On Comparisons of Random, Partition, and Proportional Partition Testing

Simeon C. Ntafos, *Member*, *IEEE*

Abstract—Early studies of random versus partition testing used the probability of detecting at least one failure as a measure of test effectiveness and indicated that partition testing is not significantly more effective than random testing. More recent studies have focused on proportional partition testing because a proportional allocation of the test cases (according to the probabilities of the subdomains) can guarantee that partition testing will perform at least as well as random testing. In this paper, we show that this goal for partition testing is not a worthwhile one. Guaranteeing that partition testing has at least as high a probability of detecting a failure comes at the expense of decreasing its relative advantage over random testing. We then discuss other problems with previous studies and show that failure to include important factors (cost, relative effectiveness) can lead to misleading results.

Index Terms—Program testing, random testing, partition testing, proportional partition testing.

挑选测试用例的原因

1 Introduction

THE main method that is used in order to reach some level of confidence on the reliability of software is program testing. Test cases are selected from the input space D of the program according to some testing strategy. The size of the test set is usually very small compared to the size of D. Herein lies the problem and challenge of software testing: How do we select the test cases so that we can be confident about the reliability of the program from a very small sample? Most testing strategies use some form of information about the program or its specification to generate the test cases. This information may be the control or data flow of the program [1], functional properties of the program [17], [20], information on commonly occurring failures, or combinations thereof (e.g., mutation analysis [9]).

In the partition testing model, we consider a partition of D into disjoint subdomains $D_1, D_2, \ldots, D_k$. It is assumed that the partition is done to reflect a particular testing strategy. A subdomain could consist of inputs that cause a certain branch to be executed, or inputs that are "equivalent" with respect to some property (equivalence partitioning [20]), or even inputs that are likely to detect a certain type of fault. This allows partition testing to model a variety of strategies.

While a partition requires disjoint subdomains, most testing strategies do not yield disjoint subdomains in a natural fashion. Strategies like equivalence partitioning [20] can be adapted to produce disjoint subdomains (by intersecting the true partitions that are obtained with respect to each criterion). However, the subdomains consisting of inputs that result in the execution of specific branches may well overlap. The issue of overlapping subdomains is discussed in [15], [28]. While true partitions are probably unlikely in practice, for our purposes, we will follow previous studies and assume that we have a true partition of D.

The partition testing model associates a probability p_i and a failure rate ϑ_i with each subdomain. These are assumed to be uniform within each subdomain, i.e., all inputs are equally likely to be used and equally likely to result in failure. These are simplifying assumptions that may not hold in practice. Partition testing requires coverage of all the subdomains, i.e., at least one test case must be selected from each subdomain.

In random testing, test cases are selected randomly from the entire input domain of the program. Random testing can be viewed as a statistical sampling experiment and statistical analysis can yield reliability estimates from the test outcomes. In order for the estimates to be applicable to real-life use of the software, the test cases need to mirror the expected use of the software in the field (the operational profile). Random testing according to the operational profile is sometimes called statistical or operational testing and has been successfully employed in practice (e.g., in the Cleanroom development methodology [7]).

The ideal situation for partition testing would be to produce homogeneous (or revealing) subdomains [15], [29], i.e., subdomains in which all inputs either result in a failure or produce correct results. Then, if any partition test set (consisting of just one test case per subdomain) does not detect any failures, we can conclude that the program is correct. Although this ideal is hard to reach in realistic situations, it may be one of the reasons behind the practice of selecting one test case per subdomain for many testing strategies.

Partition testing may be viewed as stratified random sampling while random testing corresponds to simple random sampling from the input domain D [8], [19], [22]. If some level of homogeneity is achieved, it is a well-known result in statistics that stratified random sampling can lead to precise estimates using a smaller sample than simple random sampling [8]. This would imply that partition

• *The author is with the University of Texas at Dallas, Computer Science Program, Richardson, TX 75083. E-mail: ntafos@utdallas.edu.*

Manuscript received 13 July 1999; accepted 16 Oct. 2000.
Recommended for acceptance by D. Hamlet.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 110224.

testing should perform better than random testing with an equal number of test cases.

Comparisons of random and partition testing go back to a simulation study by Duran and Ntafos [10]. They used a failure rate distribution that approximated homogeneity by forcing most failure rates to be close to 0 or 1. They compared the probabilities of detecting at least one failure for random and partition testing. These are given by the following expressions:

$$P_r = 1 - (1 - \vartheta)^n,$$

$$P_p = 1 - \Pi_i (1 - \vartheta_i)^{n_i},$$

where n_i is the number of test cases in subdomain D_i, ϑ_i is the failure rate in subdomain D_i, n is the total number of test cases ($n = \Sigma_i n_i$), and ϑ is the failure rate of the program ($\vartheta = \Sigma_i p_i \vartheta_i$).

The study in [10] found that the performance of random testing was pretty close to that of partition testing. A comparison of the expected number of failures detected by the two strategies also produced similar results. The results in [10] suggested that random testing could be more cost-effective than partition testing (assuming that its cost is lower than that of partition testing). This conclusion went against the expectation that strategies based on analysis of the structure and properties of a program (knowledge-based strategies) should be more effective than random sampling.

Hamlet and Taylor followed up with a more extensive simulation study that further investigated the various parameters of the model and obtained similar results [15]. Weuyker and Jeng [28] did an analytical study which produced conditions that make one strategy perform better than the other. One of their observations was that, if all subdomains have equal size and equal numbers of test cases are selected from each subdomain, partition testing has an equal or better chance of detecting at least one failure. This condition was generalized in [3], where it was shown that partition testing will do at least as well as random testing as long as the allocation of the test cases to the subdomains is done in proportion to the size of the subdomains (proportional partition testing). Other conditions have also been published but proportional allocation has received the most attention [2], [3], [4], [5], [6].

In this paper, we take a critical look at some of the comparisons of random versus partition testing in [2], [3], [4], [5], [6], [10], [15]. Our summary (above) of the main results from these studies is necessarily limited, (see [2], [3], [4], [5], [6], [10], [15] for more details). In Section 2, we discuss proportional partition testing and show that the very reason it was put forth does not make much sense. In Section 3, we discuss some other difficulties with the previous studies and show that some very important factors that were disregarded can easily change the results.

2 Is Proportional Partition Testing Best?

The novelty of proportional partition testing stems from an analytical result in [3] showing that it is guaranteed to perform at least as well as random testing with respect to the probability of detecting at least one failure. In this section, we show that guaranteeing that $P_p \geq P_r$ comes at a steep price makes pursuing this goal unreasonable.

Proportional partition testing allocates n test cases to the k subdomains according to the probability of each subdomain (or its size if one assumes that each input is equally likely to occur). As previously observed in [6], [28], a true proportional allocation may be infeasible or unrealistic because of the large number of test cases that may be needed. For example, consider a program in which D consists of two subdomains. Suppose that one of them corresponds to a rare special condition, which occurs once in a million runs. Then, true proportional partition testing would require a total of 1,000,001 test cases to test a program that could consist of a single IF statement. Note that rare conditions are present in many (if not all) large systems and should not be ignored as they are likely sources of failure [16].

Since a true proportional allocation may not be feasible, proportional partition testing can only try to approximate a true proportional allocation as closely as possible. Note that random testing will also tend to allocate test cases according to the probabilities of the subdomains. If random sampling from the entire domain is repeated many times, the average allocation to each subdomain will be proportional to the probability of that subdomain. With a larger number of test cases, the two allocations will tend to become more similar. In some sense, proportional partition testing is a version of partition testing that imitates random testing. While this may assure that $P_p \geq P_r$, it actually reduces the relative advantage that partition testing has over random testing by forcing the "stronger" strategy to imitate the "weaker" one.

It is easy to see why trying to guarantee that partition testing does not do worse than random testing comes at a price and is counterproductive. Suppose that we have k subdomains and we start with $n = k$ test cases. Then, partition testing allocates one test case per subdomain. This is the case discussed in [10], [15]. If the experiment is repeated many times, partition testing does better than random testing more often than not. Proportional partition testing is also forced to allocate one test case per subdomain although this is very unlikely to be close to the true proportional allocation. To allow a reasonably good approximate proportional allocation, we need to increase the number of test cases. As n increases, proportional partition testing can get closer and closer to achieving a true proportional allocation (in terms of percentage deviations). The number of cases in which proportional partition testing does worse than random testing should decrease and eventually become zero. But, at the same time, the probability of detecting at least one failure increases for both random and proportional partition testing. As a result, the number of cases in which the two strategies perform equally well should increase. Also, the difference between the probabilities of detecting at least one failure for the two strategies will decrease as n increases (the marginal contribution of each additional test case gets smaller and smaller due to the exponential form of the expressions).

The price that proportional partition testing pays in order to make sure that random testing does not outperform it is the decrease in the size of the relative advantage that it has over random testing. An analysis of this effect involves the probabilities p_i (which are available if an operational profile has been developed) and the failure rates ϑ_i (which are not known). Rather than assume specific probability and failure rate distributions, we performed a simulation study where the general form of these distributions could be controlled and each experiment could be repeated many times.

To illustrate the simulation setup, consider a program with $k = 20$ subdomains and let the number of test cases range from $n = 20$ to $n = 800$ in increments of 20. For each (k, n) pair, we generated probabilities and failure rates for the subdomains and test case allocations for the various strategies. Each such experiment was repeated 1,000 times. For each trial run, random values in the range (0,1] were assigned to the probabilities p_i and normalized. The failure rates ϑ_i were assigned randomly (uniform distribution) in the range (0,0.1] resulting in an expected overall failure rate of 0.05.

Since a true proportional allocation is usually not feasible with a limited number of test cases, we tried three different approximations, prop1, prop2, and prop3. All three approximations start with an initial allocation of $(\max(1, \text{round}\ n * p_i))$. This initial allocation may need to be adjusted because the sum of the allocations may not be equal to the specified value of n test cases. The prop1 allocation makes adjustments to the initial allocation by minimizing the resulting percentage deviation from the true allocation each time. The prop2 allocation does the same within two groups of subdomains (those above, below the true proportional allocation); by adjusting within each group before moving on to the other one, the actual size of the deviations from the true allocation is kept small. The prop3 allocation makes adjustments in the order of minimum net effect on the percentage deviation from the true proportional allocation.

Table 1 shows the number of times (out of 1,000) that random testing resulted in a probability of detecting at least one failure that was larger than, equal to, or smaller than the corresponding probability for the three versions of proportional partition testing. The last column shows the ratio $(P_p - P_r)/P_r$ for the prop3 allocation (the numbers for the other two are almost identical). Fig. 1 shows the number of trials (out of 1,000) with $P_r > P_p$ and with $P_r \geq P_p$ as functions of n (P_p is computed using the prop3 allocation). Both curves start with random testing performing better than partition testing in about 50 percent of the trials (this is in agreement with the results in [10], [15]). Both curves quickly drop to very small values as the increasing number of test cases allows reasonably close approximations to a true proportional allocation (agreeing with [3]). But then, the two curves diverge as the number of trials with $P_r = P_p$ becomes the dominant factor in the comparison; the number of trials with $P_r \geq P_p$ levels off as the number of test cases grows past 800 and the upper bound of 1,000 trials is approached. Note that looking at $P_r \geq P_p$ (rather than $P_r > P_p$ as is done in [2], [3], [4], [5], [6]) is the more realistic view of the comparison between random and partition testing since $P_r = P_p$ is likely to be viewed as a favorable outcome for random testing (assuming it is cheaper than partition testing). While random testing does well at the two ends of the "U-shaped" curve for $P_r \geq P_p$, the precipitous drop in the middle seems like a strong argument in favor of proportional partition testing. However, this is just a misleading partial view and this becomes obvious when one looks at the variation of the ratio $(P_p - P_r)/P_r$ with n (not shown) whose shape is very similar to that of the number of trials with $P_r > P_p$. While the number of wins for random testing decreases with increasingly proportional allocations, there is a matching decrease in the relative difference between the performance of the two strategies.

Besides the number of test cases, the number of subdomains (k) as well as the probability, and failure rate distributions all may affect the comparison between random and partition testing. To determine those effects, we repeated the simulations for various values of k and a variety of failure rate and probability distributions. The remainder of this section discusses these simulations.

Table 2 shows the results for $k = 10, 50,$ and 100 with the failure rates still uniformly distributed in the range (0,0.1]. The prop3 allocation is shown for proportional partition testing. Since prop3 usually performs slightly better than the other two approximations, it will be used as the standard proportional testing allocation from now on. The results in Table 2 are similar to those in Table 1. There is a minor delay in the decay of the number of trials with $P_r > P_p$ as k increases. This is to be expected since, with higher k, there are more subdomains with relatively high failure rates to which proportional partition testing allocates less than the ideal number of test cases.

Next, we consider the effect that the failure rate distribution has on the comparison of random versus proportional partition testing. The expected overall failure rate in the simulations so far was 5 percent which is probably too high for many applications. We repeated the simulations with failure rates in the ranges (0,0.01], (0,0.001], and (0,0.0001]. Sample results are shown in Table 3. As the failure rates get smaller, it becomes harder and harder to detect at least one failure and the number of trials with $P_r > P_p$ lingers around 50 percent longer (as n increases). Other than the shift towards higher numbers of test cases as the failure rates decrease, the form of the variations stays pretty much the same. Again, we have the expected pattern of a decrease in the number of trials with $P_r > P_p$ and in the ratio $(P_p - P_r)/P_r$ as n increases; the number of trials with $P_r \geq P_p$ follows an unbalanced "U" starting around 50 percent, dipping down towards 0 percent, and ending around 100 percent.

Fig. 2 shows the number of trials with $P_r \geq P_p$ for $k = 50$ and 100, failure rates in (0,0.01] (top) and failure rates in (0,0.001] (bottom). Again, the results follow the same patterns. The number of trials with $P_r > P_p$ is the same on the decreasing side but then remains very low as in Fig. 1. The last observed nonzero entries for the number of trials with $P_r > P_p$ occur at $n = 5,000$ for $k = 50$, θ_i in (0,0.01]; at $n = 6,000$ for $k = 100$, θ_i in (0,0.01]; at $n = 52,000$ for $k = 50$, θ_i in (0,0.001]; and at $n = 58,000$ for $k = 100$, θ_i in (0,0.001]. With the number of test cases starting at 500, the ratio $(P_p - P_r)/P_r$ starts with very small values (below 0.05 percent) and quickly drops to insignificant levels.

TABLE 1
Random vs. Proportional Partition Testing ($k = 20$)

n	random vs. prop1	random vs. prop2	Random vs. prop3	$(P_p-P_r)/P_r$
20	475-0-525	475-0-525	475-0-525	0.00531262
40	426-0-574	404-0-596	396-0-604	0.00262833
60	337-0-663	299-0-701	306-0-694	0.00174396
80	280-0-720	255-0-745	251-0-749	0.00064685
100	218-0-782	179-0-821	179-0-821	0.00034222
120	166-0-834	145-0-855	141-0-859	0.00017810
140	138-0-862	104-0-896	93-0-907	0.00008087
160	124-0-876	72-0-928	76-0-924	0.00003753
180	78-0-922	46-0-954	45-0-955	0.00001809
200	60-0-940	31-0-969	30-0-970	0.00001034
220	36-0-964	19-0-981	21-0-979	0.00000410
240	35-0-965	14-0-986	14-0-986	0.00000251
260	27-0-973	15-0-985	7-0-993	0.00000129
280	24-0-976	4-0-996	5-0-995	0.00000046
300	14-0-986	3-0-997	2-0-998	0.00000027
320	17-0-983	4-0-996	1-0-999	0.00000014
340	9-0-991	1-0-999	1-0-999	0.00000007
360	11-0-989	2-0-998	0-0-1000	0.00000005
380	7-0-993	2-0-998	2-0-998	0.00000003
400	3-0-997	0-0-1000	0-0-1000	0.00000001
420	2-0-998	0-0-1000	0-0-1000	-
440	5-0-995	0-0-1000	0-0-1000	0.00000001
460	2-2-996	0-2-998	0-1-999	-
480	1-5-994	0-5-995	0-5-995	-
500	2-3-995	0-3-997	0-3-997	-
520	1-8-991	0-7-993	0-7-993	-
540	0-29-971	0-30-970	0-28-972	-
560	0-50-950	0-52-948	0-49-951	-
580	0-81-919	0-83-917	0-81-919	-
600	0-135-865	0-136-864	0-136-864	-
620	0-184-816	0-183-817	0-184-816	-
640	0-231-769	0-230-770	0-230-770	-
660	1-329-670	0-326-674	0-322-678	-
680	0-417-583	0-417-583	0-417-583	-
700	0-478-522	0-480-520	0-479-521	-
720	0-517-483	0-518-482	0-517-483	-
740	0-614-386	0-611-389	0-612-388	-
760	0-689-311	0-687-313	0-693-307	-
780	0-698-302	0-696-304	0-689-311	-
800	0-771-229	0-771-229	0-772-228	-

A failure rate distribution that has received attention in previous studies is one that clusters most of the failure rates close to 0 or 1 to approximate homogeneous subdomains. Note that the distributions used so far may also be viewed as approximating homogeneous subdomains but with all the failure rates close to 0. Table 4 shows sample results for several bimodal distributions. The first (Table 4a) is a failure rate distribution in which 95 percent of the θ_i are 0, while the remaining 5 percent are 1. With $k = 20$ subdomains, this results in one homogeneous faulty subdomain. Thus, it is not surprising that proportional partition testing does significantly better than random testing. However, the critical factor in this comparison is not the proportional allocation but the homogeneity of the subdomains. A comparison of random versus uniform partition testing (allocate the test cases evenly to the subdomains) yields the same results. The results shown in Table 4b are for $k = 20$ and for two failure rate distributions that have 95 percent (90 percent) of the failure rates in (0,0.025] and the remaining 5 percent (10 percent) uniformly distributed in (0.030,1], respectively. Now, with perfectly homogeneous subdomains very unlikely, the results again follow the standard patterns. Note that, with both distributions, proportional partition testing does not always do better than random testing even with 1,000 test cases. The results shown in Table 4c are for $(k = 50)$ and three distributions with 98 percent, 96 percent, and 92 percent of the failure rates in (0,0.025], respectively, while the rest of the failure rates are in (0.03,1]. Again, we have the same patterns, with the difference in performance quickly becoming negligible before proportional partition testing can provide any assurance that it always does better than random testing.

Since uniform probability distributions are unlikely in practice, we tried distributions in which a certain percentage of the subdomains had lower probabilities. There were some shifts but the patterns did not change. We also tried keeping the probabilities and failure rates the same while allowing the number of test cases to change. Again, there

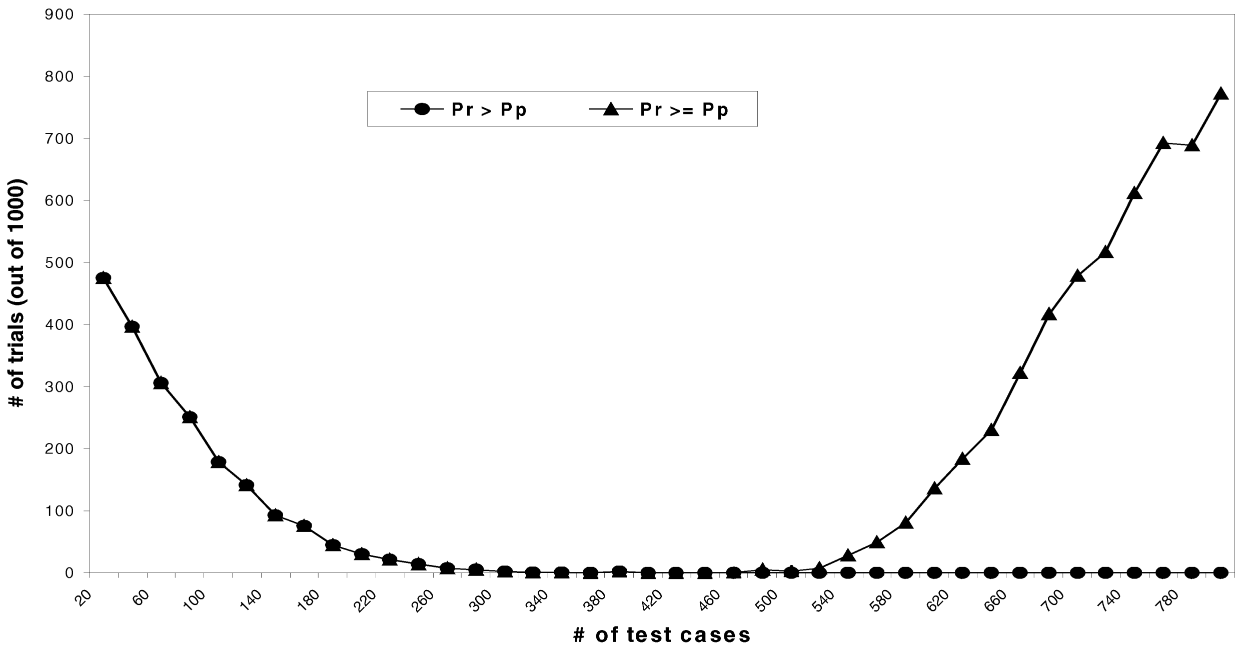


Fig.1. Number of trials with $P_r > P_p$ and with $P_r \geq P_p$ ($k = 20, \theta_i$ in (0,0.1]).

was no significant change. A more interesting issue arises from the fact that the proportional allocations are only approximate. This could raise questions as to what approximation should be used. If the failure rates are known, one can optimize the proportional allocation by maximizing the resulting probability of detecting at least one failure. Since failure rates for the subdomains are unlikely to be known in practice, we considered probability distributions that would allow a true proportional allocation with a limited number of test cases. We did this by forcing the probabilities to be multiples of 0.01. Then, a true proportional allocation results for values of n that are multiples of 100 but the allocations are only approximately proportional for other values of n. Table 5 shows the results for $k = 20$ and $k = 50$, θ_i in (0,0.1]. Note that the number of trials with $P_r \geq P_p$ drops to zero for $n = 100, 200, 300, ,$ and 400. This is in agreement with [3] (we can not have $P_r > P_p$ when a true proportional allocation occurs and there are no trials with $P_r = P_p$ for $n = 100, 200, 300, ,$ and 400.) When $n = 500, 600, \ldots,$ we still do not have any trials with $P_r > P_p$, but, now, the number of trials with $P_r = P_p$ steadily increases. When n is not a multiple of 100, a true

TABLE 2
Random vs. Proportional Partition Testing for θ_i in (0,0.1] and $k = 10, 50,$ and 100

	k = 10		k = 50		k = 100	
n	rand– prop3	$(P_p-P_r)/P_r$	rand– prop3	$(P_p-P_r)/P_r$	rand– prop3	$(P_p-P_r)/P_r$
20	421-0-579	0.00591205				
60	255-0-745	0.00132654	401-0-599	0.00174452		
100	115-0-885	0.00043601	303-0-697	0.00038120	380-0-620	0.00045566
140	49-0-951	0.00012725	221-0-779	0.00007139	332-0-668	0.00006241
180	17-0-983	0.00003620	171-0-829	0.00001255	243-0-757	0.00001179
220	10-0-990	0.00001142	92-0-908	0.00000226	206-0-794	0.00000190
260	3-0-997	0.00000393	56-0-944	0.00000044	140-0-860	0.00000031
300	2-0-998	0.00000098	24-0-976	0.00000008	108-0-892	0.00000005
340	2-0-998	0.00000054	24-0-976	0.00000001	70-0-930	0.00000001
380	0-1-999	0.00000036	9-0-991	-	49-0-951	-
420	0-0-1000	0.00000013	4-0-996	-	35-0-965	-
460	0-11-989	0.00000010	2-0-998	-	30-0-970	-
500	0-36-964	0.00000005	0-0-1000	-	8-0-992	-
540	0-96-904	0.00000001	0-0-1000	-	10-0-990	-
580	0-158-842	0.00000001	0-19-981	-	5-3-992	-
620	0-279-721	-	0-80-920	-	1-31-968	-
660	0-378-622	-	0-243-757	-	1-184-815	-
700	0-485-515	-	0-439-661	-	0-433-567	-
740	0-588-412	-	0-668-332	-	0-704-296	-
780	0-627-373	-	0-815-185	-	0-880-120	-

TABLE 3
Random vs. Proportional Partition Testing for $k = 20$ and θ_i in (0,0.1], (0,0.001], and (0,0.0001]

θ_i in (0,0.01]			θ_i in (0,0.001]			θ_i in (0,0.0001]		
n	rand– pro3	$(P_p-P_r)/P_r$	n	rand– pro3	$(P_p-P_r)/P_r$	n	rand– pro3	$(P_p-P_r)/P_r$
20	522-0-478	-0.0011	20	525-0-475	-0.00207	20	525-0-475	-0.00217
100	464-0-536	0.0009	1000	457-0-543	0.00008	1000	493-0-507	0.00003
400	340-0-660	0.0003	2000	436-0-564	0.00006	4000	464-0-536	0.00002
1000	157-0-483	0.0000	4000	333-0-667	0.00003	10000	479-0-521	0.00001
2000	175-0-825	-	10000	171-0-829	0.00000	40000	354-0-646	0.00000
2000	24-0-976	-	20000	28-0-972	-	100000	167-0-833	-
3000	2-0-998	-	30000	1-0-999	-	300000	5-0-995	-
4000	0-0-1000	-	40000	1-0-999	-	500000	0-147-853	-

proportional allocation is not achieved and the number of trials with $P_r \geq P_p$ follows the standard "U-shape." Table 5 illustrates the sensitivity of the comparison to the quality of the proportional approximation; even small deviations from a true proportional allocation can significantly impact the ability of proportional partition testing to assure that no cases where $P_r > P_p$ occur. The more important comparison in terms of the ratio $(P_p - P_r)/P_r$ still follows the standard pattern (even when n is a multiple of 100). We note that the probability distribution used for Table 5 may well be more representative of real operational profiles (in that the precision of probability estimates in a realistic operational profile may be limited). This may make it easier to generate true proportional allocations. However, this does not make proportional partition testing more attractive. In fact, if operational profiles are only approximate in practice, the sensitivity of proportional partition testing to the quality of the approximation (when it comes to guaranteeing that $P_p \geq P_r$) may make this goal impossible to achieve in practice. Just because the best available profile may allow true proportional allocations, there is no reason to expect that the underlying actual profile will also do so.

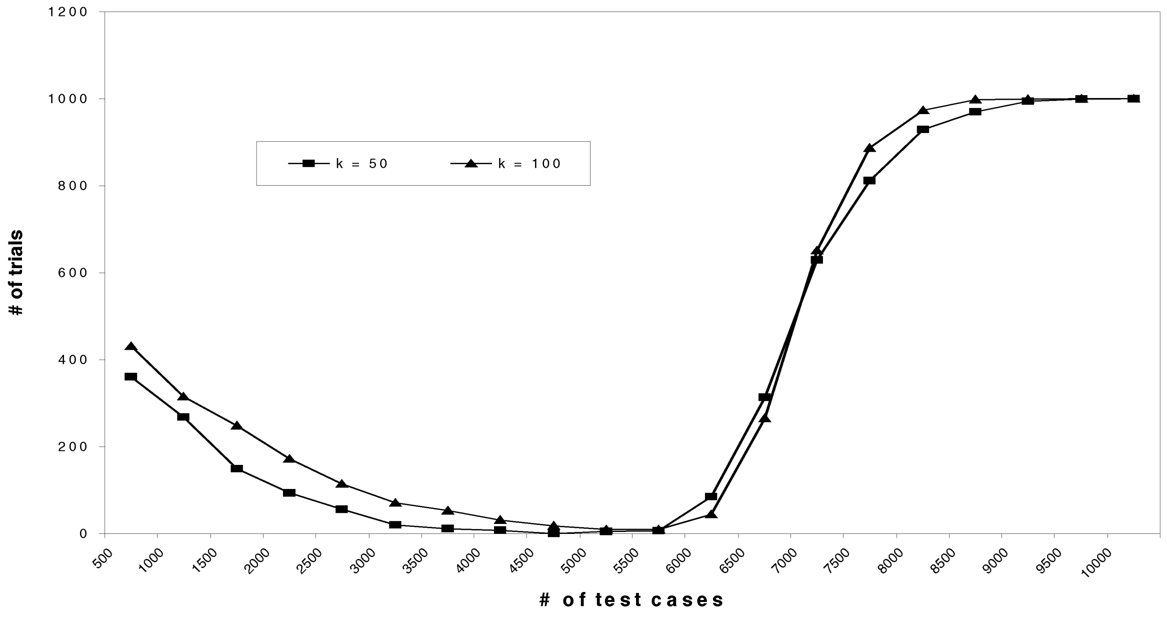
We have shown that the goal of guaranteeing that $P_p \geq P_r$ (the main attraction of proportional partition testing) will usually require a large number of test cases; this, in turn, reduces the difference in effectiveness between partition testing and random testing. Since the difference in effectiveness is relatively small to start with, this makes the goal not worth pursuing and leaves proportional partition testing with little to recommend it. Note that the difference in performance between random and partition testing can increase significantly when other factors (e.g., cost [26]) are considered.

3 The Cost and Relative Effectiveness Factors

The basic question in the random versus partition testing debate remains largely unresolved despite the many studies that have been published. Many testers would still expect knowledge-based testing strategies to outperform random testing despite what the comparisons in [10], [15] seem to indicate. One reason for this may be that research has focused on the partition testing model rather than actual knowledge-based strategies and the practical benefits of the "knowledge" used have been disregarded. Different testing strategies may be more effective at various stages/levels of testing (e.g., unit versus system test) since the failure rates and homogeneity of subdomains will keep changing as testing progresses. Since most practical testing strategies do not naturally induce true partitions of the input domain, it may well be that the failure rates of certain subdomains are not independent of each other; then, knowledge-based strategies may be more advantageous [18]. Also, most studies of random versus partition testing have disregarded important factors such as cost and relative effectiveness. In this section, we discuss these factors and show how easily they can alter the outcome of a comparison between random and partition testing.

Cost is clearly a central factor in any realistic comparison but it is hard to measure, data are not easy to obtain, and little has been done to deal with it. The cost of testing should not only include the cost of preparing test plans, executing the test plans and buying/developing tools to support the process but it should also include the cost of failures that remain undetected and will eventually manifest themselves as field failures. An effort to include failure cost factors was reported in [26]. This was an analytical study that added a cost factor c_i to each subdomain in the partition testing model (reflecting the cost of a failure in subdomain D_i) and considered the goal of minimizing the sum $\Sigma_i c_i p_i \vartheta_i$. It found that including costs in the model tends to increase the difference in performance between partition and random testing. It was also shown in [26] that the optimal way (with respect to minimizing the sum $\Sigma_i c_i p_i \vartheta_i$) to allocate the test cases is in proportion to the product $c_i p_i$. Note that the proportional allocation scheme turns out to be optimal under this measure as well (i.e., consider the special case where all the cost factors are equal). Some other efforts that take the cost of failures into account are reported in [14], [21], [23], [24], [25], [27].

Besides cost, another factor that has been disregarded in comparisons of random and partition testing is the relative effectiveness of a test case under the two strategies. The partition testing model assumes that test case selection within a subdomain is done at random. For many real knowledge-based testing strategies, it is often the case that better test cases can be generated. For example, functional testing considers the functions that the program performs and finds test inputs that are important to those functions. Domain analysis [30] checks for domain faults using a



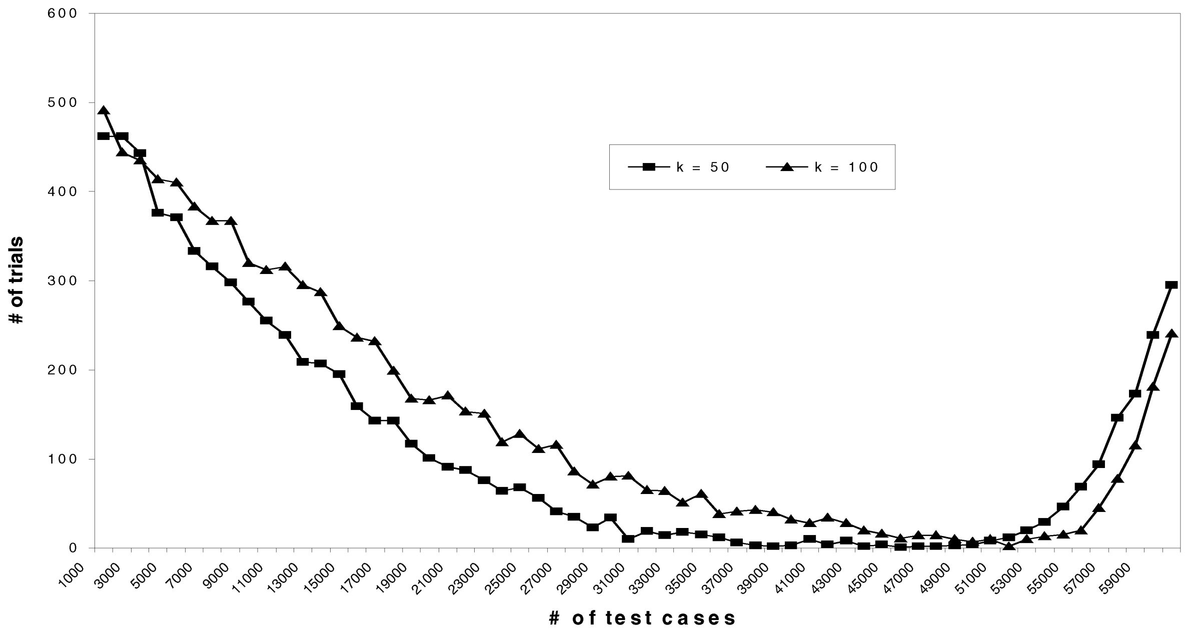


Fig. 2. Number of trials with $P_r \geq P_p$ as a function of n, θ_i in (0,0.01] (top), θ_i in (0,0.001] (bottom).

number of test cases that is related to the dimensionality of the boundaries and can assure detection of certain fault types. Fault-based strategies will target specific types of faults and are more likely to find them than randomly selected test cases (in fact, the fault types themselves could be used to determine the subdomains). All this points out that the best way to do partition testing should be determined by the actual strategy that is used to create the partition. Selecting one test case per subdomain essentially implies a reliance on homogeneity. While homogeneity is the ideal for the partition testing model, it is very hard to achieve in practice. Homogeneity is achieved in strategies like exhaustive testing but there is no realistic strategy that has been shown to induce homogeneous subdomains. Then, selecting one case per subdomain is just the least expensive way to perform partition testing (in that it does the minimum required coverage).

Cost and relative effectiveness can both be injected into the partition testing model in a similar fashion by translating them into some equivalent number of additional test cases. Suppose that random testing is no more expensive than partition testing and that a partition testing

TABLE 4a
Comparison of Random Testing with Proportional Partition Testing and Uniform Partition Testing for a Homogeneous Distribution $(k = 20)$

	Homogeneous distribution		Homogeneous distribution	
n	ran– pro3	$(P_p-P_r)/P_r$	ran– uniform	$(P_p-P_r)/P_r$
20	0-0-1000	0.7816	0-0-1000	0.7816
40	0-0-1000	0.2966	0-0-1000	0.2966
60	0-0-1000	0.2009	0-0-1000	0.2008
100	0-0-1000	0.0947	0-0-1000	0.0947
200	0-0-1000	0.0468	0-0-1000	0.0468
300	0-3-997	0.0254	0-3-997	0.0254
400	0-96-904	0.0219	0-96-904	0.0219
600	0-383-617	0.0160	0-383-617	0.0160
800	0-583-417	0.0100	0-583-417	0.0100

TABLE 4b
Comparisons of Random, Proportional Partition Testing for Two Nonhomogenous Failure Rate Distributions $(k = 20)$

k=20	θ_i: 95% in (0,.025], rest >.03%		θ_i: 90% in (0,.025], rest >.03%	
n	ran– pro3	$(P_n-P_r)/P_r$	ran– pro3	$(P_n-P_r)/P_r$
20	244-0-756	0.2706	155-0-845	0.2132
40	128-0-872	0.1047	32-0-968	0.0712
60	97-0-903	0.0599	26-0-974	0.0326
80	78-0-922	0.0338	15-0-985	0.0149
100	59-0-941	0.0192	10-0-990	0.0089
200	20-0-980	0.0031	0-1-999	0.0007
300	18-0-982	0.0006	1-47-952	0.0001
400	13-26-961	0.0001	0-176-824	
500	9-78-913	-	0-349-651	-
600	6-170-824	-	0-497-503	-
700	4-232-765	-	0-579-421	-
800	3-315-682	-	0-668-332	-
900	1-381-618	-	0-732-268	-
1000	3-436-561	-	1-754-245	-

TABLE 4c
Comparisons of Random, Proportional Partition Testing for Three Nonhomogeneous Failure Rate Distributions $(k = 50)$

k=50	98% of θ_i in (0,.025], 2% of θ_i in (0.03,1]		96% of θ_i in (0,.025], 4% of θ_i in (0.03,1]		92% of θ_i in (0,.025], 8% of θ_i in (0.03,1]	
n	ran– pro3	$(P_n-P_r)/P_r$	ran– pro3	$(P_n-P_r)/P_r$	Ran– pro3	$(P_n-P_r)/P_r$
50	262-0-738	0.1385	138-0-862	0.1251	54-0-946	0.0674
100	113-0-887	0.0424	26-0-974	0.0337	5-0-995	0.0101
200	65-0-935	0.0068	15-0-985	0.0035	1-0-999	0.0005
300	57-0-943	0.0014	12-0-988	0.0006	0-1-999	0.0001
400	26-0-974	0.0003	3-0-997	0.0001	0-28-972	-
500	16-0-984	0.0001	1-5-994	-	0-141-859	-
600	11-0-989	-	3-35-960	-	0-315-685	-
700	16-1-983	-	1-80-919	-	0-496-504	-
800	8-14-978	-	0-153-847	-	0-615-385	-
900	7-47-946	-	1-247-752	-	0-688-312	-
1000	3-79-915	-	0-354-646	-	0-834-166	-

test set is at least as effective as a random test set. While this may be the more common view, one can argue the opposite as well. For example, the cost of the test "oracle" (deciding if a test outcome is correct) may be much higher for random testing in some situations which may make random testing more expensive than some partition testing strategy that relies on tests with known outcomes. A random test set may be more effective than a partition test set if the partition test set duplicates checks considered during code inspections while random tests exercise the code in novel ways. For example, this is one of the justifications for the sole use of statistical testing in Cleanroom [7]. Since our purpose is only to show how important the cost and relative effectiveness factors are, we will assume that the cost (effectiveness) of a partition test case is at least as high as the cost (effectiveness) of a random test case.

Suppose that we have determined that random testing for a certain application, development environment, etc.,

TABLE 5
Comparison with Probability Distributions that Allow Perfect Proportional Allocations

	k=20, θ_i in (0,0.1]		k = 20, θ_i in (0,0.01]	
n	trials with $P_r \geq P_p$	$(P_p-P_r)/P_r$	trials with $P_r \geq P_p$	$(P_p-P_r)/P_r$
20	418	0.00837859	467	0.00378602
40	365	0.00318607	478	0.00086987
60	267	0.00155702	466	0.00092310
80	220	0.00069668	470	0.00060711
100	0	0.00032172	0	0.00061003
120	106	0.00015786	453	0.00069956
140	93	0.00006537	462	0.00038710
160	59	0.00003955	449	0.00062255
180	45	0.00001692	432	0.00042813
200	0	0.00000892	0	0.00046795
220	17	0.00000387	415	0.00034391
240	18	0.00000163	409	0.00037358
260	8	0.00000137	419	0.00038686
280	5	0.00000087	377	0.00043953
300	0	0.00000021	0	0.00034289
320	0	0.00000014	398	0.00026598
340	3	0.00000007	379	0.00023844
360	1	0.00000005	382	0.00029195
380	1	0.00000004	384	0.00030147
400	0	0.00000003	0	0.00025977
420	0	-	360	0.00019579
440	0	-	322	0.00021382
460	0	-	347	0.00022795
480	2	-	349	0.00020496
500	6	-	0	0.00018449
520	7	-	289	0.00015600
540	29	-	296	0.00015299
560	64	-	301	0.00017280
580	72	-	268	0.00017094
600	131	-	0	0.00013546
620	205	-	267	0.00010055
640	241	-	276	0.00009926
660	307	-	273	0.00012201
680	406	-	246	0.00010897
700	446	-	0	0.00009598
720	500	-	239	0.00008343
740	606	-	264	0.00007009
760	702	-	212	0.00008714
780	682	-	244	0.00007955
800	755	-	0	0.00006834

is m times cheaper than some partition testing scheme. If the cost is proportional to the number of test cases used, that would imply that we can run m times more random test cases at the same cost. This assumes that the cost of testing is proportional to the number of test cases used (i.e., $\text{cost} = m * n$). Since there may be cost components (e.g., setup, tools) that are independent of the number of test cases, $\text{cost} = a + m * n$ may be more appropriate. Actually, it is not really known if the functional form should be linear. Also, if we include the cost of undetected faults, another term needs to be added. We are not proposing any particular functional form for the relation of testing cost to the number of test cases or claim that one form is more realistic than another. The point is that, regardless of the functional form that the cost of testing takes, as long as it is measurable, one can translate the difference in cost between random and partition testing into a number of additional random tests that could be performed at the same cost.

The relative effectiveness factor is harder to quantify but we can take a parallel view as follows: One way to do partition testing is to obtain the subdomains according to some testing strategy and then do random sampling within each subdomain. Regardless of the number of test cases used per subdomain, we could view this as throwing darts on a target (the faulty inputs) blindfolded. An alternative view of partition testing is that the use of "knowledge" does not stop with the creation of the partition but continues with the test case selection process. We could view this as throwing darts without the blindfold. Although the target may not be clear (we do not know where the faults are) or the tester's aim may be questionable, we would expect that the "aiming" tester will cluster more hits near the target than the "blindfolded" tester. We can then take the areas

TABLE 6
Comparing $m * n$ Random Test Cases vs. n Partition Test Cases

	Set 1		Set 2		Set 3		Set 4		Set 5	
m	$P_r >$	% diff	$P_r >$	% diff	$P_r >$	% diff	$P_r >$	% diff	$P_r >$	% diff
1.00	475	-0.53	525	0.21	93	-11.85	6	-2.65	1	-38.27
1.05	710	2.23	752	5.19	140	-10.33	20	-2.13	2	-36.71
1.10	870	4.85	887	10.17	178	-8.94	50	-1.68	3	-35.22
1.15	955	7.33	961	15.15	215	-7.65	100	-1.29	4	-33.81
1.20	978	9.69	984	20.12	248	-6.48	160	-0.96	5	-32.48
1.25	988	11.94	993	25.10	284	-5.39	232	-0.69	10	-31.21
1.30	997	14.07	997	30.07	325	-4.39	327	-0.42	18	-30.00
1.35	999	16.09	999	35.04	368	-3.47	411	-0.20	26	-28.85
1.40	999	18.02	999	40.00	414	-2.62	472	-0.01	39	-27.76
1.45	999	19.85	999	44.97	440	-1.84	537	0.15	47	-26.72
1.50	999	21.58	999	49.93	485	-1.11	602	0.29	66	-25.73
1.55	999	23.24	1000	54.88	521	-0.04	644	0.41	77	-24.79
1.60	1000	24.80	1000	59.84	554	0.17	693	0.52	89	-23.88
1.65	1000	26.29	1000	64.79	591	0.75	730	0.61	101	-23.02
1.70	1000	27.72	1000	69.74	631	1.27	763	0.70	119	-22.20
1.75	1000	29.06	1000	74.69	657	1.76	782	0.77	144	-21.41
1.80	1000	30.34	1000	79.63	683	2.21	805	0.83	156	-20.66
1.85	1000	31.56	1000	84.58	702	2.63	828	0.89	172	-19.94
1.90	1000	32.71	1000	89.52	725	3.02	848	0.94	187	-19.25
1.95	1000	33.82	1000	94.45	752	3.38	863	0.98	201	-18.59
2.00	1000	34.86	1000	99.39	780	3.72	879	1.02	215	-17.96

that the blindfolded person and the aiming tester hit with frequency over some threshold and use the relative size of them as a measure of relative effectiveness. Another way to look at this would be to determine what number of randomly selected test cases would result in the same expected value for the probability of hitting the target.

We can now look at the cost and relative effectiveness factors in terms of additional test cases. Again, we do this only for the purpose of illustrating how important these factors can be and how easily they can alter the conclusions one draws. To this end, let us assume that random testing can use $m * n$ test cases at the same cost as n partition test cases. How large does m need to be in order to make up for the advantage of partition testing over random testing noted in previous studies?

Table 6 shows sample comparisons between random and proportional partition testing with the number of random test cases ranging from n to $2n$, while the number of partition test cases is kept at n. The first set of results gives the number of trials (out of 1,000) with $P_r > P_p$ and the percentage difference between the two (computed as $100 * (P_r - P_p)/P_p$) for $k = 20$, $n = 20$, and the ϑ_i uniformly distributed in (0,0.1]. Just one additional random test case suffices to tilt the balance in favor of random testing. A few more random test cases result in clear dominance for random testing. Unlike the results in Table 1, where dominance for proportional partition testing (in terms of the number of trials with $P_r > P_p$) was accompanied by a decline in the difference between the two to insignificant levels, the increase in the number of trials with $P_r > P_p$ is accompanied by solid gains in the percentage difference between the two strategies. With 40 test cases ($m = 2$), the probability of detecting at least one failure is about 30 percent higher for random testing. If the base number of test cases is increased, it becomes even easier for random testing to overtake proportional partition testing. This holds as long as the probabilities of detecting one failure do not get very close to 100 percent, in which case the additional random test cases are of little value. Increasing the number of subdomains again makes it easier for random testing to overtake proportional partition testing. The same is true of we consider smaller failure rates. The results shown as Set 1 in Table 6 are the least favorable for random testing among the 12 combinations of $k = 20, 50, ,$ and 100, and upper bounds on the ϑ_i at 0.1, 0.01, 0.001, and 0.0001 (excluding cases where saturation occurs). One more of these 12 combinations, the case with $k = 20$, $n = 20$, and the ϑ_i uniformly distributed in (0,0.001], is shown as Set 2 in Table 6.

The remainder of Table 6 shows cases that are favorable to partition testing. For Sets 3, 4 we have $k = 20$, 95 percent of the ϑ_i are in (0,0.1] and the remaining 5 percent are in (0.6-1.0], and $n = 20$ (Set 3), $n = 40$ (Set 4). The failure rate distribution is approximately homogeneous and that is reflected in the results. We now need 50 percent more random test cases (in this case 30 rather than 20) for random testing to break even with proportional partition testing. If we start with 40 test cases, then random testing breaks even at 40 percent. Set 5 represents an even more favorable setup for partition testing. We have $k = 20$, 95 percent of the ϑ_i are in (0,0.1] and the remaining 5 percent are in (0.9-1.0]. The built-in advantage for partition testing is now too much to overcome, but doubling the number of random test cases, does reduce the percentage difference between the two strategies in half. Overall, unless some level of homogeneity is achieved, a small percentage increase in the number of random test cases suffices for random testing to overtake proportional partition testing.

Clearly, relative effectiveness can be an equally important factor. Assuming that a partition testing strategy is

m times more effective than random sampling within subdomains translates into m times more test cases per subdomain. Then, comparisons like those in Table 6 would quickly turn the small advantage that partition testing with one test case per subdomain enjoys over random testing into clear dominance for partition testing. If the assumption that partition testing is at least as effective as random sampling while random testing is at most as expensive as partition testing holds, we note that the effects of the cost and relative effectiveness factors will counter each other. Since the difference between random and partition testing is usually small when the cost and relative effectiveness factors are not included, it follows that the central issue in a comparison of random versus partition testing when these factors are included will be which of them has the most effect. In particular, the relative effectiveness factor may be a likely explanation for the expectation that partition testing should outperform a random test set of equal size.

4 Concluding Remarks

Guaranteeing that $P_p \geq P_r$ is the main reason for all the attention proportional partition testing has received. The conclusion we draw from the simulations presented in Section 2 is that this is not a worthwhile goal to pursue. Thus, proportional partition testing should not be viewed as "the" way to do partition testing. Partition testing is relatively more effective if one test case per subdomain is used and reaching the goal of always having $P_p \geq P_r$ may well require an inordinate number of test cases or may even be impossible if the available operational profile is not very accurate. Furthermore, making sure that $P_p \geq P_r$ usually means that the two strategies become nearly indistinguishable. If cost effectiveness is considered (as it usually is in practice), the situation becomes clearly untenable because of the large number of test cases that may be needed to assure that $P_p \geq P_r$.

Partition testing strategies are expected to perform better than random testing; the real issue has always been cost-effectiveness. The results in Section 3 illustrate how easy it is to tilt the balance in favor of random testing with a small number of additional test cases. Relative effectiveness is an equally important factor that will usually favor partition testing strategies. These factors need to be studied by considering specific testing strategies and using data from real projects. They are much too important to ignore and ignoring them comes at the risk of obtaining misleading and/or meaningless results.

Note that very low probability subdomains are likely to be disregarded by random testing. While that may not affect the operational reliability much, it could have a large impact if the cost of failures in such low probability subdomains is very high. This is a major flaw of random testing and a good reason for using it as a complementary rather than the main/only testing strategy. Note that the use of statistical testing in the Cleanroom methodology, where no other "testing" is done, is actually as a complementary check. The Cleanroom approach relies heavily on inspections and other early lifecycle methods to eliminate faults. Its proponents argue that standard testing strategies would mostly duplicate checks that have already been performed while statistical testing may provide new checks and yields operational reliability estimates.

Another potential problem with many studies of random versus partition testing is the nearly exclusive use of the probability of detecting at least one failure as a measure of test effectiveness. The expected number of failures detected has been shown to be highly correlated to the probability of detecting at least one failure in the partition model [4]. However, test effectiveness can be measured in many ways. A comparison in terms of the delivered reliability is reported in [11]. Cost should be a central factor in realistic comparisons of testing strategies. An alternative measure of testing effectiveness, the expected cost of field failures, was introduced in [21] as the sum $\Sigma_i c_i (1 - \vartheta_i)^{n_i}$, where c_i is the cost of a failure in D_i. Under this measure, the operational profile is not important and uniform partition testing outperforms proportional partition testing [21]. While the expected cost of field failures may not be useful as a practical metric, the probability of detecting at least one failure is not a generally used metric either. Comparisons that use practical metrics and involve actual testing strategies (e.g., [12], [13]) are needed to better illuminate the random versus partition testing question.

Acknowledgments

This material is based in part upon work supported by the Texas Advanced Technology Program under Grant No. 009741-021. A preliminary version of some of this material appears in the Proceedings of ISSTA-98, SIFSOFT News, vol. 23, no. 2, pp. 42-48, Mar. 1998.

References

[1] B. Beizer, *Software Testing Techniques,* second ed. New York: Van Nostrand, 1990.
[2] F. Chan, T. Chen, I. Mak, and Y. Yu, "Proportional Sampling Strategy: Guidelines for Software Test Practitioners," *Information and Software Technology,* vol. 38, no. 12, pp. 775-782, 1996.
[3] T. Chen and Y. Yu, "On the Relationship Between Partition and Random Testing," *IEEE Trans. Software Eng.,* vol. 20, no. 12, pp. 977-980, Dec. 1994.
[4] T. Chen and Y. Yu, "On the Expected Number of Failures Detected by Subdomain Testing and Random Testing," *IEEE Trans. Software Eng.,* vol. 22, no. 2, pp. 109-119, Feb. 1996.
[5] T. Chen and Y. Yu, "A More General Sufficient Condition for Partition Testing to be Better than Random Testing," *Information Processing Letters,* vol. 57, no. 3, pp. 145-149, 1996.
[6] T. Chen and Y. Yu, "Constraints for Safe Partition Testing Strategies," *Computer J.,* vol. 39, no. 7, pp. 619-625, 1996.
[7] R. Cobb and H. Mills, "Engineering Software Under Statistical Quality Control," *IEEE Software,* pp. 44-54, Nov. 1990.
[8] W. Cochran, *Sampling Techniques.* Wiley, 1977.
[9] R. DeMillo, R.R. Lipton, and F. Sayward, "Hints on Test Data Selection: Help for the Practicing Programmer," *IEEE Computer,* vol. 11, no. 4, pp. 34-41, 1978.
[10] J. Duran and S. Ntafos, "An Evaluation of Random Testing," *IEEE Trans. Software Eng.,* vol. 10, no. 4, pp. 438-444, July 1984.
[11] P. Frankl, D. Hamlet, B. Littlewood, and L. Strigini, "Choosing a Testing Method to Deliver Reliability," *Proc. 19th Int'l Conf. Software Eng.,* pp. 68-78, May 1997.
[12] P. Frankl and O. Iakounenko, "Further Empirical Studies of Test Effectiveness," *Proc. ACM SIGSOFT Sixth Symp. Foundations of Software Eng., SIGSOFT Software Eng. Notes,* vol. 23, no. 6, pp. 153-162, Nov. 1998.
[13] P. Frankl, S. Weiss, and C. Hu, "All-Uses versus Mutation Testing: An Experimental Comparison of Effectiveness," *J. Systems and Software,* vol. 38, no. 3, pp. 235-253, Sept. 1997.

[14] W. Gutjahr, "Optimal Test Distributions for Software Failure Cost Estimation," *IEEE Trans. Software Eng.,* vol. 21, no. 3, pp. 219-228, Mar. 1995.

[15] R. Hamlet and R. Taylor, "Partition Testing does not Inspire Confidence," *IEEE Trans. Software Eng.,* vol. 16, no. 12, pp. 1402-1411, Dec. 1990.

[16] H. Hecht, "Rare Conditions—An Important Cause of Failures," *Proc. Ann. Conf. Computer Assurance, (COMPASS-93),* pp. 81-85, 1993.

[17] W. Howden, "A Functional Approach to Program Testing and Analysis," *IEEE Trans. Software Eng.,* vol. 12, pp. 997-1005, 1986.

[18] B. Jeng, "Toward an Integration of Data Flow and Domain Testing," *J. Systems and Software,* vol. 45, no. 1, pp. 19-30, Feb. 1999.

[19] B. Mitchell and S. Zeil, "An Experiment in Estimating Reliability Growth Under Both Representative and Directed Testing," *Proc. ISSTA-98 ACM Software Eng. Notes,* vol. 23, no. 2, pp. 32-41, Mar. 1998.

[20] G.J. Myers, *The Art of Software Testing.* New York: John Wiley and Sons, 1979.

[21] S. Ntafos, "The Cost of Software Failures," *Proc. Int'l Assoc. of Science and Technology for Development, Int'l Conf. Software Eng.,* pp. 53-57, Nov. 1997.

[22] A. Pudgurski and C. Yang, "Partition Testing, Stratified Sampling and Cluster Analysis," *Proc. SIGSOFT Symp. Foundations of Software Eng.,* pp. 169-181, 1993.

[23] D. Rosenbloom and E. Weyuker, "Using Coverage Information to Predict the Cost Effectiveness of Regression Testing Strategies," *IEEE Trans. Software Eng,* vol. 23, no. 3, pp. 146-156, Mar. 1997.

[24] S. Sherer, "Measuring Software Failure Risk: Methodology and an Example," *J. Systems and Software,* vol. 25, pp. 257-269, 1994.

[25] S. Sherer, "A Cost-Effective Approach to Testing," *IEEE Software,* pp. 34-40, 1991.

[26] M. Tsoukalas, M.J. Duran, and S. Ntafos, "On Some Reliability Estimation Problems in Random and Partition Testing," *IEEE Trans. Software Eng.,* vol. 19, no. 7, pp. 687-697, July 1993.

[27] E. Weyuker, "Using Failure Cost Information for Testing and Reliability Assessment," *ACM Trans. Software Eng. and Methodology,* vol. 5, no. 2, pp. 87-98, Apr. 1996.

[28] E. Weyuker and B. Jeng, "Analyzing Partition Testing Strategies," *IEEE Trans. Software Eng.,* vol. 17, no. 7, pp. 703-711, July 1991.

[29] E. Weyuker and T. Ostrand, "Theories of Program Testing and the Application of Revealing Subdomains," *IEEE Trans. Software Eng.,* vol. 6, pp. 236-246, 1980.

[30] L. White and E. Cohen, "A Domain Strategy for Computer Program Testing," *IEEE Trans. Software Eng.,* vol. 6, pp. 247-257, 1980.

Simeon C. Ntafos received the BS degree in electrical engineering from Wilkes College, Wilkes-Barre, Pennsylvania, in 1974, and the MS degree in electrical engineering and PhD degree in computer science from Northwestern University, Evanston, Illinois, in 1976 and 1979, respectively. He was a visiting assistant professor in the Department of Electrical Engineering and Computer Science at Northwestern University during 1978-1979. He joined the faculty of the University of Texas at Dallas in 1979 as an assistant professor in the Department of Mathematical Sciences and is now a professor in the Department of Computer Science. He was the head of the Computer Science Program at University of Texas at Dallas from 1985-1987 and has been the associate chair of the Department of Computer Science since the Fall of 1998. His current research interests include software testing, software reliability, and computational geometry. He is a member of the IEEE.