

Testing and the Cost of Field Failures

Simeon Ntafos (ntafos@utdallas.edu)

Dept. of Computer Science, The University of Texas at Dallas

1. Introduction

Software reliability is the probability that a program will operate correctly for a specified time in a specified environment. What we may really be interested in is the total cost of faults that remain in the released software and will eventually manifest themselves as failures in the field. We present an approach to estimate failure costs in terms of the input/output behavior of the program and use the estimates to better allocate testing effort so as to reduce the expected cost of field failures. Simulation studies using the partition testing model indicate that substantial benefits may be attainable even if failure rates/costs can only be roughly approximated. The expected cost of field failures can also be combined with measures of the cost of carrying out more testing to lead to better software release decisions.

2. The Partition Testing Model

Partition testing models strategies in which the input domain is partitioned into subdomains and test cases are selected from each subdomain. Let D be the input domain and consider a partition of D into disjoint subdomains D_1, \dots, D_k . The model associates a probability p_i and a failure rate ϑ_i with each subdomain. The partition testing model has been used to compare partition strategies with random testing. Early studies [1,2] used the probability of detecting at least one failure as a measure of effectiveness and found little difference between the performance of random and partition testing (indicating that random testing may be more cost-effective). The model was extended in [6] by introducing costs (c_i is the cost incurred if the program fails to execute correctly on an input from D_i). The introduction of variable cost failures tends to increase the gap in effectiveness between partition and random testing. Relative effectiveness is another crucial factor that was ignored in the early studies and may explain the perceived advantage of partition testing [5].

The expected cost of field failures was introduced in [4] as a measure that focuses on the impact of failures. This is given by the expression

$\sum_i (1-\vartheta_i)^{n_i} * c_i$, where n_i is the number of test cases from D_i . This assumes that all undetected faults will eventually result in a field failure. It is true that some faults will never manifest themselves as field failures within the period the software is operational. We could disregard the cost of subdomains that have a very low probability of execution or cost factors below a certain threshold. However, it may well be advisable to take a worst case view of things, i.e., assume that every undetected fault will result in a failure. Note that the expected cost of field failures is a measure that depends on the amount of testing and on the distribution of test cases to subdomains. Then one can reduce the expected cost of field failures by doing more testing and/or by allocating a specified number of test cases better. While the probabilities may be available from an operational profile, the failure rates and failure costs within each subdomain are generally not

known a priori. Then, is it feasible to develop testing strategies that can be effective with respect to the expected cost of field failures?

3. Using Expected Failure Costs in Testing

To determine the feasibility of using failure costs in determining testing methodology, allocation of testing effort, release time, we performed simulation studies based on the partition testing model. If the ϑ_i, c_i are known, we can optimally allocate any number of test cases so as to minimize the expected cost of undetected failures. The question we try to answer is what benefits are attainable if we make educated guesses for the failure and cost profiles and how that compares to the alternative of treating all failures as having equal cost (i.e., the current state of affairs).

Several strategies based on approximate failure rate/cost profiles were introduced in [4]. They include: the "average-theta" strategy (costs are known but the ϑ_i are approximated by the expected failure rate of the program); the "two-theta" strategy (failure rates are correctly classified into high/low groups and all values in each group are approximated by the group's mean); the "two-cost" strategy (uses the average ϑ in place of the ϑ_i and classifies costs into two groups represented by their means); the "two-two" strategy (combination of "two-theta" and "two-cost"). We compared the expected cost of field failures for these strategies with that for random testing, uniform partition testing (equal n_i) and proportional partition testing (n_i proportional to p_i). Simulation results [4,5] clearly demonstrate the enormous potential that a testing methodology based on reasonably good estimates of the cost and failure profiles would have over the current practice of mostly disregarding the issue.

Among the findings from the simulation studies is the importance of the failure rate estimates. This is especially the case when a significant number of subdomains may have zero failure rates. The "two-cost" and "two-two" strategies will replace the zero failure rates with a significantly higher value. This, coupled with subdomains that have high costs, can result in a high allocation of test cases to subdomains where they produce no benefit. We introduced the "three-theta" (classify failure rates into three groups and use the group means as estimates) and the "three-two" (combine three-theta with two-cost) strategies to allow for better modeling of near homogeneous failure rate distributions like those used in [1,2]. This reduces, but does not eliminate, the wasted allocation of test cases to high-cost subdomains that have zero failure rates. Further insight is obtained by looking at best and worst case scenarios for the various strategies. The best case scenarios have the estimates used in place of the c_i, ϑ_i be the mean values of normal distributions with very small variances. Then, the new strategies perform at near optimum levels. The advantage over random (and proportional partition testing) is maximized when the high costs and high failure rates are concentrated

on subdomains with very low probabilities. The advantage over uniform partition testing is maximized when there are very few subdomains with high cost and high failure rates while the rest of the subdomains have low costs and failure rates. Note that the operational profile is not important when the expected cost of field failures is used while it is very significant to random and proportional partition testing. It is not surprising then that uniform partition testing outperforms random and proportional partition testing. Because of the diminishing returns from each additional test case, the new strategies are vulnerable to concentrating very large numbers of test cases on high cost subdomains. Worst case scenarios result either if the high cost estimate is wrong or the failure rate estimate is relatively high (while $\theta_1 = 0$).

Failure data analysis indicates that fine-grain cost estimates like the ones we used in the simulations in [4] may be unrealistic in practice (e.g., costs ranging from \$100 to \$10,000 in increments of \$100). Often costs can only be assigned in terms of orders of magnitude. Based on this, we performed simulation experiments using costs that were powers of two. For example, cost factors from the set {\$128, \$256, ..., \$32k} cover a wider range with only 9 distinct values. The two-cost strategy uses mid values of the exponents (\$512 and \$8k). The results show that the benefits of including cost factors are even more pronounced. Note that such costs may be viewed as severity classes and offer a compromise between the very few severity classes that are suggested in the reliability growth modeling literature (which may not provide enough distinction) and the exact costs (which may be hard to obtain).

4. Estimation of Cost/Failure Profiles

A central issue is the development of realistic approximate cost and failure profiles. One approach is to extend operational profiles to get an initial cost profile. Musa has made a suggestion along these lines in [3] where he recommends developing distinct operational profiles for operations with different criticality. Operational profiles are needed to estimate the probabilities p_i so that they will reflect the actual distribution of inputs in the field. A detailed procedure for developing an operational profile is described in [3]. It entails collecting information from the users as to how they plan to use the software. The users may also be best suited to estimate what the loss of a certain function will cost. This would allow the estimation of cost factors for subdomains that correspond to program functions. A second approach is to use the results of early testing to set up cost factors and failure rate profiles for system testing. A third approach is to analyze real programs with documented failures experienced in the field. Although documentation rarely includes cost data (especially from the user's side), it usually includes a description of the problem and the man-hours required to fix it. Fairly accurate cost estimates can be made from that information. The cost/failure rate profiles thus

obtained may be useful in developing empirical estimates for similar software under development and getting insight into the distribution of cost factors and failure rates.

Combining all three approaches may be the more promising choice. Historical data can be used to set up initial models which can then be refined as cost profiles are developed in conjunction with operational profiles. Then, the results of inspections and unit testing can be used to validate and further refine the models.

5. Concluding Remarks

The cost of testing should be a central factor in realistic comparisons of testing strategies. The total cost consists of the cost of carrying out the testing itself but should also include the cost of undetected errors that will eventually show up as field failures. The decision to release the software can be viewed as a tradeoff between the cost of additional testing and the cost of more field failures. We discussed the expected cost of field failures as a measure of test effectiveness and as a potential way to improve testing by considering approximate cost and failure rate profiles. Simulation results show the potential gains that can be realized. What is needed next is to validate the approach by employing it in realistic pilot projects.

Acknowledgment

This material is based in part upon work supported by the Texas Advanced Technology Program under Grant No. 009741-021.

References

- 1 Duran, J. and S. Ntafos, "An Evaluation of Random Testing," IEEE Trans. on Software Engineering, Vol. SE-10, No. 4, July 1984, pp. 438-444.
- 2 Hamlet, R. and R. Taylor "Partition Testing does not Inspire Confidence", IEEE Trans. on Software Eng.g, Vol. 16, No. 12, Dec. 1990, p. 1402-1411.
- 3 Musa, J. "Operational Profiles in Software Reliability Engineering," IEEE Software, pp. 14-32, March 1993.
- 4 Ntafos, S., "The Cost of Software Failures", Proc. of IASTED Int. Conf. on Software Engineering, pp. 53-57, Nov. 1997.
- 5 Ntafos, S., "On Random and Partition Testing", Proc. ISSTA-98 in ACM SIGSOFT Software Engineering Notes, Vol. 23, No. 2, pp. 42-48, March 1998.
- 6 Tsoukalas, M., J. Duran and S. Ntafos, "On Some Reliability Estimation Problems in Random and Partition Testing", IEEE Trans. on Software Engineering, Vol. 19, No. 7, pp. 687-697, July 1993.