Comments

Some Critical Comments on the Paper "An Optimal Approach to Fault Tolerant Software Systems Design" by Gannon and Shapiro

P. A. LEE, J. L. LLOYD, AND S. K. SHRIVASTAVA

Fault tolerance in software systems is becoming increasingly important for the attainment of high reliability in computing systems. Therefore, it was with some interest that we noted the above paper that appeared in the September 1978 issue of IEEE TRANSACTIONS ON SOFTWARE ENGINEERING (vol. SE-4, pp. 390–409). However, in reading this paper, we were somewhat concerned about some of the concepts expressed, mainly in the first two sections of the paper, which we feel give a misleading impression of certain aspects of fault tolerance. Moreover, an examination of the mathematics presented in Appendix A reveals many inaccuracies.

We would make the following points.

1) When fault tolerance is being discussed, one must carefully define the types of faults that are being considered. The types Gannon and Shapiro are considering are only those faults that can occur in the hardware components of the system, although their definition of a fault tolerant system (p. 391) may confuse the reader in that it includes ". . . software fault conditions." It is true that hardware faults have in the past been the subject of much of the work on fault tolerance. However, it is generally agreed that other faults in systems, for example, faults in the software, are a major cause of unreliability whose effects should not be ignored. In the Bell ESS systems, for example, processor hardware failures account for only $\frac{1}{5}$ of the total downtime [6]. While this low figure is dependent upon the hardware redundancy provided, it seems clear that there are many other sources of faults in complex systems, and not to consider their effects can provide a most misleading impression.

Even if one wishes to concentrate solely on hardware faults, it would seem that those considered by Gannon and Shapiro form at most a very limited set, comprising essentially those faults that can arbitrarily change the bits representing a word, with a single bit parity check being the only form of hardware error detection provided. In practice, fault tolerant systems usually have several (hardware) error detection mechanisms to detect the different types of fault that could occur (for example, see [6]). No justification is given by Gannon and Shapiro for the limited viewpoint which they adopt. Some of these points are discussed in greater detail below.

2) Gannon and Shapiro make many references to the short paper by Hill [2]. It seems, however, that many of their references are inaccurate. The authors state (p. 390) that Hill has provided a relative comparison of language constructs that have been devised to provide software recovery in the presence of a detectable hardware fault. In fact, all that Hill discusses is how, in Algol and Fortran, an indication of a fault can be passed to the invoker of a function or procedure, and makes no mention of detectable hardware faults. The authors state (p. 391) that the LABEL mechanism provides ". . . a structured approach to fault recovery." If "structured" is taken with its usual connotation (i.e., as in structured programming), then this is not true—indeed, Hill specifically points this out.

On p. 391 the authors state that the LABEL mechanism "only provides fault coverage for that class of errors which is detectable by the computer hardware." This is not so. Indeed, the only type of error detection discussed by Hill is software parameter checking. It is only to Gannon and Shapiro's paper that this statement applies.

3) It seems clear from the authors' various discussions of the recovery block scheme (e.g., the incorrect footnote on p. 393) that they have not understood the aims and purpose of the scheme, and to compare recovery blocks with Hill's [2] LABEL mechanism (p. 390) and with the approach presented by themselves (p. 391) is *most* misleading. Recovery blocks support the provision of fault tolerant software, which is software that is capable of tolerating *unanticipated faults, particularly in the software itself.* Techniques such as Hill's and Gannon and Shapiro's, now usually termed exception handling techniques, are often quite suitable for handling anticipated erroneous situations, but are *not at all suitable for coping with faults whose exact locations and consequences are unanticipated or unanticipatable.* Faults in design (which is what software faults usually are) are examples of the latter kind of fault. Schemes such as the recovery block scheme can provide tolerance against a wide class of faults, *including design faults.* This distinction is exemplified by Melliar-Smith and Randell [7] who discuss a system employing both techniques. (A thorough treatment of exception handling techniques is given in [3] and [5].)

Moreover, the recovery block scheme is not simply a recovery mechanism. It encompasses all of the techniques that are necessary for any fault tolerant system. For example, one categorization of such techniques [4] identifies error detection, assessment of the damage that may have been caused, error recovery, and fault treatment and the provision of continued service as being necessary constituent parts of fault tolerance. However, many aspects of such techniques are ignored or treated superficially by Gannon and Shapiro. For example, simply terminating the execution of the task as suggested in Section II-C would hardly seem to be the approach of a fault tolerant system.

4) In Section II-C the authors discuss the "fault coverage" of the LABEL mechanism. The statement that ". . . detectable hardware faults . . . represent approximately 58 percent of all possible faults" is most questionable. First, ignoring the fact that the figure of 58 percent is wrong (see below), the statement should, of course, be qualified to indicate that "all possible faults" actually refers to the limited set of faults being considered by Gannon and Shapiro with the percentage referring to those detected by the parity check. Even then, one must seriously question the value of parity codes for monitoring ". . . all computer hardware operations" (p. 404). Even if one ignores the present-day trend towards semiconductor memories with error correcting codes, surely parity alone cannot detect faults in arithmetic units, address decoding circuits, and the like. Second, on p. 404 the authors claim that ". . . the fault coverage of the LABEL mechanism is equal to the percentage of possible faults which are detected by a parity code," and hence by implication is 58 percent. Again this is a statement from which one could be easily misled into believing that all one needs to have for a fault tolerant system is a means of

Manuscript received January 31, 1979; revised July 20, 1980.

The authors are with the Computing Laboratory, The University, Newcastle-upon-Tyne, England.

0098-5589/81/1100-0608$00.75 © 1981 IEEE

error detection. As discussed above, there are many other actions necessary to achieve fault tolerance, and the fault coverage taking these actions into account is probably impossible to determine analytically. The authors can only make the above statements because they are assuming that the fault coverage of these actions is unity (p. 390), a value which one must surely query.

5) The optimization procedure used by the authors consists essentially of maximizing coverage subject to constraints on the cost and fault rate of the recovery scheme. To obtain the necessary data for the optimization, a number of estimates have to be made, many of which would be very difficult to compute. For example, it is necessary to estimate the probabilities of task executions, segment executions, number of times an instruction will be executed within a segment, and so on. One wonders whether realistic figures can be obtained for any real system. Similarly, the recovery scheme cost is taken to be the sum of the estimated number of times the various computer instructions are executed in each recovery segment. The reasoning behind this is that a recovery scheme is assumed to be costly if it is complex, and complexity is assumed to be equatable with the total number of instructions executed when that recovery scheme is invoked!

The authors justify their analytical model by showing the close comparison between its results and those from a simulation run. However, as the system under consideration is rather small (less that 500 words), it is not convincing that the analytical model will be applicable to real systems.

Bearing in mind the limited class of faults being considered by Gannon and Shapiro and the simplistic view taken of the techniques necessary for fault tolerance, it hardly seems worthwhile formulating an optimization technique for the design of the software. Even if the analytical model presented were accurate, many realistic fault tolerant systems are likely to require a coverage factor in excess of 90 percent, for which Figs. 13 and 14 (p. 401) would further indicate that optimizations are not worthwhile.

6) It is somewhat surprising to note that despite the careful refereeing to which papers are usually subjected, an examination of the mathematics presented in Appendix A of Gannon and Shapiro's paper indicates that the derivation of several of these equations is wrong and, moreover, that the evaluation of some equations is incorrect. While we note that the effect of some of these inaccuracies is slight, we feel it necessary that a paper present the correct equations, even if simplifying assumptions for specific cases and values are then applied. We also feel that some of the arguments presented are unnecessarily complicated and specific. More detailed comments on these aspects are to be found in the Appendix to this note.

7) In Appendix A, part B, the authors attempt to determine the fault coverage of the software recovery mechanisms of Section II-C by making some simplifying assumptions about the effectiveness of the range error detection test. Again, when reading the figures, one must bear in mind that the faults which the authors are considering are those which can arbitrarily change the bits in a 16 bit word. Errors arising from faults in arithmetic units or in the software itself are not considered. Thus, the statement (p. 405) that "only 4 percent of the total possible fault conditions remain undetected by the LABEL and software recovery mechanisms" is only valid for the very small class of faults which the authors are considering. Evaluations of real systems based on such figures would, we feel, be meaningless. Also, to determine the fault coverage of the LABEL mechanism solely from the percentage of detectable hardware faults (even if one included faults detected by hardware other than by parity checks) is surely inappropriate, as the mechanism can be used, as mentioned above, for software detected faults.

Some comments on the range test being analyzed by Gannon and Shapiro are also perhaps in order. The authors assume that the range of a parameter ". . . can be estimated to within 10 percent of its current value" (although the coverage is subsequently stated in terms of the expected value). This assumes that the current value is known and available from an error-free word, and has not been erroneously calculated. Furthermore, the possibility of cumulative errors is not considered. The analysis of the range test presented is essentially very simple, and is unnecessarily strongly based on the assumptions about the test that have been made. For example, their analysis does not indicate the significant effect that a change in number representation has—using unsigned integers instead of two's complement reduces the range test coverage to 80 percent from 90 percent. In fact, the presented arguments can be easily generalized to avoid such confusions. Also, it is, we feel, misleading to suggest that a single percentage figure can be applied over the whole range of x, even as a percentage of a current value. This is implicit in the remark that ". . . the fault coverage of the range test is inversely proportional to the expected value of x." In fact, if the expected value of x were zero, only the value zero is acceptable, however generously the percentage is chosen.

8) Some minor points are in order. The reference to Hill's paper is incorrect. The paper appeared in *Computer Journal* (not *Computer*, as stated) and in March 1971 (not March 1972, as stated). Horning *et al.* [1] do not document ". . . concepts of fault tolerant software that have been applied to operating systems" (p. 390). What that paper does is introduce the recovery block scheme. The recovery block scheme is in no way limited to operating systems software. Also, for accuracy, the authors should not attribute the recovery block scheme to Randell (p. 390), but to Horning *et al.* [1].

9) In their conclusions, the authors state ". . . many classes of software fault condition can be shown to have identical characteristics to hardware fault conditions." While other authors have attempted to show such similarities, in general it would seem that the faults that can affect a physical system such as hardware (e.g., wearing out) are hardly likely to occur in software systems. It is now becoming recognized that software and hardware faults should not be classified in the same manner (e.g., see [4] and [8]), and the simple application of fault tolerant techniques that have been applied in hardware systems is unlikely to solve the problems that occur with software. Nevertheless, the increasing complexity of hardware systems with the possibility of such systems containing design faults does mean that in the future, some classes of hardware faults may be shown to have identical characteristics to software fault conditions.

APPENDIX

There would appear to be some mistakes in the mathematics presented in Appendix A, part A (p. 404). The expression (A4) for $E(w)$ simplifies to

$$\frac{1}{2}\left(1 - \frac{1}{2^n - 1}\right)$$

so that $E(w)$ is asymptotic to $\frac{1}{2}$, and for $n = 16$ evaluates to 32767/65535 rather that 32767/78405, as stated. Thus, the figure of 58 percent (p. 404) is incorrect and should be 50 percent.

The expression for $P(w)$ (A7) is also incorrect. The correct expression should be

$$P(w) = \frac{\sum_{i=1}^{n/2} \binom{n}{2i} p^{2i}(1-p)^{n-2i}}{\sum_{i=1}^{n} \binom{n}{i} p^{i}(1-p)^{n-i}}.$$

However, for very small values of p and moderate values of n, there will be little difference between the evaluation of this ex-

pression and the one used by the authors. There is a seemingly simpler and more accurate way to approximate $P(w)$ than that used in (A8). For small values of p, the probability that a bit of a word is in error, the binomial distribution is approximated by the Poisson distribution with mean $k = np$. This gives an expression for $\Pr_i$, the probability of i errors in a word, of

$$\Pr_i = \frac{e^{-k}k^i}{i!}$$

for which

$$\Pr_0 = e^{-k} \text{ and (assuming } n \text{ even)}$$

$$\Pr_{\text{even}} = \Pr_2 + \Pr_4 + \cdots \Pr_n$$

$$\cong e^{-k}(\cosh(k) - 1) = \tfrac{1}{2}(1 - e^{-k})^2.$$

Hence,

$$P(w) = \frac{\Pr_{\text{even}}}{1 - \Pr_0} \cong \frac{1}{2}(1 - e^{-k}) \cong \frac{1}{2}np.$$

For $n = 16$, this gives the answer $P(w) = 8p$, compared with $P(w) = 7.5p$ of (A11) on p. 404.

We also note some errors in the mathematics presented in Appendix A, part B. First, we assume from the rest of the text that the typographical error on p. 405 (". . . a non-bit word") should be (". . .an n + 1-bit word"), and that ". . . signed two's complement form" is not a new form of representation (in two's complement, the sign is implicit). The equation for $S(x)$ (A12) is wrong and should be

$$S(x) = \frac{2(0.1)}{2^n - 1}|x|$$

since the total number of errors in a word represented (as stated) in n bits must be $2^n - 1$, not $2^{n+1} - 1$, as stated in the paper. This is assuming that the parity bit is not included in the evaluation of the total number of errors, as must be the case in this situation. Even if (A12) were correct, its evaluation with $n = 15$ does *not* give the answer stated in (A13),which is a factor of 2 too small. However, as (A13) is not used again (which makes one wonder why its evaluation was attempted in the first instance), this mistake is immaterial.

Following (A13), Gannon and Shapiro state that the worst case for the range test is when the parameter x assumes its largest value. This is wrong: when x assumes its largest value ($x^{\max}$), the possible range is reduced to $(1 - p)x^{\max} \rightarrow x^{\max}$ (where p is the proportion adopted for the range test), since values in the range $x^{\max} + 1 \rightarrow (1 + p)x^{\max}$ have no valid representation. The worst case is actually obtained when $(1 + p)x = x^{\max}$, i.e., $x = x^{\max}/(1 + p)$ and (A14) has to be altered accordingly.

Because of the incorrect evaluation of $E(w)$ in (A6), the results of evaluating (A16) and (A17) (p. 405) are slightly wrong, with the corrected result for (A17) being 0.95. As the authors make no further use of these results, assuming instead that for the 15-data bit word, this figure can be rounded up to 1, the calculations in Section III (based on a PDP-11 with a 16-data bit word) are unaffected by these small inaccuracies.

REFERENCES

[1] J. J. Horning *et al.*, "A program structure for error detection and recovery," in *Lecture Notes in Computer Science*, vol. 16. Berlin: Springer, 1974, pp. 177–193.

[2] I. D. Hill, "Faults in function in ALGOL and FORTRAN," *Computer J.*, vol. 14, pp. 315–316, Mar. 1971.

[3] J. B. Goodenough, "Exception handling: Issues and a proposed notation," *Commun. Ass. Comput. Mach.*, vol. 18, no. 12, pp. 683–696, 1975.

[4] B. Randell, P. A. Lee, and P. C. Treleaven, "Reliability issues in computing system design," *Comput. Surveys*, vol. 10, pp. 123–165, June 1978.

[5] R. B. Levin, "Program structures for exceptional condition handling," Ph.D. dissertation, Carnegie-Mellon Univ., Pittsburgh, PA, 1977.

[6] W. N. Toy, "Fault-tolerant design of local ESS processors," *Proc. IEEE*, vol. 66, pp. 1126–1145, Oct. 1978.

[7] P. M. Melliar-Smith and B. Randell, "Software reliability: The role of programmed exception handling," in *Proc. ACM Conf. Language Design for Rel. Software, Sigplan Notices*, vol. 12, Mar. 1977, pp. 95–100.

[8] A. Avizienis, "Fault-tolerance: The survival attribute of digital systems," *Proc. IEEE*, vol. 66, pp. 1109–1125, Oct. 1978.

Comments on the Critique of Lee, Lloyd, and Shrivastava

THOMAS F. GANNON AND STEPHEN D. SHAPIRO

Our paper gives an engineering approach, based on system modeling and performance evaluation, for the design of fault tolerant software systems. Utilizing a system model and measureable system parameters, resource allocation is used to achieve reliable system performance.

We wish to thank our critics for their comments. However, the comments focus on special cases, examples, and questions of interpretation—not the fundamental notions of our paper.

In particular, 1) referring to the first item in the comments, the context of the quotation cited is: "Finally, a fault tolerant system is normally used to denote system software which will continue to yield correct results in the presence of hardware and software fault conditions. As mentioned previously, the authors only address software recovery procedures which provide recovery for detectable hardware fault conditions." 2) Regarding Hill's paper, we urge the interested reader to examine this paper and draw his own conclusions. 3) We do not claim that the examples of software error recovery and testing presented are identical to those of Randell. 4) Our intent is to provide a realistic approach to fault tolerant system design. From a commercial point of view, the 90 percent fault coverage referred to in the comments is impractical at the present time. It is important that "fault coverage" not be equated with "fault rate." System performance can be very reliable, while at the same time, the fault coverage for the system can be relatively small. 5) The paper by Horning *et al.* does document "concepts" of fault tolerant software design. We did not say that the paper documents instances of operating system applications. 6) Regarding the Appendix, we acknowledge the incorrect evaluation of the expression (A6)—however, the value of this expression has no impact on the results of the paper. Since p is a very small number which cannot be measured to great accuracy, the details on approximation, presented beginning with the discussion of (A7), do not seem to be warranted.

The underlying sources of computing errors we used in the

Manuscript received April 28, 1981.

T. F. Gannon is with Manufacturing Data Systems, Inc., Ann Arbor, MI 48106.

S. D. Shapiro is with the Department of Electrical Engineering, State University of New York at Stony Brook, Stony Brook, NY 11794.

0098-5589/81/1100-0610$00.75