Further Empirical Studies of Test Effectiveness *

Phyllis G. Frankl
Oleg Iakounenko

Computer and Information Sciences Dept.
Polytechnic University
6 Metrotech Center
Brooklyn, N.Y. 11201
e-mail: phyllis@morph.poly.edu

Abstract

This paper reports on an empirical evaluation of the fault-detecting ability of two white-box software testing techniques: decision coverage (branch testing) and the all-uses data flow testing criterion. Each subject program was tested using a very large number of randomly generated test sets. For each test set, the extent to which it satisfied the given testing criterion was measured and it was determined whether or not the test set detected a program fault. These data were used to explore the relationship between the coverage achieved by test sets and the likelihood that they will detect a fault.

Previous experiments of this nature have used relatively small subject programs and/or have used programs with seeded faults. In contrast, the subjects used here were eight versions of an antenna configuration program written for the European Space Agency, each consisting of over 10,000 lines of C code.

For each of the subject programs studied, the likelihood of detecting a fault increased sharply as very high coverage levels were reached. Thus, this data supports the belief that these testing techniques can be more effective than random testing. However, the magnitudes of the increases were rather inconsistent and it was difficult to achieve high coverage levels.

1 Introduction

White-box testing techniques based on control flow and data flow analysis have been widely studied in the software testing research literature, but there is no definitive answer to the question of how effective these techniques are at finding software faults. In the absence of data indicating how effective (and how expensive) these techniques are, it is hardly surprising that they have not been adopted widely in practice. On the other hand, there is no easy answer to this question. This paper contributes toward answering this question, by presenting results of an empirical study performed on eight versions of a "real-world" C program, each with a naturally occurring fault.

In white-box testing a set of test requirements is derived by examining the source code of the program being tested. These requirements can be used as a basis for assessing whether a given test data set is "adequate". For example, in *statement testing*, a test set is considered adequate if it causes the execution of every statement in the program and in *branch testing* (or *decision coverage*) a test set is considered adequate if it causes every edge in the control flow graph to be traversed (equivalently, if it causes every boolean expression controlling a decision or looping construct to evaluate to true at least once and to false at least once.) In the *all-uses* data flow testing criterion [10, 4], each test requirement is a *definition-use association (dua)*, i.e., a triple $(d, u, \mathtt{v})$, where `v` is a variable, d is a program point where `v` is defined, and u is a program point where `v` is used; to cover such a requirement a test case must execute a path that goes from d to u without redefining `v`.

Although these techniques can, in principle, be used as the basis for test generation, it is more practical to use them as *adequacy criteria* intended to determine how thoroughly a test set exercises the program. To use an adequacy criterion of this nature, one generates a set of test data (typically without regard to the adequacy criterion), instruments the program being tested, executes the instrumented program on the test set, and uses the results of the instrumentation to determine the *coverage level* achieved by the test set, i.e. the proportion of requirements that have been satisfied. If the coverage level is too low, the tester may generate additional test cases using the original test generation technique or may select additional test cases targeted to requirements that have not yet been covered. In practice, it is often difficult or impossible to achieve 100% coverage of a test data adequacy criterion. This is because some of the requirements may be unexecutable and others, while executable, may be very difficult to cover. When these criteria are used in practice the tester typically strives to attain a fairly high coverage level, without shooting for 100% coverage. In this paper, we explore how the effectiveness of testing with a given criterion varies as coverage increases.

Eight subject programs are considered. These programs were selected from a suite of 33 versions of an antenna configuration program, developed by Ingegneria Dei Sistemi (Pisa, Italy) for the European Space Agency. Each version has a

*Supported in part by NSF Grant CCR-9206910.

different fault that actually occurred and was discovered as the program was developed. This suite of programs was previously used by Pasquini, *et al.* for an experiment on software reliability models [9]. For the current experiment, we selected from the suite, those programs that appeared to have the lowest failure rates. The selection procedure is described in more detail, below.

2 Experiment Design

It is difficult to measure the effectiveness of an adequacy criterion in a meaningful way. Consider an erroneous program P, its specification S, and a test data adequacy criterion C. Even if we restrict the size of the test sets to be considered, there are a large number of different test sets that satisfy criterion C for P and S. These adequate test sets typically have different properties: some may detect one or many faults, while others detect no faults; some may be difficult to execute and check while others are easy. Thus, in defining and measuring effectiveness (or cost) of testing criterion C for P and S, it is not sufficient to consider a single representative test set. Rather, the space of test sets must be sampled and appropriate statistical techniques must be used to interpret the results.

We previously developed an experiment design that addresses this issue and used it to compare the effectiveness of all-uses to that of branch testing for a suite of nine small programs [2]. We subsequently refined the experiment design and used it to compare all-uses to mutation testing on the same suite [3]. Hutchins *et al.* used a similar design to study branch and data flow testing on a suite of programs with seeded faults [7].

There are several plausible probabilistic measures of the effectiveness of an adequacy criterion. The measure considered here is $Eff(P, C, \mathcal{D})$, the probability that a C-adequate test set for program P, selected according to distribution $\mathcal{D}$ on the space of all such test sets, will detect *at least one fault*. A variant of this measure, in which the distribution $\mathcal{D}$ arises from an idealized test generation strategy, has been the subject of numerous analytical investigations of test effectiveness. In our experiments, the distribution $\mathcal{D}$ arises from a somewhat more realistic test generation strategy, in which a universe of possible test cases is created and then test sets of a given size are randomly selected from that universe.

For a given subject program P and adequacy criterion C the experiment procedure is as follows:

1. Generate a *universe* of test cases. The universe is a large set of test cases from which test sets will be selected. For these experiments, the universe consisted of 10,000 test cases. They were randomly generated using the test generator developed by Pasquini *et al.* for their experiments on reliability growth [9] models, which generated test cases according to a modeled operational distribution.

2. Construct a *coverage matrix*. The coverage matrix has one row for each test case and one column for each test requirement (decision or definition-use association). Entry (i, j) is '1' if test case i covers requirement j and is '0' otherwise. The ATAC software testing tool [6], along with some pre and post-processing scripts, were used to identify the requirements and to determine which requirements each test case satisfied.

3. Construct a *results vector*. This vector has one entry for each test case, indicating whether or not that test case detects a fault in the subject program. To construct this vector, the subject program was run on each test case and the results were compared to the "correct" version of the program, i.e., the version from which all of the detected faults had been removed.

4. Simulate the execution of a large number of test sets of a given size. To simulate the execution of a single test set of size s, randomly select s rows of the coverage matrix, '`or`' them together to determine the total number of requirements covered and '`or`' together the corresponding entries in the results vector to determine whether the test set exposes a fault. Determine the coverage level c of that test set (requirements covered divided by (total number of requirement minus number of requirements deemed unexecutable[1])) and increment `total[c]` for that coverage level. If the test set exposes a fault increment `exposing[c]` for that coverage level. Repeat this for a large number of test sets of the given size.

5. Determine estimates of effectiveness and error bounds on those estimates. For a given coverage level c, Let

$$\begin{aligned} n_c &= \Sigma_{i \geq c} \texttt{total[i]} \\ x_c &= \Sigma_{i \geq c} \texttt{exposing[i]} \\ \hat{p_c} &= x_c / n_c. \end{aligned}$$

   Then $\hat{p_c}$ provides an estimate of the proportion p of (size s) test sets of coverage *at least* c that detect a fault. The values of `exposing[i]` and `total[i]` can also be used to calculate confidence intervals around those estimates [11]. Using the normal approximation method for confidence intervals around an estimate of a binomial parameter,

$$e_c = 1.96\sqrt{\hat{p_c}(1 - \hat{p_c})/n_c}$$

   approximates half the size of the 95% confidence interval around the estimate $\hat{p_c}$, provided that $n_c\hat{p_c}(1 - \hat{p_c}) > 5$. That is, the probability that the true value of proportion p falls outside the interval

$$(\hat{p_c} - e_c, \hat{p_c} + e_c)$$

   is less than 0.05.

   Note that the normal approximation can never be applied when $\hat{p_c} = 1.0$. For such points, we derived the confidence interval by referring to a table of exact confidence limits [11].

The benefits and limitations of this design are discussed in detail elsewhere [3]. One of the main benefits is that it allows us to control for test set size. In general, test sets that satisfy (or almost satisfy) more sophisticated adequacy criteria, such as dua coverage, tend to be larger than those that satisfy less sophisticated criteria, such as decision coverage. Many software testing experiments have, in essence, compared large test sets that satisfy one testing criterion $C1$ to smaller test sets that satisfy another testing criterion $C2$ [8]. With such experiments, it is impossible to tell whether any reported benefit of $C1$ is due to its inherent properties, or due to the fact that the test sets considered are larger, hence more likely to detect faults.

[1] Those requirements that were not executed by any test case in the universe were considered to be unexecutable.

In contrast, in our experiments we fix a test set size s and compare test sets of size s satisfying $C1$ to test sets of the same size satisfying $C2$. Thus we know that any reported differences in effectiveness are not artifacts of test set size. Our design also facilitates comparisons with random testing without use of an adequacy criterion, as the effectiveness of those test sets is computed when we consider sufficiently low coverage levels.

3 Subject Programs and Test Universe Generation

The subject programs are derived from an antenna configuration system developed by professional programmers. The system provides a language-oriented user interface for configuration of antenna arrays. Users enter a high level description and the program computes the antenna orientations.

During integration testing and operational use as faults were discovered and corrected, the faulty versions were maintained. For their experiments on reliability growth models, Pasquini *et al.* encapsulated the incorrect and corrected code for each correction in `#ifdef/#else/#endif` directives, so the faulty versions could be easily isolated. The final version had a very low failure rate (less than 10^{-4} with 99.99% confidence) when tested using an operational distribution and no additional failures after extensive use. We use this final version as a test oracle.

The source code along with all of the `#ifdef/#else/#endif` directives is 13,968 lines of C code (including comments and white space). The final (oracle) version source code has 11,640 lines of code after preprocessing to remove the faulty code.

Pasquini *et al.* developed a test generator for this program [9] for use in an investigation of reliability growth models. We used this generator to generate a universe of 10,000 test cases.

In each of the versions we reintroduced a single fault (by preprocessing with the appropriate `#ifdef` enabled). This yielded a suite of 33 programs, each with a single fault. This is somewhat artificial, since in the actual development process the programs that occurred were the one with all of the faults, the one with all of the faults except fault number 1, the one with all of the faults except faults numbers 1 and 2, etc. However it allowed us to create numerous subject programs with real faults and with low failure rates. Also, it is possible that the data gathered using this approach (isolated faults) could provide insight into what kinds of faults the different testing techniques are good at detecting.

We were primarily interested in programs with low failure rates. We expect adequacy criteria like decision coverage and all-uses to be applied (if at all) to programs that have already undergone a fair amount of testing and debugging, thus have relatively low failure rates. In addition the question of which adequacy criterion works best is moot for programs with high failure rates: if the failure rate is so high that almost any reasonable test set will detect a fault, then there will be little or no distinction between those tests that satisfy a sophisticated adequacy criterion and those that do not.

We estimated the failure rates of the 33 program versions by running them each on about 5,000 test cases, then selected those versions with the estimated failure rates below 0.015 for use in these experiments. There were 11 such programs. With two of these programs (versions 27 and 33) we had problems running the program versions that had been instrumented with ATAC. Due to human error we omitted one such program (version 2). For one of the versions that was selected (version 12) the failure rate when the program was run on all of the test cases in the universe was slightly higher than 0.015; we included this subject in the study. Table 1 shows the subject program version numbers, their failure rates (when executed on the entire universe), the numbers of decisions (branches) and definition-use associations identified by ATAC, and the number of decisions and definition-use associations that are unexecutable (relative to this universe).

The choice of test set sizes was somewhat arbitrary. If test sets are too small, it will be difficult, or even impossible, to achieve high coverage levels. On the other hand, if they are too large, it will be difficult to achieve low coverage levels (for comparison) and the test sets will be more likely to expose a fault, possibly obscuring the relationship between coverage and effectiveness. To some extent, one can compensate for use of smaller test sets, by using more of them. Even if it is fairly unlikely that a test set of size s will achieve a high coverage level c, by running enough test sets, we can obtain a statistically significant number of test sets at coverage level c. The second to last column of Table 1 indicates the test set sizes used and the last column indicates the number of test sets selected. Note that in spite of the very large numbers of test sets used, the number of test sets achieving the highest coverage levels was usually fairly small.

4 Results and Discussion

Graphs of coverage versus effectiveness for the decision coverage and all-uses criteria for the eight subject programs are shown in Figures 1 to 8. **Decision coverage** results are plotted with **TRIANGLES** and **all-uses** results with **SQUARES** . A plotted point (x, y) indicates that $y \times 100\%$ of the test sets that achieved coverage level of *at least* x exposed the fault in the given version (that is, $y = \hat{p_x}$). Note that the ranges and scales on the y-axes are different for different subject programs.

The 95% confidence intervals around each such point estimate are shown with vertical bars. For most of the plotted points, these error bars are extremely small, often smaller than the triangles and boxes. However, for some of the points, mainly those at high coverage levels, the error bars are more pronounced. This is because it was difficult to achieve very high coverage levels, thus the sample size (number of test sets at or above the given coverage level) was fairly small for high coverage levels. Consequently the confidence in the accuracy of those estimates is smaller, or in other words, the confidence intervals are larger.

In each of these graphs, a coverage value of 1.0 represents test sets that covered all of the *executable* decisions or definition-use associations. (Readers who prefer to consider the coverage value in terms of the total number of requirements, rather than in terms of the number of executable requirements can use the data in Table 1 to re-calibrate the coordinates on the x-axis.) Recall that in this paper *executable* means executable relative to the universe, i.e., that some test case in the universe covered the requirement.

For example, the square at $(0.96, 0.055)$ in Figure 1 indicates that 5.5% of the size 200 test sets in the sample for Version 1 that covered at least 96% of the executable definition-use associations (equivalently, $96\% \times (5255 - 1437)/5255 = 70\%$ of all of the definition-use associations) detected the fault. The vertical bar (of length 0.022) through that point indicates that the probability is less than 0.05 that the true percentage of fault-exposing test sets among *all* size 200 test

subject	decis	unexec decis	duas	unexec duas	failure rate	test set size	number test sets
Version1	1175	353	5255	1437	0.0001	200	10^5
Version3	1175	353	5255	1437	0.0001	200	10^5
Version7	1171	353	5235	1437	0.0150	20	10^6
Version8	1171	353	5235	1437	0.0094	20	10^6
Version12	1175	353	5256	1437	0.0185	100	10^6
Version18	1175	353	5256	1437	0.0014	50	10^6
Version22	1169	352	5198	1411	0.0036	50	10^6
Version32	1175	353	5256	1437	0.0001	100	10^6

Table 1: Subject Programs

sets achieving that coverage level that could be drawn from the universe is outside the interval $(0.44, 0.66)$.

These graphs can be used to do several different kinds of comparisons:

- Compare the effectiveness of coverage level at least c to that of coverage level at least c' for a given criterion. This allows investigation of whether effectiveness increases as coverage increases, and if so, the magnitude of the increase, whether the increase is monotonic, whether it is linear, etc.
- Compare the effectiveness of coverage level at least c for decision coverage to that of coverage level at least c' for all-uses. This provides some insight into whether all-uses is a better choice than decision coverage (disregarding other factors, such as cost).
- Compare the effectiveness of coverage level at least c for a given criterion to the effectiveness of random testing without any adequacy criterion. To do this, notice that in each graph, the plotted points at sufficiently low coverage levels lie on a horizontal line. The y coordinate of these points represents the effectiveness of random test sets of the given size drawn uniformly from the universe. By extending this line to the right, one can determine visually whether test sets achieving coverage level c are significantly above the line, hence whether such test sets are more effective than random ones. In addition, the probability that random test sets of some other size s' detect a fault can be calculated from the failure rate data and can be used to answer questions like "how big an increase in the size s' of a random test would be needed to achieve the same effectiveness as that achieved by size s tests of a given coverage level?". Such calculations (not included in this paper) give some insight into whether the gains in effectiveness when high coverage is reached are large enough to justify the expensive of using a coverage criterion.

The graphs for Versions 1 and 3 appear to be identical. In fact, it turns out that, although the faults in these programs are different (in fact, occurring in different functions, written in different files), they are quite similar and both versions fail on precisely the same set of test cases. Version 32 also has a similar graph and, in fact, fails on the same set of test cases as Versions 1 and 2. The graph for Version 32 is somewhat different, only because it is based on a smaller test set size.

For all eight of the subjects, for both decision coverage and all-uses, effectiveness is significantly higher for high coverage levels than for low coverage levels. Effectiveness was close to 1.0 for the highest levels of dua coverage for Version 8 and of both dua and decision coverage for Version 12. For the other subjects, however, the effectiveness, even at the highest coverage levels considered was still rather low, numerically. Given the cost of using these criteria, this calls into question the cost effectiveness of the techniques.

The performance of decision coverage and dua coverage is quite similar in almost all of the subjects. This is in sharp contrast to our previous experiments on smaller programs [3], in which decision coverage was generally little better than random testing without a coverage criterion.

For some subjects, the highest level of decision coverage was more effective than that level of dua coverage. This may appear to contradict folk-wisdom about the relative power of the criteria and the results of analytical comparisons of those criteria [5]. However, it is important to note that *these experiments control for test set size:* we are comparing test sets of size n that reach decision coverage level c to test sets of the *same size* that reach dua-coverage level c'. Also note that the versions in which decision coverage surpasses dua coverage (Versions 1, 3, and perhaps 32) are the three versions that had the same set of failure points, so it can be argued that this phenomenon occurred less often than it appears.

It appears that for Version 22, the effectiveness of dua coverage falls at the highest coverage level considered. In previous experiments [3] we observed that effectiveness did not always increase monotonically as coverage increased. This point could be an instance of that phenomenon. Alternatively, it is possible that this is an instance where the true value of the proportion p_c is outside the 95% confidence interval $(\hat{p_c} - e_c, \hat{p_c} + e_c)$ around the measured point estimate. Considering the very large number of point estimates measured in these experiments, it is reasonably likely that a few of the actual proportions would lie outside the confidence intervals.

In all of the subjects except Version 7, there is a sharp increase in effectiveness when the very highest coverage level is reached. This phenomenon, which also occurred in our earlier experiments indicates that the benefit of using these coverage criterion often does not kick in until quite high coverage levels are achieved.

Comparison to Related Work

The experiments most closely related to the work reported here are our previous work [2] and the study by Hutchins, *et al.* [7]. Both of these studies compared effectiveness of a data flow based criterion, branch testing, and random testing without an adequacy criterion by sampling the space of adequate test sets in a meaningful way.

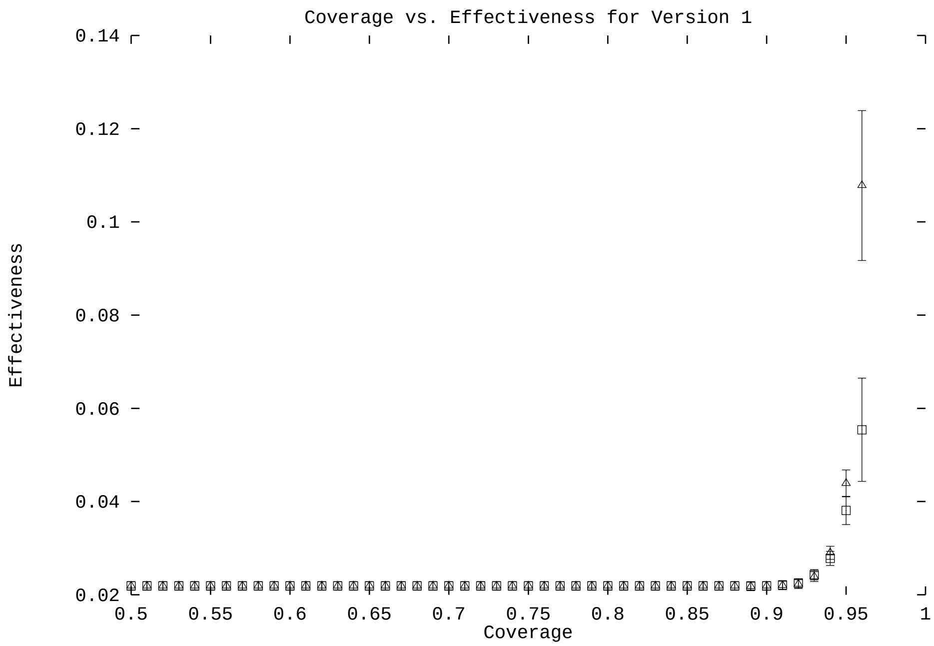

Figure 1: Coverage vs. effectiveness for Version 1. Size 200.

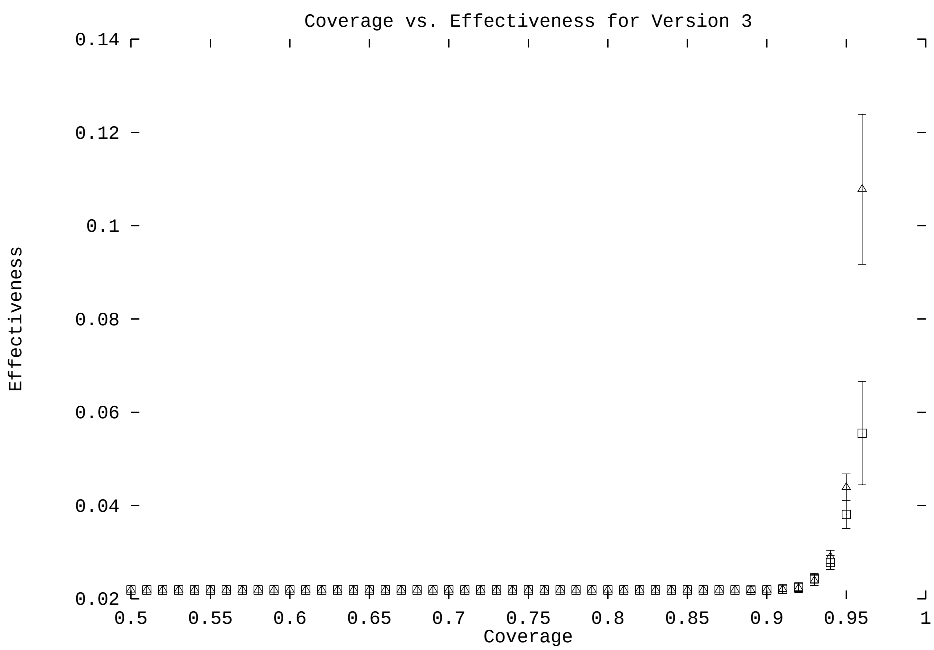

Figure 2: Coverage vs. effectiveness for Version 3. Size 200.

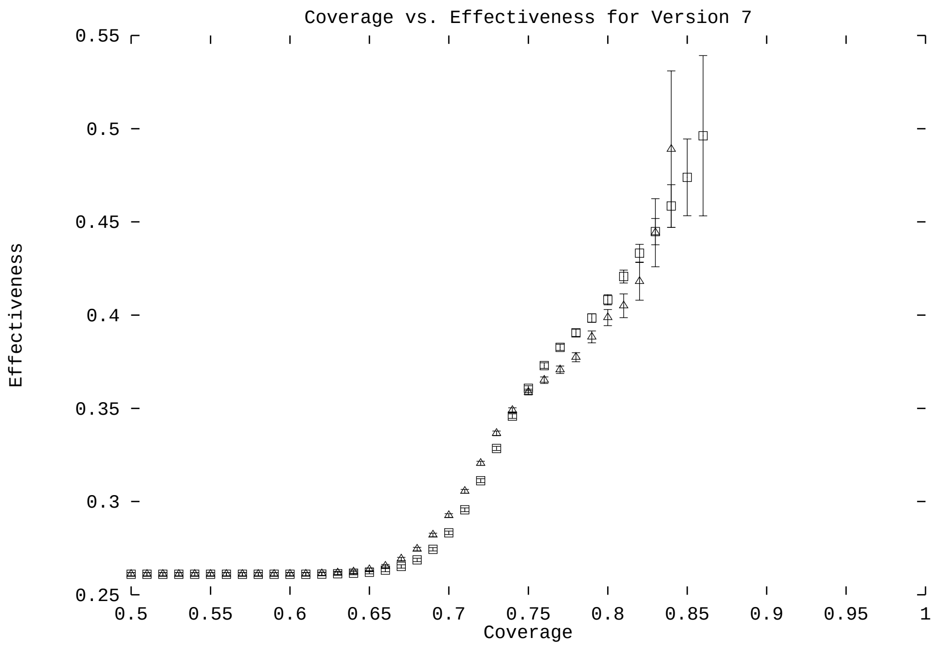

Figure 3: Coverage vs. effectiveness for Version 7. Size 20.

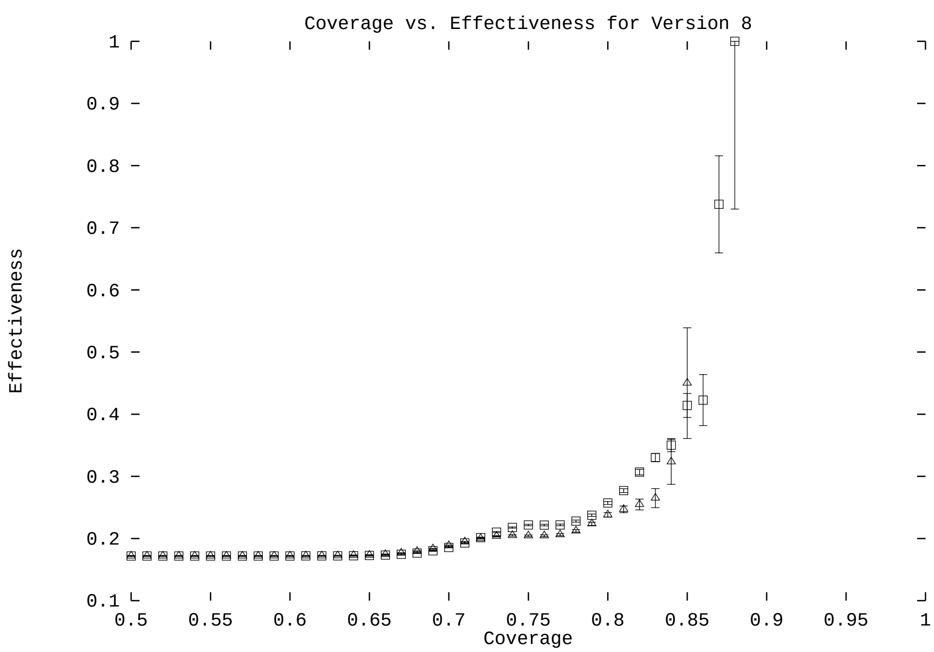

Figure 4: Coverage vs. effectiveness for Version 8. Size 20.

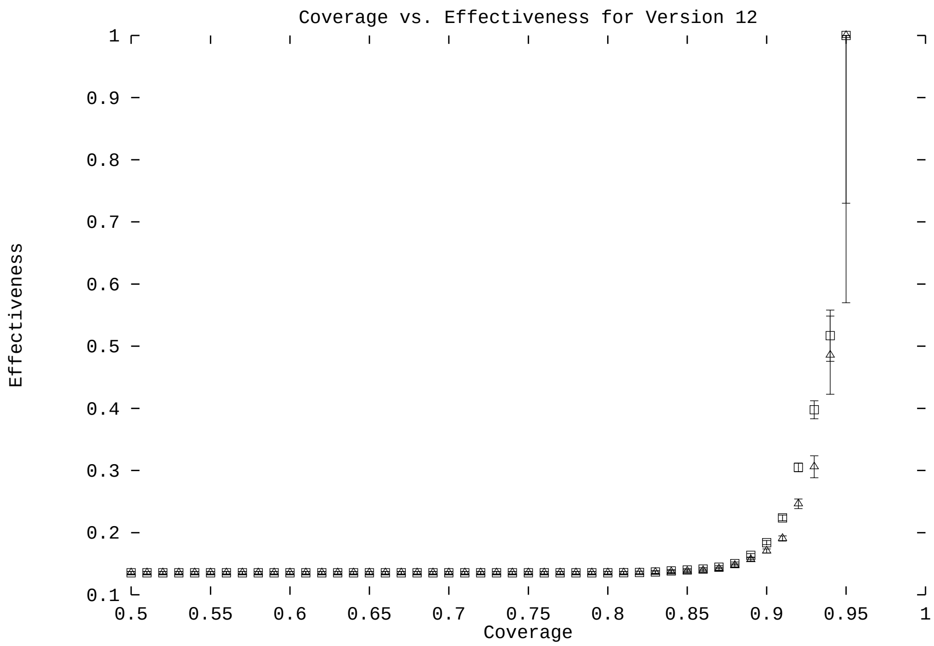

Figure 5: Coverage vs. effectiveness for Version 12. Size 100.

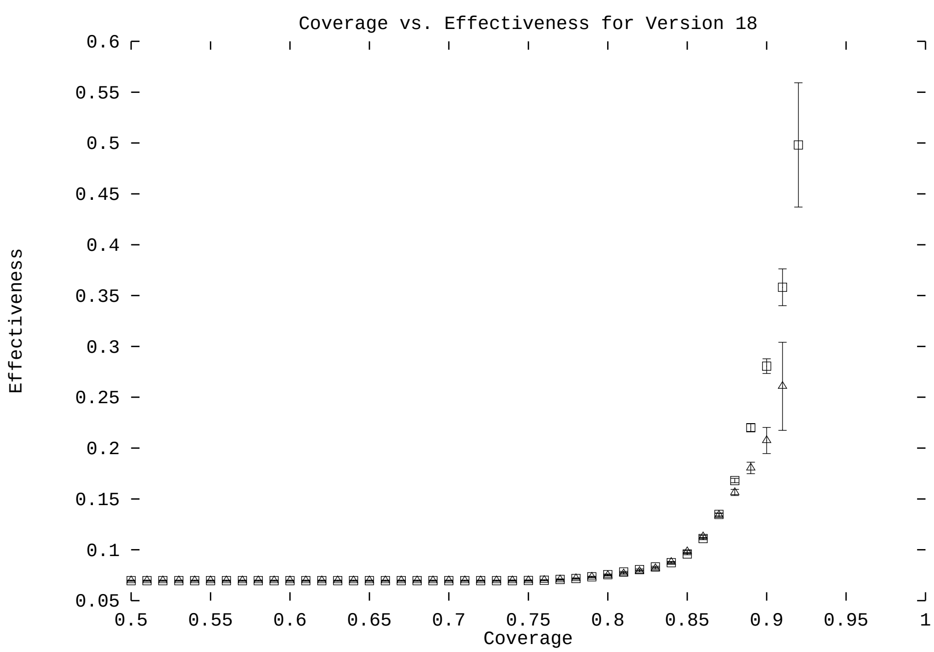

Figure 6: Coverage vs. effectiveness for Version 18. Size 50.

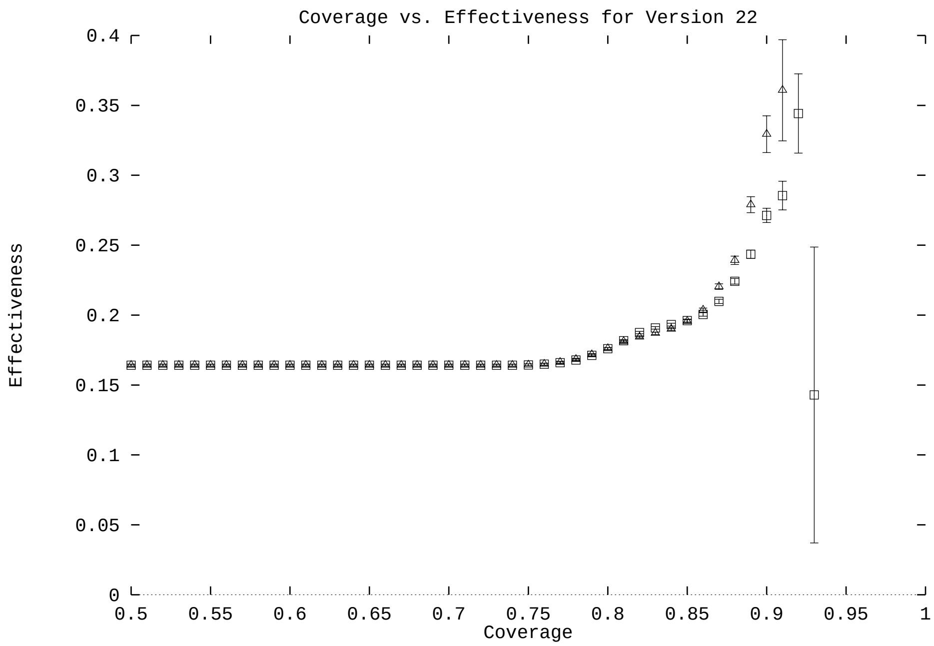

Figure 7: Coverage vs. effectiveness for Version 22. Size 50.

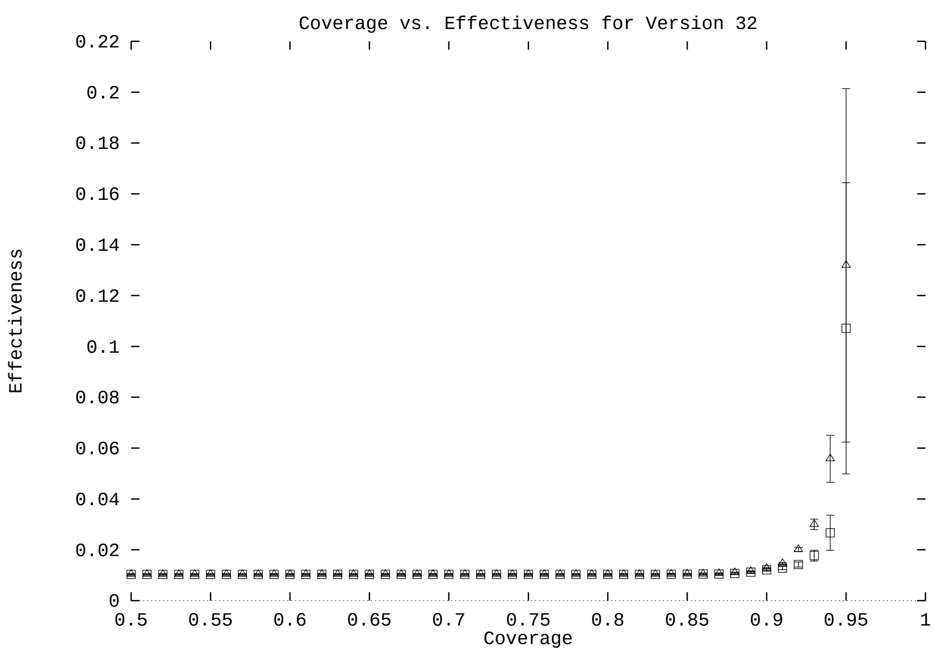

Figure 8: Coverage vs. effectiveness for Version 32. Size 100.

Surprisingly, the results of this experiment were more consistent than those of our earlier experiments on smaller subject programs [2, 3], in which all-uses and mutation testing sometimes performed very well, but sometimes performed poorly. This consistency may be due to the fact that the subject programs were very similar to one another (although in general, different faulty versions of the same underlying program could yield very different results.) Unlike the earlier experiment results, effectiveness appeared to increase monotonically as coverage increased for all of the subjects (except for the one outlying point discussed above). As in the earlier experiments, there were a few subject programs in which high coverage levels appear to guarantee detection of the fault; for many other subjects, the effectiveness at the highest coverage levels considered was many times greater than that of random test sets (with no adequacy criterion), but still rather low numerically.

Hutchins *et al.* compared the effectiveness of branch testing to all-DU testing (a data flow testing criterion that is similar, but not identical, to all-uses) and random testing without an adequacy criterion, using moderately small (141 to 512 LOC) C programs with seeded faults as subjects. As with our study, numerous test sets were generated for each subject, although the details of how they were generated and the statistical techniques used to analyze the data were somewhat different. In 18 of their 106 subjects neither DU coverage nor branch coverage was significantly more effective than random testing. In contrast, all eight of our subjects showed both all-uses and branch coverage to be significantly more effective than random testing. This difference may be the result of the small differences in the details of the coverage criteria and the experiment design, or the substantial difference in the nature of the subject programs and the faults. However, we suspect that applying our experiment design to additional large subject programs will yield some in which neither all-uses nor decision testing is more effective than random testing.

Threats to Validity

There are several caveats that must be noted in interpreting these results.

- All eight subjects were drawn from the same development project; although they had different faults, the programs are extremely similar to one another. In fact, although they have different faults, three of the subjects all fail on precisely the same set of test cases from the universe.
- In all experiments of this type, the results are dependent on the specific universe of test cases used; if this universe is not representative of the entire input domain, the results can be biased. For example, it may be the case that the only test case in the universe that covers a particular branch b also exposes a fault, but that there are many test cases outside the universe that cover b but do not expose the fault. Then the experiment will erroneously indicate that 100% branch coverage is guaranteed to expose the fault. Similarly, inclusion of a single non-exposing test case covering b, when most of the test cases covering b *do* expose the fault, will bias the results in the other direction. Using a large universe, as done in this research, should somewhat alleviate this problem, but does not eliminate it.
- In addition, our notion of which decisions or duas are unexecutable was dictated by the universe. If branch b was deemed unexecutable but a high proportion of the test cases (outside the universe) covering b expose a fault, our failure to consider b will make branch coverage appear less effective than it actually is. Similar considerations apply, of course, for all-uses testing and for biases in the other direction. The proportion of decisions and duas that were unexecutable is somewhat higher than that reported in previous experiments [2], in which unexecutable requirements were analyzed by hand. This may be due to deficiencies in the universe, or may be due to the nature of the subject programs.
- For each subject program, we selected a fixed test set size; previous results have indicated that the general shape of the curves is similar for different test set sizes for the same subject program, but we have not investigated that issue carefully on these subjects. Preliminary results indicate that smaller test set sizes might yield more gradual increases in effectiveness as coverage increases.
- In earlier pilot studies, ATAC occasionally gave results that were different than those we obtained analyzing the program by hand; in these experiments, the notions of what constitutes a decision or dua and of when a test case covers a test requirement are the notions used in ATAC, which may occasionally differ from those of other data flow testing tools.
- In this experiment design, and others like it, one selects test sets randomly from a universe of test cases. This is done in order to make it practical to generate huge numbers of test sets and thereby to obtain a statistically significant number of test sets with high coverage levels. It is possible that the randomly generated test sets with coverage level c may have a different character than test sets generated by a human tester (or by a more sophisticated automatic test generation method) to achieve coverage level c. For example, it is possible that an automated testing technique that favors certain kinds of test cases covering a given requirement (such as those near the boundary), might produce more (or less) effective test sets than our random test sets. The experiment design used here could be adapted to study such questions in a statistically meaningful way. Similarly, it is possible to imagine that a very talented human might have a knack for, say, picking test cases that expose faults from among those test cases that cover a given decision or dua. Unfortunately, there is no proof that such individuals exist, much less an algorithm for finding them.

5 Conclusions

This paper presents the results of an experiment comparing the effectiveness all-uses data flow testing criterion to decision coverage (branch testing) and to random testing without any adequacy criterion. The subjects were 8 versions of a large C program, each of which had a fault which had occurred during the actual development process. The experiment was designed so as to control for test set size and rigorous statistical techniques were used to evaluate the data.

For all of the subject programs considered, test sets that attained a high level of decision coverage or dua coverage were significantly more likely to detect the fault than random test sets of the same size. However, in most subjects, even the high coverage level test sets were not terribly likely to detect the fault. These results are more promising than previous results (based on experiments with much smaller programs), but leave open the question of whether the benefits of using such coverage criteria out-weigh the costs.

These results represent only a small step toward answering the question of how effective these testing techniques are. There are several directions for continuing work, including further experimentation using other programs from the suite, further experimentation using different large programs, experiments using other measures of effectiveness, such as the failure rate after debugging [1] and experiments incorporating some means of measuring the cost of using a coverage criterion.

Acknowledgments

Bob Horgan's group at Bellcore developed ATAC and allowed us to use it. Alberto Pasquini and Paolo Matrella provided access to the the subject programs and provided the test generator. Hong Cui and Vera Peshchansky wrote some of the scripts used for building coverage matrices from ATAC output and performed pilot experiments. Cang Hu wrote the program used to simulate test set execution. Stewart Weiss was instrumental in developing the experiment design and the infrastructure to support it.

References

[1] P. G. Frankl, D. Hamlet, B. Littlewood, and L. Strigini. Choosing a testing method to deliver reliability. In *Proceedings International Conference on Software Engineering*, pages 68–78. IEEE Computer Society Press, May 1997.

[2] P. G. Frankl and S. N. Weiss. An experimental comparison of the effectiveness of branch testing and data flow testing. *IEEE Transactions on Software Engineering*, 19(8):774–787, Aug. 1993.

[3] P. G. Frankl, S. N. Weiss, and C. Hu. All-uses versus mutation: An experimental comparison of effectiveness. *Journal of Systems and Software*, (38):235–253, Sept. 1997.

[4] P. G. Frankl and E. J. Weyuker. An applicable family of data flow testing criteria. *IEEE Transactions on Software Engineering*, SE-14(10):1483–1498, Oct. 1988.

[5] P. G. Frankl and E. J. Weyuker. Provable improvements on branch testing. *IEEE Transactions on Software Engineering*, 19(10):962–975, Oct. 1993.

[6] J. Horgan and S. London. Data flow coverage and the C language. In *Proceedings Fourth Symposium on Software Testing, Analysis, and Verification*, pages 87–97. ACM Press, Oct. 1991.

[7] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand. Experiments on the effectiveness of dataflow- and controlflow-based test adequacy criteria. In *Proceedings 16th International Conference on Software Engineering*. IEEE Computer Society Press, May 1994.

[8] A. Mathur and W. E. Wong. An empirical comparison of mutation and data flow-based test adequacy criteria. Technical Report SERC-TR-135-P, Software Engineering Research Center, Purdue University, Mar. 1993.

[9] A. Pasquini, A. Crespo, and P. Matrella. Sensitivity of reliability-growth models to operational profiles errors vs testing accuracy. *IEEE Transactions on Reliability*, R-45(4):531–540, Dec. 1996.

[10] S. Rapps and E. J. Weyuker. Selecting software test data using data flow information. *IEEE Transactions on Software Engineering*, SE-14(4):367–375, Apr. 1985.

[11] B. Rosner. *Fundamentals of Biostatistics*. PWS-KENT, Boston, Mass., 1990.