N86-30362

AN EXPERIMENTAL EVALUATION OF ERROR SEEDING AS A PROGRAM VALIDATION TECHNIQUE

John C. Knight   Paul E. Ammann
Department of Computer Science
University of Virginia
Charlottesville, Virginia.

A Summary

Submitted To The Tenth Annual Software Engineering Workshop
Goddard Space Flight Center
Greenbelt, Maryland.

The *error seeding* technique was originally proposed by Mills [1] as a method for determining when a program has been adequately tested using functional or random testing. The procedure resulted from a desire to apply statistical methods to the problem of predicting the number of errors in a program in the hope that the number of errors discovered during testing could be used to estimate the number of remaining undetected errors. The method involves deliberately introducing or *seeding* artificial errors into a program and subsequently testing that program.

Error seeding has the desirable property that it is apparently simple to employ and it provides a stopping condition for testing. Unfortunately, it has the major drawback that, in order to work effectively and for the existing statistical model to apply, it relies upon the following three assumptions:

(1) Indigenous errors, those introduced by the programmer, are all approximately equally difficult to locate.

(2) Seeded errors are approximately as difficult to locate as indigenous errors.

(3) Errors, whether indigenous or seeded, do not interfere with one another.

A priori there is no reason to believe that any of these assumptions hold. The first and third seem reasonable. However, error seeding has been criticized on the basis of the second assumption. It seems unlikely that realistic seeded errors can be generated but no definitive, empirical evidence for any of the assumptions has been gathered previously. We have performed an experiment designed to check the validity of each of the underlying assumptions. In particular, we were interested in evaluating very simple, syntax-based algorithms for generating seeded errors.

Briefly, as part of a separate experiment [2, 3], twenty-seven Pascal programs have been written independently by *different* programmers to a single specification. Thus all twenty-seven are intended to perform the same function, the processing of radar data in a simple antimissile system. As part of the other experiment, the programs have been subjected to one million tests, and a great deal is known about the indigenous errors present in the programs. These programs represent an excellent starting point for an experiment with error seeding. Any results obtained can be averaged thereby eliminating any bias attributable to individual programmers.

In the error seeding experiment, seventeen of the twenty-seven programs were selected at random, errors were seeded into all seventeen, and the resulting programs were tested. The algorithms used for seeding errors were very simple: two algorithms modified the bounds on **for** statements, three algorithms modified the Boolean expression in **if** statments, and one algorithm deleted assignment statements. Each of these algorithms was applied four times to each of the 17 programs for a total of 408 modified programs, each of which contained one seeded error. The programs were tested using 25,000 of the 1,000,000 test cases from the previous experiment.

The metric used for evaluating the seeded errors was the mean time to failure (MTF). The MTF for a particular program containing a seeded error is defined as the average number of test cases executed between detected failures. The MTF's for the seeded errors had a wide range. Some seeded errors caused a failure on every test case; some had a very small number of failures in 25,000 test cases; and others caused *no failures at all* in 25,000 test cases. We conclude that it *is* possible to generate seeded errors that are arbitrarily difficult to locate, albeit at the expense of creating others that are easy to locate. These results suggest, surprisingly, that it is possible to comply with the second assumption listed above.

An examination of the MTF's of the *indigenous* errors revealed a similar wide range of failure rates. In fact, there was a very strong resemblance in mean time to failure between the resilient seeded errors and the indigenous errors. However, in neither case were errors equally likely to be discovered, in conflict with the first assumption cited above.

Finally it was discovered during the experiment that in two cases a seeded error corrected, or partially corrected, an indigenous error. Clearly, the implication is that assumption three above was violated. We conclude that the first and third assumptions, those that seem most believable, are in fact violated, and that the second, the one that seems totally unreasonable, can be complied with. Using the data from this experiment, the underlying model of error seeding can be modified and error seeding made a useful, practical technique.

REFERENCES

(1) Mills, H.D., "On The Statistical Validation of Computer Programs", in *Software Productivity*, Little Brown, Toronto.

(2) Knight, J.C., and N.G. Leveson, "A Large-Scale Experiment In N-Version Programming", Proceedings of the *Ninth Annual Software Engineering Workshop*, NASA Goddard Space Flight Center, November 1984, Greenbelt, MD.

(3) Knight J.C., and N.G. Leveson, "A Large Scale Experiment In N-Version Programming" Digest of Papers FTCS-15: *Fifteenth Annual Symposium on Fault-Tolerant Computing*, June 1985, Ann Arbor, MI.