in both channels. Physical monitor faults are classified into benign and latent. Benign faults are within the self-checking capability of the monitor and produce a FAILURE indication. Latent faults fall outside the self-checking capability of the monitor and do not produce that indication.

It is assumed that the comparison monitor is design fault-free. Design faults are viewed as regions in the system input space, which in general would consist of sequence of input vectors.

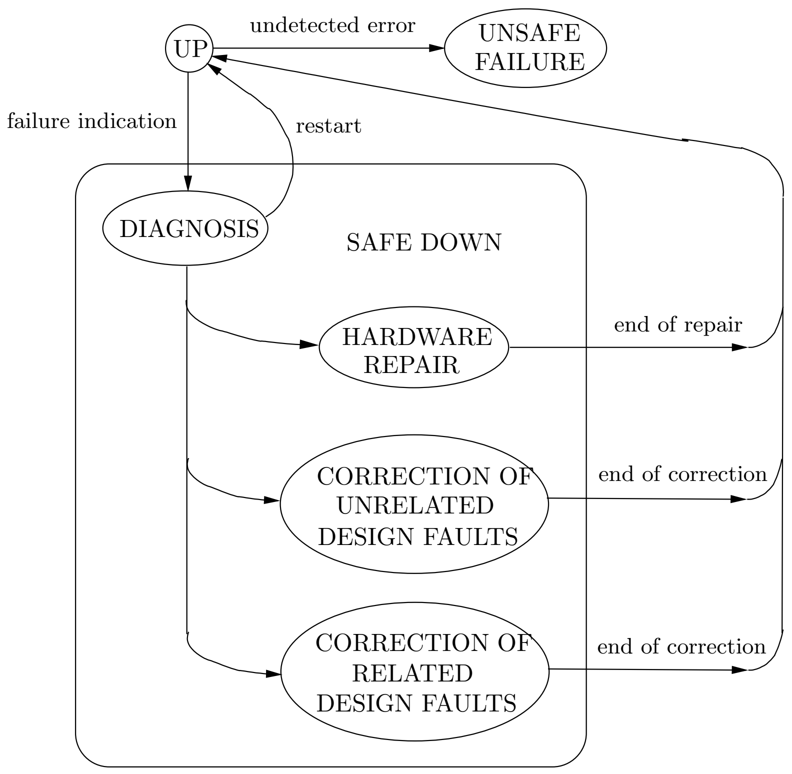

Figure 2: Conceptual behavioral model.

When the input enters the region associated with a design fault, the fault is activated and an error occurs. In our model, regions in different versions are either disjoint (unrelated design faults) or coincident (related design faults). Related design faults may or may not produce identical errors.

2.3 Behavioral model

The behavioral model is obtained by combining the fault model with the maintenance strategy and is shown in Figure 2. An unsafe failure occurs if an erroneous output is given without a FAILURE indication. This happens if a related physical or design fault produces identical errors in both channels, or if the first channel gives an erroneous output in the presence of a latent fault in the comparison monitor preventing the recognition of the disagreement. When a FAILURE indication is issued, the operation of the system is stopped for diagnosis. If a permanent physical fault or a design fault is found, a maintenance operation starts. Otherwise, a transient fault is assumed and the system is immediately restarted. From a purely statistical point of view, it would seem reasonable to restart the system's operation when a design fault is diagnosed and perform the correction off-line: the system is, after all, as good as it was before. However, from a psychological point of view, this does not seem reasonable, even if the design fault is belived to be confined to only one channel.

From the user's point of view, the system alternates between the UP and SAFE DOWN states until the undesired unsafe failure occurs. In order to quantify the dependability of the system, we will use the unsafety $US(t)$, defined as the probability of having an unsafe failure over the first t time units of operation in the UP state, i.e., ignoring the time spent in the SAFE DOWN state. This choice is motivated by the following reasons:

1) it is a reasonable one in the context of our measure,
2) it gives an upper bound for the unsafety computed over the real time,
3) the bound is tight in the frequent case in which the maintenance and correction rates are much higher than the failure rates,
4) it simplifies the model significantly.

A stochastic model for the evaluation of the unsafety of the system is developed in the next section.

3 The Evaluation Model

3.1 Model Hypotheses

The actual faulty behavior of the comparison monitor in coordination with the channels is complex. First, the self-checking attribute of the comparison monitor only implies that benign faults are detected (cause a FAILURE indication) for some pairs of channel outputs. Hence, in fact, benign faults are "latent" for some time. In addition, a latent fault in the comparison monitor can result in a FAILURE indication for some agreeing pair of channel outputs or in the absence for some disagreeing pair. A detailed model would be dependent on the actual design of the comparison monitor and on the actual operation of the channels since the production of the fault, and would be in general very difficult, if not impossible, to build. A simpler model will be used in this paper. The latency of bening faults will be neglected and it will be assumed, pessimistically, that an unsafe failure follows immediately after a latent fault. Under these hypotheses and bearing in mind that the unsafety has been defined over the "up" time, only the faults leading to an unsafe failure need be considered.

Having defined the unsafety over the "up" time, only the repair processes modifying the production of faults leading to an unsafe failure (critical faults) need be modeled. Since physical failure processes are not modified by repair actions and the system has only one operational mode, critical physical faults are modeled by a constant rate λ_{CP}, which can be obtained by adding the rate of related physical faults producing identical errors and the rate of monitor latent faults.

Unrelated design faults are always detected and can be ignored in the model. Related design faults can cause an unsafe failure and have to be considered. An activation/correction model for related design faults is needed. The one proposed is a generalization of the Goel and Okumoto model [4], and is described by the following hypotheses:

1) initially there are k related design faults with probability q_k, $k \geq 1$,

2) the total related design fault activation rate when $k \geq 1$ faults are present is ψ_k,

3) a related design fault, when activated, causes identical errors with probability c (error correlation),

4) an activated related desing fault causing a disagreement is diagnosed as such with probability E_d (diagnosis efficiency) and with probability $1 - E_d$ is treated as a transient physical fault[1], i.e., the system is restarted without correction,

5) a diagnosed related design fault is properly corrected in a version with probability E_c (correction efficiency),

6) a related design fault not properly corrected in any version leaves the same related design fault activation rate as the system had before.

Thus, with probability c the activation of a related design fault causes an unsafe failure, with probability

$$\alpha = (1 - c)E_d \left[1 - (1 - E_c)^2\right] \tag{1}$$

the fault is removed from the system, and with probability $1 - c - \alpha$ either the fault is not well diagnosed and therefore not removed, or is improperly corrected in both versions, in both cases leaving the system with the same related design fault activation rate as it had before.

3.2 Model Solution

Assume now that the number of initial related faults k is bounded by n (we will show later how to choose a suitable value for n). Then, it is possible to describe the evolution of the system on the up states until its first unsafe failure by the homogeneous, continuous-time Markov chain depicted in Figure 3a. Considering that all the up states have a common transition with rate λ_{CP} to the unsafe state, it is possible to decompose the model in two submodels: one accounting for critical physical faults (Figure 3b) and a second one accounting for related design faults (Figure 3c). The safety $S(t)$ can be expressed as

$$S(t) = S_p(t)S_d(t)$$

where $S_p(t)$ (physical safety) and $S_d(t)$ (design safety) are obtained using the submodels. When dealing with safety-related applications it is often more significant the unsafety $US(t) = 1 - S(t)$ or probability of having an unsafe failure over the first t time units. $US(t)$ can be computed from $US_p(t)$ and $US_d(t)$ using

$$US(t) = US_p(t) + US_d(t) - US_p(t)\,US_d(t)$$

[1]Note that the model neglects the probability of diagnosing a design fault for only one version, which is estimated to be a very unlikely event, considering that related desing faults have a common cause.

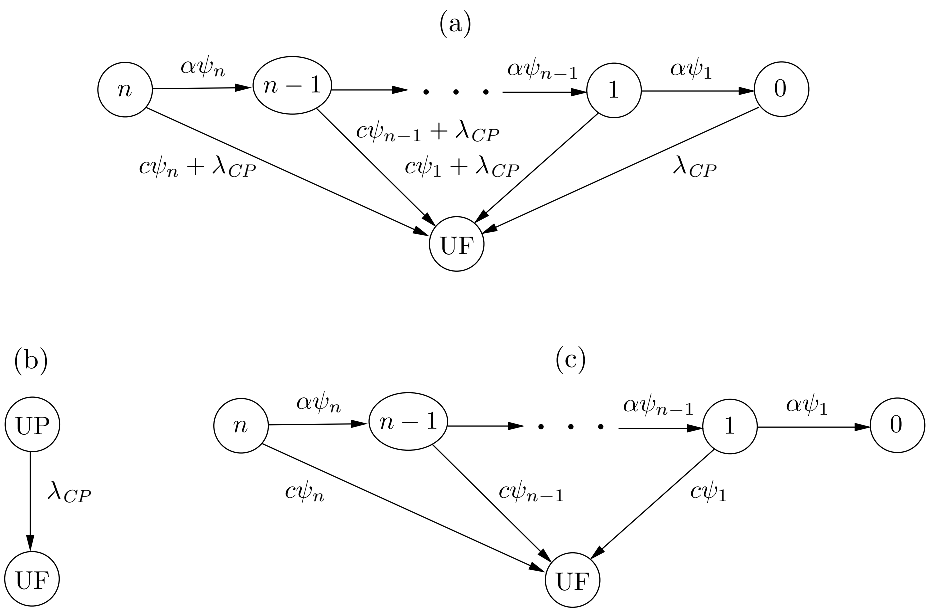

Figure 3: Unsafety evaluation model (a) and its decomposition into a physical unsafety evaluation model (b) and a design unsafety evaluation model (c).

The model decomposition is interesting because it makes it possible to analyze separately the contributions of the two types of faults to the overall unsafety. The unsafety due to critical physical faults is given by

$$US_p(t) = 1 - e^{-\lambda_{CP} t}$$

The evaluation of $US_d(t)$ requires the transient solution of the Markov chain depicted in Figure 3c. Let q_i be the probability that the initial number of related faults is i, $0 \leq i \leq n$. Under the assumption $\psi_i \neq \psi_j$ for $i \neq j$, which is true for the model, the following closed-form solution (see Appendix) can be obtained.

$$US_d(t) = \sum_{i=1}^{n} q_i A_i - \sum_{k=1}^{n} \left(\sum_{i=k}^{n} q_i B_k^i \right) e^{-\rho_k t} \tag{2}$$

where

$$\rho_k = (c + \alpha)\psi_k \tag{3}$$

$$A_i = 1 - \left(\frac{\alpha}{c + \alpha} \right)^i \tag{4}$$

$$B_k^i = \frac{c}{\rho_k} \sum_{l=1}^{k} \left(\frac{\alpha}{c+\alpha} \right)^{i-l} \frac{\prod_{m=l}^{i} \psi_m}{\prod_{\substack{m=l \\ m \neq k}}^{i} (\psi_m - \psi_k)} . \tag{5}$$

Two particular cases are worth mentioning: a) total error correlation ($c = 1$), and b) no version correction. The latter can be obtained from the general model by making $E_d = 0$. In both cases

$\alpha = 0$ (1) and the model of Figure 3c is reduced to a Markov chain having only transitions from the states $i = 1, \ldots, n$ to the unsafe state, with rates ψ_i for case a and $c\psi_i$ for case b. These models are easily solved yielding

$$US_d(t) = \sum_{i=1}^{n} q_i \left(1 - e^{-\psi_i t}\right) \qquad \text{for case a} \tag{6}$$

$$US_d(t) = \sum_{i=1}^{n} q_i \left(1 - e^{-c\psi_i t}\right) \qquad \text{for case b} \tag{7}$$

Let us consider the problem of selecting a truncation value n for the initial number of related design faults. The limit design unsafety is given by (2)

$$US_d(\infty) = \sum_{i=1}^{n} q_i A_i \tag{8}$$

where $A_i = 1$ for the particular cases. A suitable criterion is to take n so that the relative truncation error in $US_d(\infty)$ is lower than a specified tolerance TOL. According to (4), $A_i \leq 1$. Then, it suffices to take the smallest n with

$$\frac{1 - \sum_{i=0}^{n} q_i}{\sum_{i=0}^{n} q_i A_i} < TOL\,,$$

since the numerator is an upper bound of the absolute truncation error and the denominator is a lower bound of the absolute value.

4 Model Parameter Estimation

The critical physical fault rate λ_{CP} can be evaluated by architectural and circuit analysis, fault injection, etc., and will not be discussed here. As it will be shown in the next section, the other most influencial parameters are c, q_k, and ψ_k. When the system has only one output signal, it can be ensured that $c = 1$. Otherwise, c is likely to be smaller. In general, the higher the design level to which the fault belongs, the more likely it is that the errors resulting from related design faults will be correlated. Analysis of the modular structure of the two versions and correlation with statistical data could be used to estimate c. Of course, it is also possible to take the pessimistic assumption that $c = 1$. In this case, the simpler model (6) can be used.

Perhaps the most difficult problem is the estimation of the distribution of the number of remaining related faults when the operation of the system starts. A suggestive approach is to use an underlying model for the production of design faults accounting for related faults. Such a model should include parameters characterizing: a) the complexity of the design, b) the specification methods, c) the level of diversity of the development methods and tools, and d) the mastery of the designers and tools. However, we consider it doubtful that such an approach will be workable because of

1) the difficulty in indentifying a reduced set of significant parameters,

2) the limited amount of data on related design faults,

3) the rapid evolution of design methods and tools.

Our approach is a generalization of the methods currently used in "software science" to monitor the reliability growth during debugging and estimate the software reliability at the release point. Our suggestion is as follows.

Once both versions have been cleaned out of coarse flaws, they are debugged in parallel using the same test inputs ("face-to-face" debugging). Errors are monitored and design faults corrected. At a given point, both versions are considered good enough to be released for operation. Let D_1 and D_2 be the number of observed (and corrected) unrelated design faults in, respectively, each version, and D_r the number of observed (and corrected) related design faults. "Software science" (see, for instance, [11]) can be used to estimate the number of unrelated design faults in each version before the "face-to-face" debugging started. Let N_1 and N_2 be the number of those faults, $N_u = N_1 + N_2$ and $D_u = D_1 + D_2$. Now, we can think in the debugging process as a sample without replacement from a "pool" containing two types of objects:

1) N_u unrelated design faults,

2) N_r related design faults,

and our problem is the estimation of $N_r - D_r$, or equivalently N_r, knowing the outcome of the sample (D_u and D_t) and N_u.

The only assumption of the model is that all faults are extracted from the "pool" with the same probability. This is equivalent to assume that for the debugging input sequence all design faults are activated at the same rate. Using this "pool" model it is possible to evaluate "a posteriori" probabilities q'_k and from them the "a priori" probabilities q_k of having $k = N_r - D_r$ related design faults after the sample. This yields

$$q_k = \frac{q'_k}{\sum_{l=0}^{\infty} q'_l} \tag{9}$$

where the "a posteriori" probabilities can be computed by

$$q'_l = \frac{\binom{N_u}{D_u}\binom{D_r + l}{D_r}}{\binom{N_u + D_r + l}{D_u + D_r}} \tag{10}$$

A potential drawback of the method is that the amount of fault data collected during the "face-to-face" debugging establishes a bound on the provable diversity. The bound for given N_u and D_u is obtained for $D_r = 0$ and is illustrated in Figure 4, where the predicted average number of remaining related design faults ANF is plotted against the number of undetected unrelated design faults for

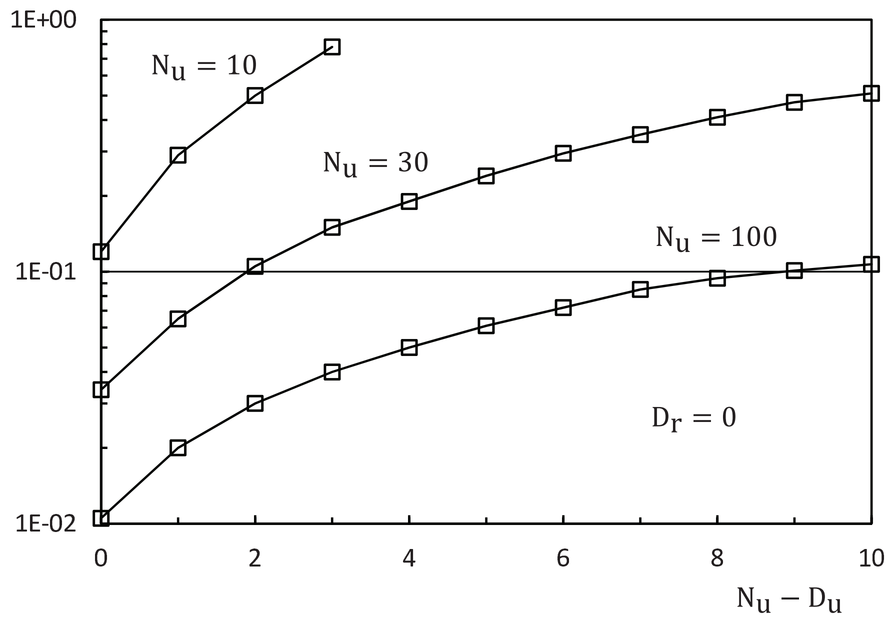

Figure 4: Average number of related faults predicted at the beginning of the operation of the system when no faults of this class are observed during debugging.

several values of N_u. The results clearly show the convenience of carrying out an extensive "face-to-face" debugging to minimize the number of undetected unrelated design faults $N_u - D_u$ and starting it as soon as possible, to maximize N_u. The former is limited by economical factors, the latter by the fact that coarse faults may obey different statistics.

Experimental results for two-version software [8] have given an average probability of error detection in a two-version system of 0.9968. This is within the bound imposed by the method proposed if a number of unrelated faults for complex software (the applicatiosn of interest) is observed during the "face-to-face" debugging. For instance, if the number of unrelated faults detected in each version is 49, and the system is released when it is estiamted that one unrelated design fault remains in each version, the estimate for the average number of related design faults when the versions are released can be as good as 0.03 if $D_r = 0$ (see Fig. 4 with $N_u = 100$). Then, if we assume $c = 0.2$ and that all the design faults are activated during operation at the same rate ψ, the total activation rate of design faults will be approximately 2ψ, whereas coincident errors will be produced only with rate $(0.2)(0.03)\psi$. This gives a provable error detection probability bound of 0.9974, which is slightly large than that observed empirically.

In order to estimate ψ_k, it can be assumed that all design faults have the same activation rate, irrespective of whether they are related or not. This assumption si also used in software growth models for single-version systems [6]. Then, $\psi = k\psi$. The activation rate per design fault could be estimated from the rates observed during debugging. Usually, special-case inputs rather than random inputs would be used in order to accelerate the debugging. In this case, an appropriate correction factor, which could be estimated by correlating the observed failure rates with those obtained with random inputs, should be used.

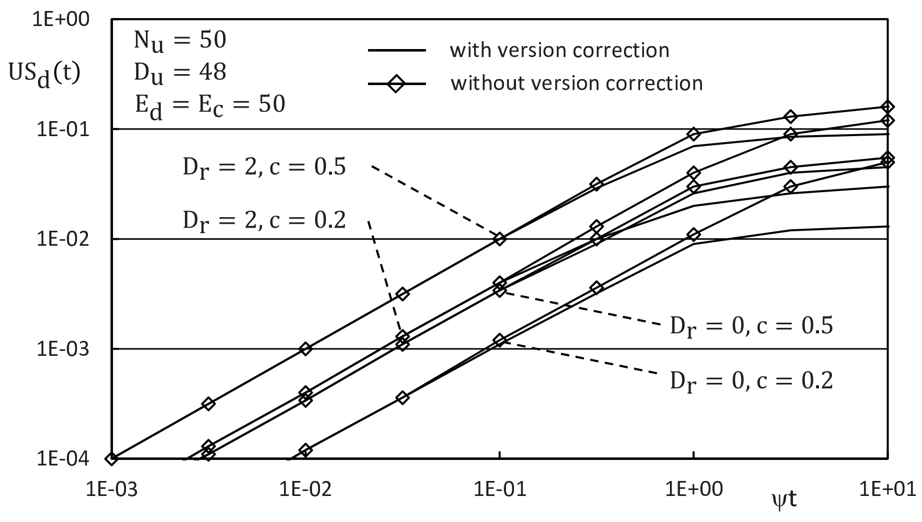

Figure 5: Influence of error correlation and distribution of initial number of related design faults in the design unsafety with and without version correction during operation.

5 Model Analysis

In this section a qualitative analysis of the unsafety of two-version systems using the model developed in Section 3 will be carried out. Since the behavior of $US_p(t)$ is trivial, only $US_d(t)$ will be considered.

Figure 5 illustrates the influence of the error correlation factor and the initial distribution of related design faults on the design unsafety. Two cases are considered: operation with version correction and operation without version correction. The initial distribution of related design faults is computed using the method proposed in the previous section using two sets of values for N_u, D_u, and D_r, differing only in the value of D_r. The design unsafety is evaluated using (1)–(5) for the case with version correction, and (7) for the case without version correction, with $\psi_k = k\psi$. It can be seen that both c and the initial number of related design faults have an important impact on the design unsafety. In addition, and at first sight suprisingly, the initial behavior is independent of whether related design faults are corrected. This is due to the fact that fault correction necessarily follows fault ocurrence and the initial behavior of $US_d(t)$ is mainly determined by the activation of the first fault. The conclusion is that for critical applications, where a very low probability of unsafe failure has to be guaranteed, version correction during operation does not help, at least from a statistical point of view.

The impact of the diagnosis and correction efficiency is thus limited to the asymptotic behavior of the design unsafety ans is analyzed in Figure 6. It can be seen that diagnosis efficiency is more important than correction efficiency, and that moderate values for both are enough.

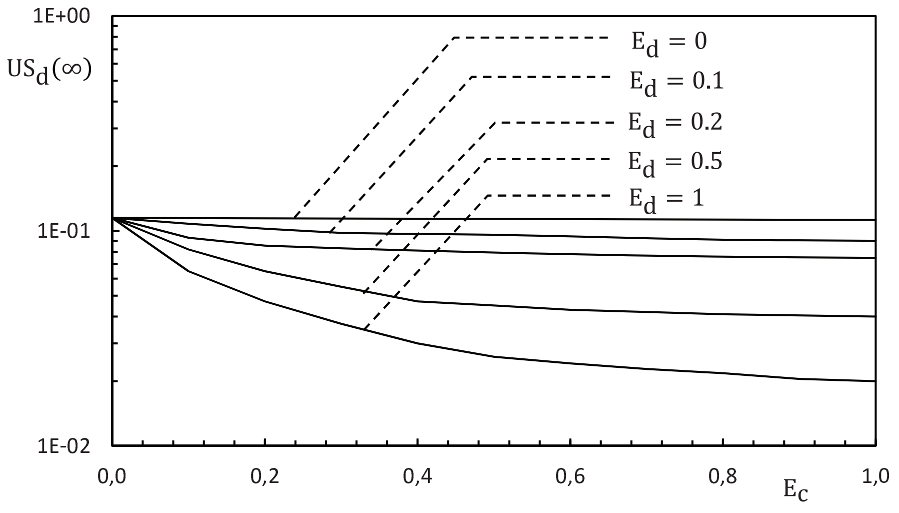

Figure 6: Impact of diagnosis and correction efficiencies on the asymtotic behavior of the design unsafety.

6 Conclusions

Starting from a behavioral model, an evaluation model for a two-version architecture for safety-oriented applications has been developed. The model is sufficiently simple for a closed form expression for the unsafety to be obtained. The problem of parameter estimation has been studied and a method for the prediction of the distribution of the number of related design faults at the beginning of the operation of the system has been proposed. The method has the advantage of not requiring an underlying model for the production of design faults during the specification and design processes. It has been shown that, if the face-to-face" debugging is carried out from an early stage, the method is capable of predicting diversities of the order of magnitude reported in recent experiments for two-version systems. For a higher number of versions the estimate might be coarse if the versions are very diverse.

By analyzing the model it has been shown that version correction during operation has a negligible influence during the period of interest for critical applications. This has two consequences. First, from a practical point of view, it stresses the need for extensive debugging before operation, even if different versions are used. Second, from a modeling point of view, only the initial related design fault rate is significant and correction need not be modeled. It must be emphasized that these conclusions apply onlt to two-version architectures and critical applications, where a very low unsafety has to be guaranteed.

Currently we are considering the application of a similar methodology for the modeling of three-version systems for reliability-oriented applications. Much more complex Markov models are needed for these systems. A software tool, METFAC [2], is being used to define and process the models.

A Derivation of the closed-form expression for $US_d(t)$

$US_d(t)$ is the probability of being in the unsafe failure state at time t. Since the state is absorbing and cannot be reached from the state 0 (see Figure 3c),

$$US_d(t) = \sum_{i=1}^{n} q_i p_{i,UF}(t) \tag{11}$$

where $p_{ij}(t)$ are the interval transition probabilities of the continuous-time Markov chain ($p_{ij}(t)$ is the probability that the chain is in state j at time t given that it was in state i at the initial time).

Let

$$\rho_k = (c+\alpha)\psi_k \,. \tag{12}$$

The transition probabilities are governed by the set of differential equations (see, for instance [5])

$$\frac{dp_{ii}}{dt} = -\rho_i p_{ii}(t)$$

$$\frac{dp_{ij}}{dt} = -\rho_j p_{ij}(t) + \alpha\psi_{j+1} p_{i,j+1}(t)\,, \quad 1 \le j < i$$

$$\frac{dp_{i,UF}}{dt} = \sum_{j=1}^{i} c\psi_j p_{ij}(t)$$

with initial conditions

$$p_{ii}(0) = 1$$

$$p_{ij}(0) = 0\,, \quad 1 \le j < i$$

$$p_{i,UF}(0) = 0\,.$$

Using the Laplace transform the following linear system is obtained:

$$sP_{ii}(s) - 1 = -\rho_i P_{ii}(s)$$

$$sP_{ij}(s) = -\rho_j P_{ij}(s) + \alpha\psi_{j+1} P_{i,j+1}(s)\,, \quad 1 \le j < i$$

$$sP_{i,UF}(s) = \sum_{j=1}^{i} c\psi_j P_{ij}(s)\,. \tag{13}$$

The system can easily be solved iteratively in $P_{ij}(s)$, resulting

$$P_{ij}(s) = \frac{\alpha^{i-j} \displaystyle\prod_{k=j+1}^{i} \psi_k}{\displaystyle\prod_{k=j}^{i} (s+\rho_k)}\,, \quad 1 \le j \le i\,.$$

By substituing in (13), the following expression is obtained:

$$P_{i,UF}(s) = \sum_{j=1}^{i} c\psi_j \alpha^{i-j} \frac{\displaystyle\prod_{k=j+1}^{i} \psi_k}{s \displaystyle\prod_{k=j}^{i} (s+\rho_k)} .$$

After fractional expansion, and making use of the fact that $\psi_i \neq \psi_j$ for $i \neq j$, the reverse Laplace transfrom can be found to be

$$p_{i,UF}(t) = \sum_{j=1}^{i} c\psi_j \alpha^{i-j} \frac{\displaystyle\prod_{k=j+1}^{i} \psi_k}{\displaystyle\prod_{k=j}^{i} \rho_k} + \sum_{j=1}^{i} c\psi_j \alpha^{i-j} \left(\prod_{k=j+1}^{i} \psi_k \right) \sum_{k=j}^{i} C_{jk}^{i} e^{-\rho_k t}$$

with

$$C_{jk}^{i} = -\frac{1}{\rho_k \displaystyle\prod_{\substack{l=j \\ l\neq k}}^{i} (\rho_l - \rho_k)} .$$

After some algebraic manipulations and changes of indices in the summations one obtains

$$p_{i,UF}(t) = A_i - \sum_{k=1}^{i} B_k^i e^{-\rho_k t} \tag{14}$$

with

$$A_i = 1 - \left(\frac{\alpha}{c+\alpha} \right)^i \tag{15}$$

$$B_k^i = \frac{c}{\rho : k} \sum_{l=1}^{k} \left(\frac{\alpha}{c+\alpha} \right)^{i-l} \frac{\displaystyle\prod_{m=l}^{i} \psi_m}{\displaystyle\prod_{\substack{m=l \\ m\neq k}}^{i} (\psi_m - \psi_k)} . \tag{16}$$

Finally, by substitution of (14) in (11):

$$US_d(t) = \sum_{i=1}^{n} q_i A_i - \sum_{i=1}^{n} q_i \sum_{k=1}^{i} B_k^i e^{-\rho_k t}$$

and by rearranging the summations of the second term:

$$US_d(t) = \sum_{i=1}^{n} q_i A_i - \sum_{k=1}^{n} \left(\sum_{i=k}^{n} q_i B_k^i \right) e^{-\rho_k t} . \tag{17}$$

The closed-form solution is defined by (17), (15), (16), and (12).

Acknowledgments

This research work was supported by the ESPRIT project 1609 SMART (System Measurement and Architecture Techniques) and by the CICYT ("Comisión Interministerial de Ciencia y Tecnología").

References

[1] A. Avizienis, The N-version Approach to Fault-Tolerant Software, *IEEE Trans. on Software Eng.*, SE-11, 1491–1501 (1985).

[2] J. A. Carrasco and J. Figueras, METFAC: Design and Implementation of a Software Tool for Modeling and Evaluation of Complex Fault-Tolerant Computing Systems, *Proc. 16th Int. Symp. on Fayult-Tolerant Computing FTCS-16*, Vienna, Austria, July 1986, pp. 424–429.

[3] A. Costes, C. Landrault, and J. C. Laprie, Reliability and Availability Models for Maintained Systems Featuring Hardware and Design Faults, *IEEE Trans. on Computers*, C-27, 548–560 (1978).

[4] A. L. Goel and K. Okumoto, An Analysis of Recurrent Software Failures ina Real-Time Control System, *Proc. ACM Ann. Tech. Conf.*, Washington, DC, 1978, pp. 496–500.

[5] R. A. Howard, *Dynamic Probabilistic Systems, Volume II: Semi-Markov and Decision Processes*, John Wiley and Sons, Inc., New York, 1971.

[6] Z. Jelinsky and P. Moranda, Software reliability research, in *Statistical Computer Performance Evaluation*, (W. Freiberger, Ed.), Academic, New York, 1972, pp. 465–484.

[7] J. C. Knight, N. G. Levenson, and L. D. St. Jean, A Large-Scale Experiment in N-version Programming, *Proc. 15th Int. Symp. on Fault-Tolerant Computing FTCS-15*, Ann Arbor, Michigan, June 1985, pp. 135–139.

[8] J. C. Knight and N. G. Levenson, An Emperical Study of Failure Probabilities in Multiple-Version Software, *Proc. 16th Int. Symp. on Fault-Tolerant Computing FTCS-16*, Vienna, Austria, July 1986, pp. 165–170.

[9] P. K. Lala. *Fault Tolerant and Fault Testable Hardware Design*, Prentice-Hall Inc., London, 1983.

[10] J. C. Laprie, Dependability Evaluation of Software Systems in Operation, *IEEE Trans. on Software Eng.*, SE-10, 701–714 (1984).

[11] P. N. Misra, Software Reliability Analysis, *IBM Systems Journal*, 22, 262–270 (1983).

[12] U. Sumita and Y. Masuda, Analysis of Software Availability/Reliability Under the Influence of Hardware Failures, *IEEE Trans. on Software Eng.*, SE-12, 32–41 (1986).