Combining operational and debug testing for improving reliability

Domenico Cotroneo, *Member, IEEE,* Roberto Pietrantuono, *Member, IEEE,* Stefano Russo *Member, IEEE*

Index Terms

Operational testing, statistical testing, debug testing, reliability testing.

Abstract

This paper addresses the challenge of *reliability-driven* testing, i.e., of testing software systems with the specific objective of increasing its operational reliability. We first examined the most relevant approach oriented toward this goal, namely *operational testing*. The main issues that in the past hindered its wide-scale adoption and practical application are first discussed, followed by the analysis of its performance under different conditions and configurations. Then, a new approach conceived to overcome the limits of operational testing in delivering high reliability is proposed. The two testing strategies are evaluated probabilistically, and by simulation. Results report on the performance of operational testing when several involved parameters are taken into account, and on the effectiveness of the new proposed approach in achieving better reliability. At a higher level, the findings of the paper also suggest that a different view of the *testing for reliability improvement* concept may help to devise new testing approaches for high-reliability, demanding systems.

ACRONYMS

OP	Operational
OP-D	Operational-Debug
ADL	Array Definition Language

NOTATION

D	Input domain
F	Failure domain
$\|D\|$	Cardinality of D
$\|F\|$	Cardinality of F
θ	Operational failure probability
H	Probability distribution over D describing the operational profile
G	Probability distribution over D describing the testing profile of a generic testing technique
$\hat{H}$	Probability distribution over D describing the testing profile of operational testing
V	Probability distribution over D describing the testing profile of debug testing
Q	Probability distribution over F describing the operational occurrence of failure regions
T	Total number of test cases
T_1	Number of test cases for operational testing
T_2	Number of test cases for debug testing

D. Cotroneo, R. Pietrantuono, and S. Russo are with the Department of Electrical Engineering and Information Technology, University of Naples "Federico II", Via Claudio 21, 80125 Naples, Italy. E-mail: {cotroneo, roberto.pietrantuono, Stefano.Russo}@unina.it

I. Introduction

Software reliability and software testing are two research areas strictly connected to each other. Reliability is expected to improve through testing, and the consequent debugging. *Software reliability growth models* (SRGM) have been proposed in the literature to relate results of testing activities to the expected growth of software reliability [1]. These models address the question on *how to measure and predict reliability improvement as testing proceeds*, and in the scope of such objective they are known to give good results, even when data partly violates the model's assumptions they are based on [2]. Nonetheless, the opposite question has not received adequate attention in the past: *how to test software to measurably improve reliability?* The testing community addressed only partially this challenge: it is still not clear what technique is better to use for assuring high reliability.

There are roughly two ways to perform testing: *debug* testing, aimed at exposing as many failures as possible, and *operational* testing, aimed at evaluating the software with respect to the achieved reliability. In the former case, testers want to expose failures, from which to find bugs. In the latter approach, testers aim at achieving high operational reliability, intended as the probability of not failing during the operational phase. There is an inherent difference between them because the rate of failure occurrence during actual operation comes into play. It may happen that a technique exposes many failures, but they have a small impact on reliability because their occurrence frequency at operational time is low; and, oppositely, a technique may expose a few high-occurrence failures, thus delivering higher reliability.

Thus, a program containing more bugs than another may even be more reliable, inasmuch as those bugs cause less frequent failures in operation [3].

Ideally, operational testing is historically considered as the most promising approach at improving reliability, because it is the only testing method devised to find failures with probabilities matching their real runtime occurrence. However, researchers also raised serious doubts about the effectiveness of operational testing, which likely correspond to the reasons for why this approach has never gained the expected adoption in industry so far; examples follow.

- The first practical reason hindering the use of operational testing lies in the differences between the "idealized" version and the real application of operational testing. The main difference resides in the fundamental assumption of knowing the runtime *operational profile*. In operational testing, to deliver high (and accurate estimates of) reliability, it is necessary for the testing profile to be truly representative of operational use [4]. Deriving a correct profile is hard; and it is also not easy to figure out what happens if the testing profile does not match exactly the operational one. In this case, applying such a technique may even mislead testers by delivering a real reliability different than the measured one. This possibility does not inspire trust in operational testing.
- The second doubt is about the actual potential effectiveness of operational testing: stating that operational testing is oriented toward reliability does not necessarily mean that it is more efficient than any other at improving reliability. It may happen that debug testing techniques, other than uncovering many low-occurrence failures, are able to uncover also more high-occurrence failures than operational testing, with the same budget. This ability

can be exacerbated by the inaccuracy of the operational profile estimate mentioned above. With respect to this point, it is argued that, because operational testing cannot deal effectively with low-occurrence failures, it cannot replace debug testing [5]. This result is especially true for highly critical systems because, with operational testing, the most serious, potentially dangerous failures, occurring on low-probability error-condition paths, are probabilistically not exposed. This technique, applied alone, can improve the reliability level just up to a given bound; when the causes of high-occurrence failures are removed, it "exhausts" its detection power, and the reliability is no longer improvable in reasonable testing time. It may be more effective and efficient to consider using operational testing in conjunction with other techniques.

This paper intends to address the mentioned challenges, and paves the ground for reconsidering operational testing as a basic block in the frame of *reliability-driven* testing. The performance of operational testing is first examined with respect to the delivered reliability through a probabilistic evaluation, followed by simulation. The impact of the operational profile estimation error, which is the biggest issue, and of other parameters involved in the practical application of the technique, is evaluated. Then, a new testing strategy is proposed that combines *operational testing* with a *debug testing* approach to improve delivered reliability. In this strategy, a different view of *operational vs. debug* testing is developed, so as to overcome the limitations of the former in meeting its promises.

The formulated approach combines operational and debug testing based on the following conjecture: applying operational testing for some time allows achieving a level of reliability, which is hardly improvable by continuing with the same technique. To boost reliability beyond this limit, operational testing should be suitably combined with others to span over the entire code, and be directed also toward the failures with lower operational occurrence rate. The idea is that, when the number of failures exposed by operational testing starts decreasing significantly, replacing operational testing with a debug testing approach may detect additional failures, because new code portions are explored. To this aim, because operational testing exercises the system under the expected workload, the defined debug testing criterion aims at stressing the system under unexpected profile, providing unforeseeable and unexpected inputs. This, of course, does not necessarily imply an improvement of reliability, because it depends not only on the number but also on the occurrence rate of found failures. However, depending on several parameters, such as the time to switch from operational to debug testing, the number of test cases, and the operational profile features, we show that the probability to improve reliability is expected to increase as compared to operational testing.

The presented analysis explores the performance of both approaches under different conditions to help the tester capture the best settings in their application. We gain several results. *i)* We learn how operational testing is related to the profile estimation error, and how it is possible to take this error measure into account when applying the technique in practice. *ii)* We also learn how the proposed approach improves operational testing with respect to delivered reliability. *iii)* And we learn how the parameters involved in the application of both the techniques may be configured to achieve the desired performance. The analysis opens the way to exploring the performance and the power of *combining techniques* for a reliability target, rather than insisting on single techniques. The paper first reviews the relation between software reliability and testing in the literature (Section II). Then, it gives basic terminology and adopted assumptions in Section III; the probabilistic analysis of operational testing is reported in

Section IV, whereas the new proposed method is analyzed in Section V. In Section VI, both the approaches are evaluated via simulation, as well as via a case-study example (Section VII); a brief discussion is in Section VIII. Section IX concludes the paper.

II. Related Work

This Section surveys the literature about software testing and reliability that is related to the proposed approach.

A. Testing Techniques related to Reliability

Operational testing has been widely studied as the approach to test software toward a reliability goal[1]. Several researchers worked to improve this promising approach for reliability testing [8], [27], [28], [29]. The most important contexts in which operational testing has been used are in the frame of the *Cleanroom* approach, and in the process defined by Musa, i.e., the *Software Reliability Engineering Test* (SRET) practice [30].

In the *Cleanroom* development methodology [31], [32], [33], it is expected that operational testing detects any remaining failures with probabilities that are in proportion to their seriousness, and debug testing is therefore neglected. The Cleanroom method foresees adopting operational testing as a means to certify the software against a given MTTF [34]. Examples of an application of this approach are reported in [35], [32], [36]. With a similar position, Musa proposed SRET, where operational testing is the core element of the process. It entails four steps: *i)* define the necessary reliability requirement, *ii)* develop operational profiles, *iii)* prepare for testing, and *iv)* iteratively execute tests and interpret failure data [30]. Musa and his colleagues at AT&T found very relevant results in favor of operational testing, e.g., a reduction of a factor-of-10 in customer-reported problems [37]. Further evidence in favor of operational testing is reported in [38], and in [39] where operational testing is used with multimodal systems.

More recently, Chen et *al.* [40], [41] tried to improve the effectiveness of uniform random testing by taking account of the feedback information generated by those test cases that do not trigger defects. They use the concept of adaptive random testing (ART) [42], looking for test profiles, different from operational profile, able to evenly distribute test cases over the code. Adaptive testing is also the approach adopted in [43]. However, the objective of these works was directed toward improving failure detection capabilities, which, as discussed, is not the same thing as improving reliability. In [44], the same approach is used to *assess* software reliability (i.e., with the objective of minimizing the variance of the reliability estimator). From this point of view, it behaves better than other random testing strategies. One strategy intended to directly improve reliability is in [9], where the same authors of [43] and [44] adopt a Bayesian method to use online data for improving reliability. Specifically, they estimate the failure rate online, and select test cases from equivalence classes based on the observed outcomes as the testing proceeds. That work is close to ours in its objective, but the way in which the problem is faced is quite different. It aims at

[1]The term *Statistical testing* is also used to denote this technique. However, *Statistical testing* is more recently being used to denote also a slightly different approach whose goal is to satisfy an adequacy criterion expressed in terms of functional or structural properties (e.g., ensuring that each structural element is exercised as frequently as possible [26]). In this context, *Statistical testing* is intended as its original use, such as in [27], [28], and interchangeably with *Operational testing*.

improving reliability by selecting test cases from the best partition according to the history of the testing process; we aim at improving reliability by selecting test cases first from operational testing, and then from a debug testing technique, accounting for the profile estimate error, and the available budget. The two strategies are not mutually exclusive, but are complementary; the Bayesian model can be seamlessly superimposed to our strategy to improve the specific test case selection within both operational and debug testing phases. In turn, our approach would improve the overall reliability by deciding how to combine operational and debug testing, given a tolerable error, a budget T in terms of number of test cases, and an expected improvement in reliability.

Although the cited approaches promote operational and reliability-oriented testing, reporting good results, it is still difficult, at this time, to find compelling evidence that operational testing is always a good practice. In fact, there are studies raising strong doubts about the effectiveness (e.g., [5], [45]) of operational testing. For instance, Beizer strongly argued that the Cleanroom process has never been compared fairly with other approaches, and that abandoning debug testing in favor of operational testing would be a wrong choice [5]. The main debated reason is that, because operational testing cannot deal with low-occurrence failures, it cannot replace at all debug testing, especially for highly critical systems. This argument partly motivated our research on exploring the combination of techniques toward a reliability goal, to overcome this strong limitation.

One final point is about the role of the operational profile. Indeed, the effectiveness of operational testing has always been seriously undermined by the difficulties in obtaining meaningful profiles. Deriving the operational profile has been one of the major challenges in applying operational testing. A non-negligible slice of literature is devoted to proposing solutions to support this task, such as state models [46], Markovian or Bayesian models for the expected usage description [27], [47], or documentation based on UML diagrams [48]; but basically the accuracy of the operational profile depends on the accuracy of the information on how the system will be used [16], which is difficult to obtain. Empirical analyses have been conducted to evaluate the *impact* of the profile estimation error on the reliability estimate. These studies are quite contradictory. The studies in [13], [49] state a non-negative effect of operational profile errors on the reliability estimate. In [50], reliability estimates are showed to not be affected by the error on the profile. Operational profile error is also evaluated in [17], in which the performance of techniques in assessing reliability in presence of error is experimentally evaluated. Different from the previous papers, the work by Chen et *al.* [51] demonstrates how operational profile errors cause relevant errors on reliability estimates. A more recent analysis shows that the relation between the operational profile variation and reliability for component testing depends on the error of the testing effort [52].

Despite these contrasting results, it is important to remark that the relation between the profile estimation error and reliability estimate has been studied empirically so far, by observing the effect of the error once the testing has been carried out (i.e., *ex post*). No analytical, preventive evaluation is reported about the possible impact of this error on testing performance, as we intend to do here. And this is also one of the reasons for why it remains so hard to apply techniques oriented to operational reliability in practice. Our work started addressing this issue, considering the operational profile estimation error as one of the crucial factors to take preemptively into account for effective operational testing.

B. Comparison of Techniques from a Reliability Perspective

To figure out *how to test* for operational reliability improvement, and what technique to adopt, a valuable way is to compare techniques from a reliability perspective. However, evaluating techniques by their delivered reliability has not gained great attention from researchers. Several criteria have been proposed since the eighties: the simple subsumption relation, the power relation, and the BETTER relation (a survey on this topic is in [19]), but none of them is related to reliability. With respect to reliability, one relevant work is represented by the study of 1984 [20] by Duran and Ntafos, where authors addressed the problem by using a probabilistic measure, followed then by others [21]. Along this trend, authors in [22] introduced the properly covers and universally properly covers relations, which use the probability that a test suite exposes at least one failure. These papers represented an important step because research in this field switched to probabilistic comparison criteria. In fact, due to the high number of uncontrollable variables involved, it is very difficult to say that a test suite or a criterion is deterministically better than another; it is indeed more practical and easier to state that a criterion is "probabilistically" better or worse than another.

None of these studies, however, actually considered *reliability* as measure to compare techniques. Few works pursued this direction. The problem is initially discussed in [23], [21]. Chen and Mathur [24] remarked in 1995 that further research was necessary to determine which testing method, or combination thereof, could lead to higher reliability. Since then, one of the most remarkable analyses remained the one by Frankl et al. [4], where authors analytically compared debug testing against operational testing, with the goal of evaluating which technique produces a higher *delivered reliability*. Other similar works are then reported by the same authors [10], [25]. The low number of works studying testing from this point of view is justified by considering the difficulties associated with evaluations against a user-perceived attribute, such as reliability, which basically remained unsolved. Problems like the uncertainty of the operational profile, the unknown impact of the profile estimation error, and the doubts on the actual performance of operational testing, all caused a poor application by industry, which in turn limited empirical analyses and comparisons, and did not provide the expected feedback to the analytical research. These issues are, instead, the main subjects in the presented work.

III. Terminology and Assumptions

This Section clarifies the terminology and the assumptions adopted in the following analysis. Testing a program is the process of *i)* exercising the program with different test cases, selected from the set of all possible inputs according to a selection criterion; and *ii)* observing the output, and comparing it with the expected one such that if they are discordant, a *failure* is said to have occurred. The subset of inputs provoking *failures* are called *failure-causing inputs* or *failure points*. When a failure occurs, a change is made to the program to remove what is believed to be the cause of the failure, or "fault". While a failure is uniquely defined as a deviation from the expected output, a fault is defined and characterized by the semantic of the fix, i.e., by the *change* to the code necessary to make the program work [6]. However, because there may be several possible changes able to avoid the failure, the fault related to an observed failure is not uniquely defined. For this reason, we rely upon the notion of failure rather than

that of fault, as in [4], and [7]; and we borrow the notion of *failure region* of the input space, as the set of failure points that is eliminated by a program change. This is also in line with testing principle, whose aim is not to "find a fault," but to "expose a failure". The set of all failure regions is called *failure region domain* (F)[2].

An operational profile is a quantitative characterization of how a system will be used [8]. It is built by assigning probability values to all input cases representing the probability that each will occur in operation; thus it can be thought of as the probability distribution over the set of all input points, D. This distribution, which assigns a probability value in $[0, 1]$ to each value in D, has been denoted with H. Thus, probability values (denoted with h_j) represent the probability that the input $j \in D$ is selected during operation [3].

The distribution Q is also defined, and it assigns a value in $[0, 1]$ to each value in F, i.e., to each failure region. Thus, q_i with $i \in F$, is the probability that an element of the input domain belonging to the failure region F_i will be selected during operation. We call this value the *failure rate* of the failure region F_i. More formally, given a failure region $F_i \in F$, its failure rate is $q_i = \sum_{j \in F_i} h_j$. Consider the *testing profile* G, defined again as a distribution over D, used by a generic testing method as a sample space to select test cases; it assigns a value g_j to all inputs $j \in D$ representing the probability of the input j being selected by that testing method. The *detection rate* of a failure region is the probability that an element of that region will be selected during testing: given a failure region $F_i \in F$, its detection rate, denoted d_i, is $d_i = \sum_{j \in F_i} g_j$.

Testing methods are evaluated with respect to the *delivered (un)reliability*, a measure introduced by Frankl et *al.* [4], i.e., to the probability that the corrected software will fail in operation. The failure probability of the software in operation is a random variable, e.g., θ; we focus on its expected value $E[\theta]$. Operational reliability is then the probability of the program surviving N executions on inputs drawn from the operational profile: $R(N) = (1-\theta)^N$. Note that the traditional engineering definition of reliability is not a fixed probability; it is a function of the number of operational trials[4]. However, for the type of analysis we carry out, namely on testing technique behavior, reliability is typically measured as we mentioned [9], [4], [10], under the assumption that runs are *s*-independent [11].

A. Operational Testing

In operational testing, test cases are selected with the same probabilities with which they would occur in operation. Ideally, with a perfect estimate, h_j is also the probability value selected during testing. But in the real world, from the estimate of operational profile affected by error, another probability distribution is obtained, called $\hat{H}$, which is the testing profile of operational testing; hence, in this case, $G = \hat{H}$, and $g_j = \hat{h}_j$. The latter is the probability of selecting j in operational testing. Thus, in case of operational testing, the detection rate of a given failure region

[2]Note that an element of F is not a failure point, but is a set of failure points belonging to a failure region.

[3]In the following, capital letters are used to denote distributions, and the corresponding subscripted lower-case variables denote the assigned probability values.

[4]Reliability is the probability of not failing in a given interval of time. See [12] for a discussion on this point and on the relation between the frequency-based measurement, i.e., in terms of successful runs over all the runs, and the time-based measurement, i.e., as function of the hazard rate

$F_i \in F$ is $d_i = \sum_{j \in F_i} \hat{h}_j$. We call these detection rate values $\hat{q}_i$ because they try to approximate the failure rates q_i, and the corresponding distribution $\hat{Q}$. To account for the operational profile estimation error in the analysis, we consider the differences $(\hat{h}_j - h_j)$, which are the errors for each point of the input domain.

Such errors actually impact the delivered reliability only through inputs belonging to the failure regions (see [4]). Thus, for the following analysis, we define the variable $\xi = (\hat{Q} - Q)$, referring to the failure region F_i; it follows that $\xi_i = \sum_{j \in F_i} (\hat{h}_j - h_j) = \hat{q}_i - q_i$ with $i \in F$.

B. Debug Testing

To complement operational testing, we need to define a debug testing criterion aimed at stressing the application under exceptional and "unexpected" inputs, oppositely to operational testing. The idea is to provide inputs that are rare at operational time, so as to uncover the more subtle bugs, to explore the least covered code portions, and to cause low-probable failures to surface. The testing profile for such a debug testing method is denoted with V (i.e., in this case $G = V$), and an element of such distribution is v_j. Because the criterion has to exercise the system by exceptional and rare inputs, v_j should be high when h_j is low, and low when h_j is high. One additional constraint is that the sum of all probabilities should add to 1. Thus, denoting the maximum of the $\hat{h}_j$ values[5] as $\max(\hat{h}_j) = \max_{j \in D}(\hat{h}_j)$, we define the v_j normalized values, used to select test cases, as

$$v_j = \frac{(\max(\hat{h}_j) - \hat{h}_j)}{\sum_{j \in D} (\max(\hat{h}_j) - \hat{h}_j)} = \frac{(\max(\hat{h}_j) - \hat{h}_j)}{|D| \cdot \max(\hat{h}_j) - \sum_{j \in D} \hat{h}_j} = \frac{(\max(\hat{h}_j) - \hat{h}_j)}{|D| \cdot \max(\hat{h}_j) - 1} \tag{1}$$

We call such an approach "anti-profile" debug testing, to denote its contrast with operational testing. This criterion helps to combine operational (OP) and debug testing (hereafter, OP-D testing), in an approach to increase the probability of uncovering both high- and low-occurrence failures. How this approach impacts (un)reliability will be evaluated in the next sections. Note that there may be some other criteria for the debug testing approach satisfying the same constraints. For instance, one more solution is $v_j = (1 - \hat{h}_j) / \sum_{j \in D} (1 - \hat{h}_j)$. The adopted criterion is chosen because it is a more "extreme" solution, as it assigns a higher value to the same low h_j value compared to the latter criterion, and vice-versa; this result may better highlight the benefit of our approach. The detection rate, in the case of debug testing, of a given failure region $F_i \in F$, is

$$d_i = \sum_{j \in F_i} v_j = \frac{(\max(\hat{h}_j) \cdot |F_i| - \hat{q}_i)}{|D| \cdot \max(\hat{h}_j) - 1} \tag{2}$$

We call these detection rate values r_i; they will be used against $\hat{q}_i$ values in the evaluation.

C. Evaluation

To evaluate the OP and OP-D testing methods, we perform a probabilistic analysis. The assumptions made in the analysis follow. Testers are assumed to eliminate the same failure regions s-independently from the technique

[5]Actually, to define v_j we have to use $\hat{h}_j$ instead of h_j values, because the latter are unknown

used. Indeed, the debugging is in reality affected by the technique used to probe the fault, and thus different sets of failure points could be removed by different changes. Failure regions are assumed to be disjoint, and all test failures are noticed (i.e., perfect oracle). Hence, each test failure deterministically causes one failure region to be removed. Finally, the statistical nature of the analysis does not allow claiming the deterministic superiority of one technique on another, but only that a technique is likely to be superior to another.

IV. Operational Testing

This section examines the performance of operational testing analytically. The case of multiple failure regions being present in the software is considered. Suppose the failure region domain F consists of m non-overlapping failure regions $\{F_1, F_2, \ldots, F_m\}$, with probabilities $q_1, q_2, \ldots, q_m$ of being selected in operation (i.e., q_i with $i \in F$ are the failure rates).

Of course, testers do not know which inputs belong to failure regions in F. They try to minimize the expected failure probability by selecting, with different criteria, proper inputs as test cases. In the case of operational testing, practitioners give an estimate of the operational profile, which is, in general, affected by error, and then select inputs according to that profile. To evaluate the real practical applicability of OP testing, the impact of this error must be considered. To this aim, let us first assume that the estimation is fully correct, i.e., $\xi_i = 0 \quad \forall i \in F$, which implies $d_i = q_i, \quad \forall i \in F$. The expected failure probability for the error-free case, denoted as $E[\varphi]$, is

$$E[\varphi] = \sum_{i=1}^{m} q_i (1 - d_i)^T = \sum_{i=1}^{m} q_i (1 - q_i)^T \tag{3}$$

which adds up the probabilities of i not being detected during T tests $(1 - q_i)^T$, and being selected in operation (q_i) (i.e., i causing a failure). Whereas, in practice, considering the estimation error, the expected failure probability of OP testing, denoted as $E[\omega]$, is

$$E[\omega] = \sum_{i=1}^{m} q_i (1 - \hat{q}_i)^T \tag{4}$$

where $\hat{q}_i$ are the estimates of q_i. To compare it with the *Error-free* OP testing, let us consider the relation $E[\omega] < E[\varphi]$. It can be written as

$$\sum_{i=1}^{m} q_i (1 - \hat{q}_i)^T < \sum_{i=1}^{m} q_i (1 - q_i)^T \tag{5}$$

The difference between the two terms depends exclusively upon the values of q_i and $\hat{q}_i$. If we consider $m = 1$, i.e., one failure region z in the program, $E[\omega] < E[\varphi]$ if $\hat{q}_z > q_z$; thus the probability that OP testing is better than *Error-free* OP testing is equal to the probability that $\hat{q}_z > q_z$. If $m > 1$, the failure rate of each failure region must be taken into account. To this aim, let us consider the event A_i: $(\hat{q}_i \geq q_i)$. The probability that $E[\omega] < E[\varphi]$, for a given T, is

$$Pr(E[\omega] < E[\varphi]) = \sum_{i \in F} Pr(A_i) q_i \tag{6}$$

The probability of the event A_i depends on the choice of the operational profile, hence on the estimation error committed by testers and analysts. However, the result in this case does not depend only on the difference $\xi_i = \hat{q}_i - q_i$,

but also on the failure rate, q_i; errors committed on input values with higher q_i have higher impact. To evaluate how the performance of OP testing is related to these values, the following *weighted error* variable is now considered. $\Upsilon = \sum_{i \in F} q_i(\hat{q}_i - q_i) = \sum_{i \in F} q_i \xi_i$. In particular, let us consider the case in which the error $\Upsilon \neq 0$. This condition leads us to assert that the expected failure probability of OP testing, $E[\omega]$, is different from the *Error-free* case, $E[\varphi]$. We might expect that, whenever the weighted error is negative, the OP testing should deliver a higher failure probability ($E[\omega] > E[\varphi]$); in fact, if the error is negative, it means that OP testing underestimates the operational failure probability of the failing inputs, and thus it will detect those regions with a lower probability. Similarly, when $\Upsilon > 0$, OP testing is expected to be better than *Error-free* OP testing ($E[\omega] < E[\varphi]$) because failing input probabilities are overestimated. However, our intuition is not always correct. It may happen that, even with a negative error, OP testing may behave better than *Error-free* OP testing, as well as the opposite. To show this result, let us consider $E[\varphi] - E[\omega] = 0$. It can be rewritten as

$$\begin{aligned}
&\sum_{i=1}^{m} q_i[(1-q_i)^T - (1-\hat{q}_i)^T] = \sum_{i=1}^{m} q_i[(1-q_i)^T - (1-q_i-\xi_i)^T] \\
&= \sum_{i=1}^{m} q_i[(1-q_i)^T - [(1-q_i)^T + \sum_{k=1}^{T} \binom{T}{k} (1-q_i)^{T-k}(-\xi_i)^k]] \\
&= \sum_{i=1}^{m} q_i[-\sum_{k=1}^{T} \binom{T}{k} (1-q_i)^{T-k}(-\xi_i)^k]] = 0
\end{aligned} \tag{7}$$

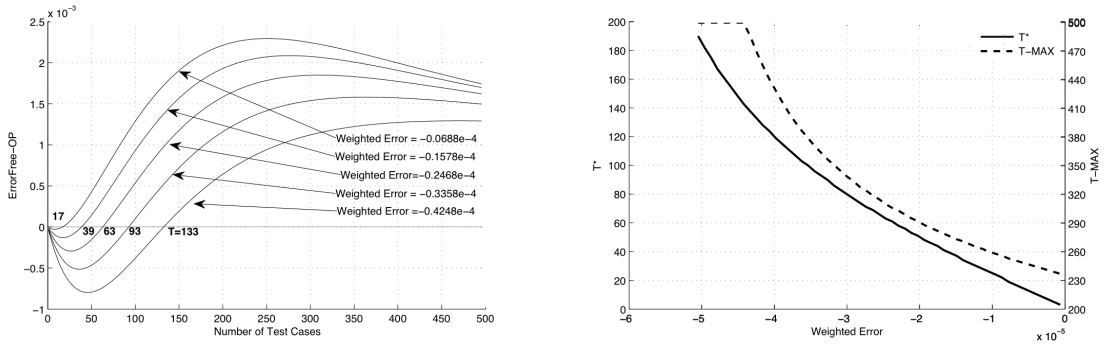
To figure out how the error is related to the difference $E[\varphi] - E[\omega]$, the equation $E[\varphi] - E[\omega] = 0$ is solved with respect to $\Upsilon = \sum_{i=1}^{m} q_i(\xi_i)$. Expanding the term $(1-q)^{T-k}$,

$$\begin{aligned}
&\sum_{i=1}^{m} q_i[-\sum_{k=1}^{T} \binom{T}{k} [\sum_{j=0}^{T-k} \binom{T-k}{j} (-q_i)^j] \ (-\xi_i)^k]] \\
&= \binom{T}{1} \Upsilon + \sum_{i=1}^{m} q_i[-\sum_{k=1}^{T} \binom{T}{k} (\sum_{j=1}^{T-k} \binom{T-k}{j} (-q_i)^j) \ \ (-\xi_i)^k] = 0
\end{aligned} \tag{8}$$

Thus, $E[\varphi] = E[\omega]$, under the condition $\Upsilon \neq 0$, if

$$\Upsilon = \frac{\sum_{i=1}^{m} q_i[\sum_{k=1}^{T} \binom{T}{k} (\sum_{j=1}^{T-k} \binom{T-k}{j} (-q_i)^j) \ (-\xi_i)^k]]}{T} \tag{9}$$

As may be noted, the error Υ is related to the number of test cases T, and to ξ_i^k values, which we call *higher-order* error terms, in turn related to q_i and $\hat{q}_i$ values. Thus, once the profile (determining the $\hat{q}_i$ values) is decided, the expression depends on T and on q_i terms: $\Upsilon = f(T, Q)/T$, with Q deriving from the correct profile. According to (9), $E[\varphi] < E[\omega]$ if $\Upsilon < f(T, Q)/T$, and conversely $E[\varphi] > E[\omega]$ if $\Upsilon > f(T, Q)/T$. The latter case is of interest because $E[\omega]$ is the technique actually applied by practitioners, and they want it to deliver a lower failure probability. Thus, the relation between the error Υ and T establishes the following bounds.



(a) Number of Test Cases vs. E(φ) - E(ω)

(b) Weighted Error Υ vs. T^* and vs. T-MAX

Fig. 1. Numerical Example of OP testing performance

- Given a fixed budget of test cases T^*, the weighted error on the operational profile must be at least greater than $f(T^*, Q)/T^*$ for OP testing to be better than the *Error-free* case. Therefore, it may be even negative, but it must be not less than $f(T^*, Q)/T^*$.
- Given a maximum tolerable error, Υ^*, a tester must execute at least T^* test cases to obtain the same or better performance than the *Error-free* case, where T^* is the value for which $f(T^*, Q)/T^* = \Upsilon^*$.

Note that, for $T = 1$, (9) provides $E[\varphi] = E[\omega]$ when $\Upsilon = -\sum_{i=1}^{m} q_i \xi_i = -\Upsilon$. This happens only for $\Upsilon = 0$.

To further clarify, we report, in Fig. 1(a), a numerical example with $E[\varphi] - E[\omega]$ plotted against the number of test cases T, given a negative Υ value (which is the worst case for the tester). In this example, the initial q_i and $\hat{q}_i$ values are randomly selected, and reported in Table I[6].

TABLE I

EXAMPLE: INITIAL q_i AND $\hat{q}_i$ VALUES

q_i values	0.0094	0.0089	0.006	0.0085	0.0012	0.0144	0.0007459	0.015	0.017	0.001
$\hat{q_i}$ values	0.0121	0.0067	0.0142	0.0127	0.0162	0.0162	0.0116	0.0086	0.0122	0.0023
$q_i(q_i - \hat{q_i})\cdot$ e-04	-0.2538	0.1958	-0.4920	-0.3570	-0.1800	-0.2592	-0.0810	0.9600	0.8160	-0.0130
Initial $\Upsilon = -3.3584$e-05										

The Figure shows five different curves, each one with a different value of the error, obtained by varying $\hat{q}_i$ values. When the curve is positive, OP testing performs better than *Error-Free* OP testing. The crucial value of the number of test cases, T^*, for which this happens is the intersection with the x-axis. Note that, as the error Υ approaches 0, such a number decreases from 133 down to 17 (i.e., with Υ being close to 0, a lower number of test cases is required for OP testing to perform better than *Error-Free* OP testing). After this critical value, the performance of OP testing is always better. With the number of test cases approaching infinity, the performance of OP testing

[6]Note that the summation is not equal to one because it includes only failure regions, not all the input values of D

gets closer to the performance of *Error-Free* OP testing (they both tend to deliver a failure probability equal to 0). Fig. 1(b) provides a more complete view of T^*. Given a (maximum tolerable) error value, the solid line suggests the number of test cases, T^*, to allocate in order to have $E[\omega] < E[\varphi]$. As the error approaches 0, fewer test cases are needed for OP testing to perform better than the *Error-free* case.

Finally, one more value of interest is the value of T for which the curves of Fig. 1(a) exhibit their maximum value, indicating the best performance of OP testing; let us denote it with T-MAX. In Fig. 1(b), the dotted line depicts the relation between Υ and T-MAX; it suggests the number of test cases to perform to have the best performance under a given error, and conversely what is the expected error to achieve in order to not overcome a given budget T-MAX, and obtaining the best performance. T-MAX starts decreasing from 500 when the error is less than about 4.3E-5, and then decreases rapidly with the error approaching 0. As an example, consider that, when the error values are the same as in Fig. 1(a), namely 4.24E-5, 3.35E-5, 2.46E-5, 1.57E-5, 0.68E-5, the corresponding T-MAX values are 470, 364, 310, 276, 251. These T-MAX values represent the number of test cases to perform in order to have the best performance of OP testing with those errors. From (9), and from this example, it is clear that, although (6) indicates the probability of OP testing being better than the ideal *Error-free* case, it is not exhaustive. That probability varies according to the number of test cases, and the profile estimation error. Equation (9) makes it explicit, giving the conditions under which OP testing outperforms the *Error-free* case.

V. OP-D Testing

This section presents the OP-D testing approach. The idea behind the strategy is to exploit the combination of specular techniques with the goal of addressing failure regions both with high and with low operational failure rates. The debug testing approach used in this work, called "anti-profile" debug testing, is defined in Section III-B. The objective is to adopt OP testing until its detection ability no longer yields satisfactory results, and then switching to debug testing to increase the number of detected failure regions. Reliability will increase only if regions found by debug testing will have a total operational occurrence probability greater than the ones potentially found by continuing with operational testing. This condition is evaluated probabilistically.

Let us denote with $E[\delta]$ the expected failure probability delivered by the OP-D technique after $T = T_1 + T_2$ test cases. With r_i being the detection rates of debug testing, and $\hat{q}_i$ the detection rates of operational testing, the expected failure probability of OP-D is given by

$$E[\delta] = \sum_{i \in F} q_i (1 - \hat{q}_i)^{T_1} (1 - r_i)^{T_2} \tag{10}$$

This equation considers the probability of the failure region i not being detected by operational testing during T_1 test cases, not being detected by debug testing during T_2 test cases, and occurring at runtime (q_i)[7]. There are two factors that affect the performance of this combination: the accuracy of the operational profile, and the number of test cases devoted to debugging and to operational testing (or, in other words, the testing time to switch from operational to debug testing). In the following, the comparison between OP-D and OP testing is performed accounting for them.

[7]In the subsequent analysis, we consider the OP testing with error in lieu of error-free OP testing, without loss of generality.

A. Single Residual Failure Region

Let us first consider the simple case in which operational testing has been carried out until $|F|-1$ failure regions have been found. The residual number of regions is one. Let us denote this subset of failure domain as F', being $|F'|=1$. The question is whether changing to debug testing improves or worsens the final reliability as compared to continuing with operational testing.

Given $T = T_1 + T_2$, the failure probability of OP testing after T test cases can be written as $E[\omega] = \sum_{i \in F} q_i (1-\hat{q_i})^{T_1}(1-\hat{q_i})^{T_2}$. Thus, the difference between $E[\omega]$ and $E[\delta]$ is determined by the second term (cf. with (10)). Assuming that all the failure regions except one, denoted with z, have been detected during T_1 operational test cases, the comparison is reduced to verify whether

$$\begin{aligned} &\Delta E[\omega] = q_z(1-\hat{q_z})^{T_1}(1-\hat{q_z})^{T_2} \geq \Delta E[\delta] = q_z(1-\hat{q_z})^{T_1}(1-r_z)^{T_2} \\ &\Rightarrow (1-\hat{q_z})^{T_2} \geq (1-r_z)^{T_2} \Rightarrow \hat{q_z} \leq r_z \end{aligned} \tag{11}$$

where $\Delta E[\cdot]$ represents the increase in the failure probability caused by the residual failure region z. In other words, the combined approach gives better reliability only if $r_z > \hat{q_z}$. For $|F'|=1$, it can be thought that this probability is high to the extent that T_1 is high. Intuitively, if the residual region z has not been detected by operational testing in T_1 test cases, and T_1 is high, this means that its $\hat{q_i}$ value is likely low; as a consequence its r_i value is high. But this is not deterministic; we need to evaluate the probability that such an event occurs. In particular, we have that $Pr(E[\omega] \geq E[\delta]) = Pr(r_z \geq \hat{q_z} | (z \text{ has not been detected during operational testing}))$. Let us define these events as follows.

$A = (r_z \geq \hat{q_z})$, without any prior information;

$B = (z$ has not been detected during operational testing), with $Pr(B) = (1-\hat{q_z})^{T_1}$.

Thus, $Pr(E[\omega] \geq E[\delta]) = Pr(A|B)$. By Bayes' formula, we have $Pr(A|B) = Pr(B|A)Pr(A)/Pr(B)$.

Let us evaluate this expression. Because r_z is derived from $\hat{q_z}$, the event *A* may be written, recalling (2), as $r_z = \max(\hat{h}_j) \cdot |F_z| - \hat{q_z}/|D| \cdot \max(\hat{h}_j) - 1 \geq \hat{q_z} \Rightarrow \hat{q_z} \leq |F_z|/|D|$. Being $F_{\hat{Q}}(t)$ the distribution function of $\hat{Q}$, and posing $c = |F_z|/|D|$, we have $Pr(A) = F_{\hat{Q}}(c) = Pr(\hat{q_z} \leq c)$. The event $B|A$ represents the failure region z not being detected during operational testing, given that $\hat{q_z} \leq r_z$. The probability that this event occurs, $Pr(B|A)$, is intuitively greater than $Pr(B) = (1-\hat{q_z})^{T_1}$ because it considers only the cases in which $\hat{q_z} \leq c$ (the smaller $\hat{q_z}$ the greater the probability that it will not be detected by operational testing). In other terms, we are taking into account the case in which $\hat{q_z}$ is selected in a range $[0,1]$ (i.e., the event B), against the case in which the value $\hat{q_z}$ is selected in a range $[0,c]$, with $c<1$ (i.e., the event $B|A$), being $|F_z| < |D|$ (typically much smaller). Let us denote with $\hat{q_z}'$ this second case; thus, $Pr(B|A) = (1-\hat{q_z}')^{T_1}$, and

$$\begin{aligned} &Pr(E[\omega] \geq E[\delta]) = Pr(A|B) \\ &= \frac{Pr(B|A)Pr(A)}{Pr(B)} = \frac{(1-\hat{q_z}')^{T_1}}{(1-\hat{q_z})^{T_1}} \cdot F_{\hat{Q}}(c). \end{aligned} \tag{12}$$

Let us evaluate the ratio $Pr(B|A)/Pr(B)$. With this formulation, we have $Pr(Pr(B|A) > Pr(B)) = Pr(\hat{q_z}' < \hat{q_z}) = Pr(\hat{q_z}' < \hat{q_z}|\hat{q_z} \leq c) + Pr(\hat{q_z}' < \hat{q_z}|\hat{q_z} > c) = 0.5F_{\hat{Q}}(c) + 1 \cdot Pr(\hat{q_z} > c) = 0.5F_{\hat{Q}}(c) + (1 - F_{\hat{Q}}(c)) = 1 - 0.5F_{\hat{Q}}(c)$. This value is always greater than 0.5, meaning the ratio $Pr(B|A)/Pr(B) > 1$ with probability greater than 0.5. This probability depends exclusively on $|F_z|$, and $|D|$. Because $|F_z| < |D|$, it approaches 1 as $|D|$ approaches infinity. From the previous expression, it derives that $Pr(A|B) > Pr(A)$ (and thus $Pr(E[\omega] \geq E[\delta]) > F_{\hat{Q}}(c)$) with probability $1 - 0.5F_{\hat{Q}}(c)$. *This relation establishes a lower bound for the probability of the OP-D strategy to be better than OP testing, and a "confidence" in this bound*. The probability that OP-D is better than OP testing is higher than $F_{\hat{Q}}(c)$ with a confidence of $(1 - 0.5F_{\hat{Q}}(c))\%$.

Consider one more point from (12): if $Pr(B|A) > Pr(B)$, then the value $Pr(A|B)$ increases (and thus OP-D improves) as T_1 increases because, in such a case, the ratio $Pr(B|A)/Pr(B) = (1 - \hat{q_z}')^{T_1}/(1 - \hat{q_z})^{T_1}$ is greater than 1. This relationship is true because, if operational testing is not able to find $\hat{q_z}$ in T_1 test cases, with T_1 very high, it means that the $\hat{q_z}$ value is very low, and thus r_z is very high. Thus, while the confidence depends only on the profile, the value of $Pr(A|B) = Pr(E[\omega] \geq E[\delta])$ depends also on the number of test cases assigned to operational testing, T_1. In fact, putting together the previous two statements leads to assert that the probability of OP-D being better than OP ($Pr(E[\omega] \geq E[\delta])$) increases with T_1 (or with T_2 decreasing) with confidence $(1 - 0.5F_{\hat{Q}}(c))\%$. The amount of increasing of unreliability is directly related to $\hat{q_z}$, i.e., to the residual failure region: the failure probability increase in $E[\delta]$ is $q_z(1 - r_z)^{T_2}$, whereas the increase in $E[\omega]$ is $q_z(1 - \hat{q_z})^{T_2}$.

B. Multiple Residual Failure Regions

Let us now consider the case in which the number of residual failure regions after operational testing is greater than 1, i.e., $|F'| > 1$. In this case, the delivered reliability depends on which regions among the residual ones are found by anti-profile debug testing, as compared to the ones that are found by continuing with operational testing, and on the sum of their failure rates. After the first phase of operational testing, the additional contribution to the failure probability is $\sum_{i \in F'} q_i(1 - \hat{q_i})^{T_1}(1 - r_i)^{T_2}$ for OP-D testing, and $\sum_{i \in F'} q_i(1 - \hat{q_i})^{T_1}(1 - \hat{q_i})^{T_2}$ for OP testing. Similar to the previous case, two events are defined:

A_i is the event $(r_i \geq \hat{q_i}) = (\hat{q_i} \leq |F_i|/|D|)$, and

B_i is the event that *i* has not been detected during operational testing, with $Pr(B_i) = (1 - \hat{q_i})^{T_1}$

The probability that $E[\omega] \geq E[\delta]$ is

$$Pr(E[\omega] \geq E[\delta]) = \sum_{i \in F'} Pr(A_i|B_i)q_i \tag{13}$$

It is clear that the increase in the failure probability depends on q_i. Recalling (12) in the previous section, the expression may be written as

$$\begin{aligned} Pr(E[\omega] \geq E[\delta]) &= \sum_{i \in F'} \frac{Pr(B_i|A_i)}{Pr(B_i)} q_i \\ &= \sum_{i \in F'} \frac{(1 - \hat{q_i}')^{T_1} F_{\hat{Q}}(|F_i|/|D|)}{(1 - \hat{q_i})^{T_1}} q_i = F_{\hat{Q}}(|F_i|/|D|) \cdot \sum_{i \in F'} \frac{(1 - \hat{q_i}')^{T_1}}{(1 - \hat{q_i})^{T_1}} q_i \end{aligned} \tag{14}$$

The expression is regulated by two parameters: by T_1, or equivalently, T_2, and by the pair $(\hat{Q}, |D|)$, depending on the profile. T_1 may determine the probability of OP-D being better than OP: in particular, *if the ratio* $Pr(B_i|A_i)/Pr(B_i)$ *is greater than* 1, then the corresponding failure probability increases with T_1, meaning that OP-D testing behaves better as T_1 increases. On the other hand, the Q and D pair determines the confidence at which that ratio is greater than 1. In fact, if $|D|$ is high, the probability that it is greater than 1 approaches 1 because the q_i' are more probably lower than q_i. However, the pair (Q,$|D|$) affects also the failure probability value, not only the confidence (suffice it to note that the term $F_{\hat{Q}}(|F_i|/|D|)$ goes to 0 as $|D|$ increases). In other words, given a selected profile, the corresponding r_i depend on the cardinality of the input domain, $|D|$ (cf. with (2)). In this sense, this parameter regulates the confidence of the information on which technique behaves better; the amount of failure probability is then also determined, other than by the profile, by how the test budget T is distributed as T_1-T_2. In the end, the parameter T_1 is completely under the control of the tester, but the profile may be not; thus, in the next section, an analysis considering the features of the profile is carried out so as to help the tester find the best settings.

C. Profile vs. Test cases in OP-D testing

Let us consider $E[\omega] - E[\delta] = 0$. It may be written as

$$\begin{aligned}
&\sum_{i=1}^{m} q_i[(1-\hat{q_i})^T - (1-\hat{q_i})^{T_1}(1-r_i)^{T_2}] \\
&= \sum_{i=1}^{m} q_i[(1-\hat{q_i})^{T_1}(1-\hat{q_i})^{T_2} - (1-\hat{q_i})^{T_1}(1-r_i)^{T_2}] \\
&= \sum_{i=1}^{m} q_i(1-\hat{q_i})^{T_1}[(1-\hat{q_i})^{T_2} - (1-r_i)^{T_2}] = 0
\end{aligned} \tag{15}$$

Results of this expression depend on the differences between $\hat{q_i}$ values and the corresponding r_i values. Thus, we define, similarly to the Υ variable, the weighted difference $\Psi = \sum_{i=1}^{m} q_i(\hat{q_i} - r_i)$. Equation (15) may be written as

$$\sum_{i=1}^{m} q_i \sum_{k=0}^{T_1} \binom{T_1}{k} (-\hat{q_i})^k [\sum_{j=0}^{T_2} \binom{T_2}{j} (\ (-\hat{q_i})^j - (-r_i)^j\ )] = 0 \tag{16}$$

Considering that, for $k = 0$, and $j = 1$, the previous expression is $\sum_{i=1}^{m} q_i T_2(-(\hat{q_i} - r_i)) = -T_2 \cdot \Psi$. Replacing $T_1 = T - T_2$, we rewrite (16) as

$$-T_2\Psi = -\sum_{i=1}^{m} q_i \sum_{k=1}^{T-T_2} \binom{(T-T_2)}{k} (-\hat{q_i})^k (\sum_{j=2}^{T_2} \binom{T_2}{j} (\ (-\hat{q_i})^j - (-r_i)^j\ )) \tag{17}$$

Ψ is therefore related to total the number of test cases T, to the number of test cases devoted to debug testing T_2, as well as to the profile determining $\hat{q_i}$ and r_i values. We then have

$E[\omega] = E[\delta]$ if $\Psi = g(T, T2, \hat{Q})/(-T_2)$
$E[\omega] > E[\delta]$ if $\Psi < g(T, T2, \hat{Q})/(-T_2)$, and
$E[\omega] < E[\delta]$ if $\Psi > g(T, T2, \hat{Q})/(-T_2)$

where g denotes a function of $T, T_2, \hat{Q}$, the second term of (17). These expressions regulate the probability of OP-D testing being better than OP testing, depending on the weighted differences between q_i and r_i values. However, it should be noted that the variable Ψ is also related to the error Υ; changing values of $\hat{q}_i$ affects also the difference $\hat{q}_i - q_i$. In particular,

$$\begin{aligned}
\Psi &= \sum_{i=1}^{m} q_i(\hat{q}_i - r_i) = \sum_{i=1}^{m} q_i(\hat{q}_i - \frac{\max(\hat{h}_j)\cdot|F_i| - \hat{q}_i}{|D|\cdot\max(\hat{h}_j) - 1}) \\
&= \sum_{i=1}^{m} q_i(\frac{\hat{q}_i\cdot|D|\cdot\max(\hat{h}_j) - \max(\hat{h}_j)\cdot|F_i|}{|D|\cdot\max(\hat{h}_j) - 1}) \\
&= \frac{\max(\hat{h}_j)}{|D|\cdot\max(\hat{h}_j) - 1}\sum_{i=1}^{m} q_i(\hat{q}_i\cdot|D| - |F_i|) \\
&= \frac{\max(\hat{h}_j)}{|D|\cdot\max(\hat{h}_j) - 1}\sum_{i=1}^{m} |D|\cdot q_i(\hat{q}_i - |F_i|/|D| + q_i - q_i) \\
&= \frac{\max(\hat{h}_j)}{|D|\cdot\max(\hat{h}_j) - 1}[\sum_{i=1}^{m}(|D|\cdot q_i(\hat{q}_i - q_i) + |D|\cdot q_i(q_i - |F_i|/|D|))\ ] \\
&= \frac{\max(\hat{h}_j)}{|D|\cdot\max(\hat{h}_j) - 1}[|D|\cdot\Upsilon + |D|\sum_{i=1}^{m} q_i(q_i - |F_i|/|D|)]
\end{aligned} \tag{18}$$

Thus,

$$\Psi = k\cdot\Upsilon + k\cdot\sum_{i=1}^{m} q_i(q_i - |F_i|/|D|) \tag{19}$$

with $k = [\max(\hat{h}_j)/(|D|\cdot\max(\hat{h}_j) - 1)]\cdot|D|$.

The relationship between Ψ and Υ implies a relationship between the error Υ, T, and T_2 in (17). It is expressed by

$$-T_2\cdot k\cdot\Upsilon = \left(-\sum_{i=1}^{m} q_i \sum_{k=1}^{T-T_2}\binom{(T-T_2)}{k}(-\hat{q}_i)^k(\sum_{j=2}^{T_2}\binom{T_2}{j}(\ (-\hat{q}_i)^j - (-r_i)^j\ ))\right) - T_2\cdot k\sum_{i=1}^{m} q_i(q_i - |F_i|/|D|) \tag{20}$$

Thus, similarly to the case of OP Testing, the relation found helps establish values of T and T_2 given a known error on the estimated profile, or to set the maximum tolerable error and the best T_2 for the OP-D testing to be better than OP testing, given a fixed budget of test cases T. Fig. 2 shows the example of Section IV, where the number of test cases T is set, and the difference $E[\omega] - E[\delta]$, as well as $E[\varphi] - E[\delta]$, is evaluated against the variation of T_2. With $T = 500$, and with both positive and negative Υ values, the best T_2 values for OP-D testing to be better than OP testing (or than *Error-free* OP testing) are those for which $E[\omega] - E[\delta]$ (or $E[\varphi] - E[\delta]$) is maximum, i.e., around $T_2 = 140 - 180$ (or $T_2 = 170 - 190$). Note also that, as the error Υ becomes larger, the range of good T_2 values gets reduced. After a given T_2 value, the curve decreases, indicating the better performance of OP testing; after that point, the decrease is rapid, and OP-D testing performs worse.

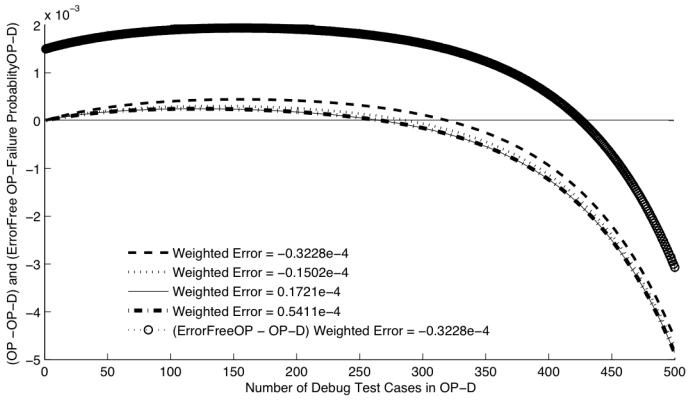


Fig. 2. Number of Test Cases for Debug Testing in the OP-D, with $T = 500$, T_2 vs. $E[\omega] - E[\delta]$, and $E[\varphi] - E[\delta]$.

Summarizing results of this section, and of Section IV, we may conclude that (6), and (14) provide the probability, respectively, *i)* of OP testing to be better or worse, from the reliability point of view, than the ideal *Error-free* OP testing, and *ii)* of OP-D testing to be better than OP testing. In comparison, (9) and (20) provide the relation between the weighted error Υ on the profile and the number of test cases, and serve to set T and T_2 to assure the best technique from a reliability perspective, given a maximum tolerable error on the profile. They may also be used to determine the range of tolerable error within which a technique is better than another, given a fixed testing budget. To have a complete picture, the impact of all the involved parameters is now evaluated by simulation.

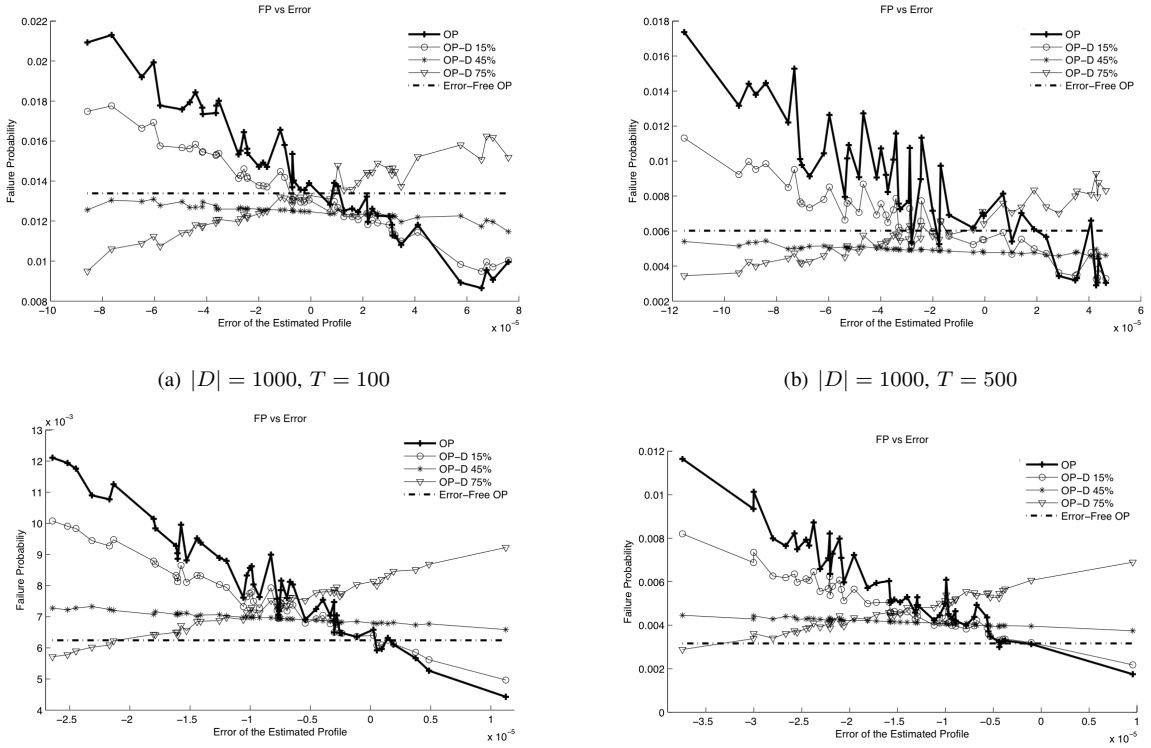
VI. Simulation

The findings of the previous sections are checked through simulation, by randomly generating several operational profiles, and evaluating the performance of techniques with respect to the involved factors.

A. Procedure

The parameters considered in the simulation are the weighted error value Υ, the *number of test cases T*, the *number of test cases devoted to debug testing* in the OP-D approach T_2, $T_2 = T - T_1$, the *cardinality of the input domain* $|D|$, and the *number of failure regions* present in the software $|F|$. Note that the weighted difference Ψ is proportional to Υ. Thus, the corresponding scenario is not showed. These parameters of interest determine a final set of four scenarios, described in the following. To carry out experiments, we implemented a procedure to accomplish the following steps.

1) Operational profile generation, by generating random numbers between 0 and 1 for each input, and then normalizing them.
2) Operational profile generation with error.
3) Debug testing profile generation, by (2).



(a) $|D| = 1000$, $T = 100$

(b) $|D| = 1000$, $T = 500$

(c) $|D| = 5000$, $T = 100$

(d) $|D| = 5000$, $T = 500$

Fig. 3. Experiment 1

4) Choice of the failing inputs by randomly picking numbers between 0 and $|D|$ for each experiment.
5) Given the limits of the parameter values to be varied in the experiment, the algorithm computes the expected failure probability of each approach.

B. Experiment 1

The first scenario deals with operational profile characteristics. In this experiment, the operational profile error is varied to observe the performance of OP and OP-D in the presence of error. To evaluate the OP-D approach, three different configurations are set for T_2: *15 %*, *45 %*, *75 %* of the total number of test cases T. For this experiment, four different configurations are evaluated: the cardinality of the input domain is $|D| = 1000$ and then $|D| = 5000$, the number of test cases is $T = 100$ and then $T = 500$. In all the four configurations, the number of failure regions is $|F| = 10$. Note that the operational profile assumed to be correct is generated in each setting, and consequently the profiles affected by error are regenerated each time. For this experiment, 50 different profiles are generated in each configuration. A profile is generated by sampling random numbers between 0 and 1 for each input (i.e., $|D|$ times), and then by normalizing each number with respect to the sum of all the generated numbers, so that the input values add to 1.

In Fig. 3, the dotted line represents the delivered failure probability resulting from *Error-free* OP testing. For each

curve (i.e., the applied testing technique), each point in the graph represents the expected failure probability (y-axis), obtained by applying the corresponding technique, considering an operational profile with a given estimation error (x-axis).

When such a value is lower than the value of the dotted line, it means that the technique performed better than *Error-free* OP testing; in these cases, the error on the profile estimation is in the conditions that we found analytically, and the observed performance is better than the ideal *Error-free* case.

In particular, if the error is greater than $f(T, \hat{Q})/T$ for OP testing, it is more probable to pick failing inputs in the testing phase than in the operational phase, and the testing performance is good. In the same way, if $\Upsilon < \{g(T, T2, \hat{Q})/(-T_2) - k \times \sum_{i=1}^{m} q_i(q_i - |F_i|/|D|)]\}/k$, then OP-D testing outperforms OP testing, and possibly the *Error-free* OP testing as well. We observe results in the left part of the graphs, where $E[\delta] < E[\omega]$, and in some cases $E[\delta] < E[\varphi]$. This condition of course happens more often when Υ is positive for OP testing, and when Υ is negative for OP-D testing. Thus, OP testing becomes better as the error Υ increases, contrarily to the OP-D approach. The specular behavior between OP and OP-D testing is observed because, when the error is negative, $\hat{q}_i$ values are on average lower than q_i ones, and at the same time r_i values are higher, making OP-D perform better.

Whereas, when the error is positive, $\hat{q}_i$ values are on average higher than q_i ones, and at the same time r_i values are lower, making OP-D perform worse.

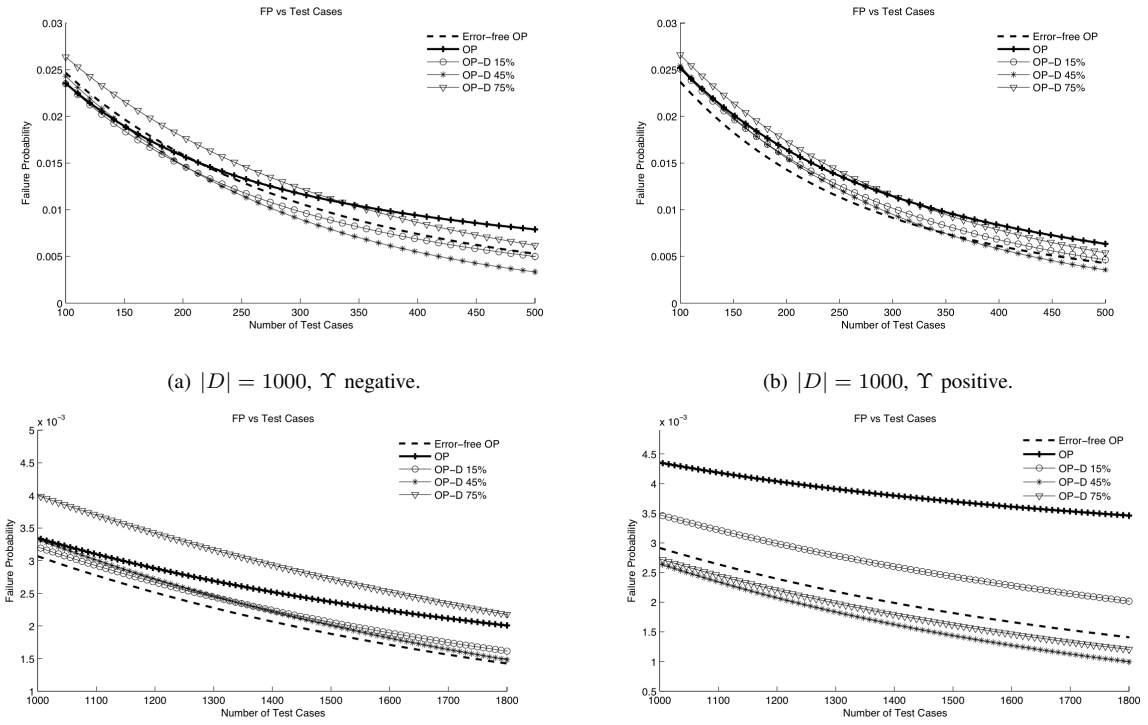
In most of the experimental points in this example, the OP-D approach exhibited a lower failure probability. OP testing behaves better when the error becomes a sufficiently large positive value. As the error approaches 0, all the experimented techniques tend to have the same performance.

One more point to highlight is that OP testing tends to have greater fluctuations than OP-D testing; combining techniques tends to limit the effect of the profile estimation error on the variation of results because, by addressing both high q_i with operational testing, and low q_i with debug testing, it balances the performance between cases with different Υ. The technique that exhibits the most stable behavior is the OP-D 45%, namely the OP-D technique with $T_2 = 45\%$ of the total number of test cases T.

C. Experiment 2

Experiment 2 compares techniques with respect to the number of test cases T. The delivered failure probability of each technique decreases with the increase of test cases, so the goal is to compare the decreasing behavior of each technique. The fixed parameters are $|D| = 1000$, and then $|D| = 5000$; considering results of Experiment 1, the error Υ is either positive or negative; and $|F| = 10$.

Figs. 4(a)-4(d) show comparable trends. In particular, regardless of the weighted error Υ, there is a number of test cases for which OP-D testing behaves better than the others; of course, as T tends to infinite, all approaches go to 0. OP testing shows the worst performance; in the first scenario (Fig. 4(a)), it starts better than the others, but as the number of test cases increases it gets worse. In most cases, OP-D is better than OP testing, in the sense that it converges earlier to 0, employing fewer test case to obtain the best performance. In particular, OP-D with 45% of T_2 exhibits the best performance. In all the cases except the one in Fig. 4(c), it goes even better than the



(a) $|D| = 1000$, Υ negative.

(b) $|D| = 1000$, Υ positive.

(c) $|D| = 5000$, Υ negative, zoomed in $T > 1000$.

(d) $|D| = 5000$, Υ positive, zoomed in $T > 1000$.

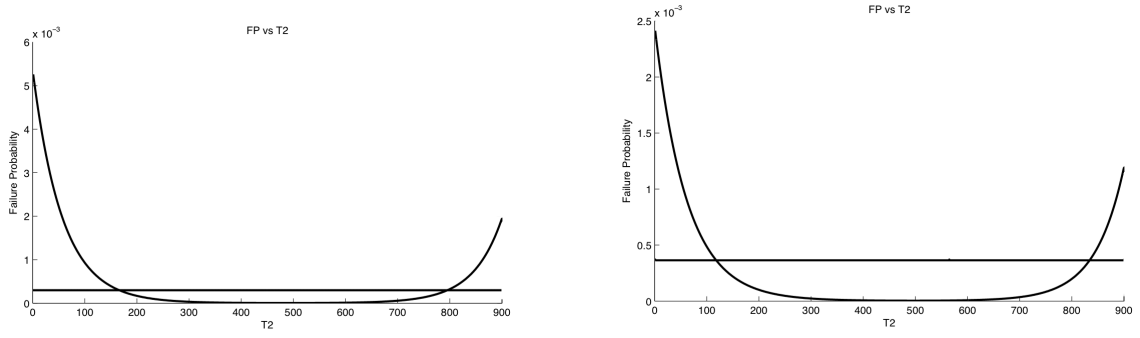
Fig. 4. Experiment 2.

Error-free case. In the scenario of Fig. 4(c), 45% OP-D is still worse at $T = 1800$; but, by extrapolating the trend, we see it should converge earlier to 0 than *Error-free* OP testing.

D. Experiment 3

In the previous experiment, we showed that the OP-D approach eventually behaves better than OP, with different convergence speed depending on the operational profile characteristics. This experiment investigates in more detail the impact of T_2, i.e., of the proportion of test cases allocated to *debug* testing, with respect to *operational* testing, in the OP-D solution. Thus, instead of considering 3 percentage values of T_2 as above, this experiment allows T_2 varying from 0% of T (which corresponds to the pure OP testing) up to 90% of T. . The fixed parameters are $T = 1000$, $|D| = 5000$, $|F| = 10$, and Υ is considered to be either positive or negative.

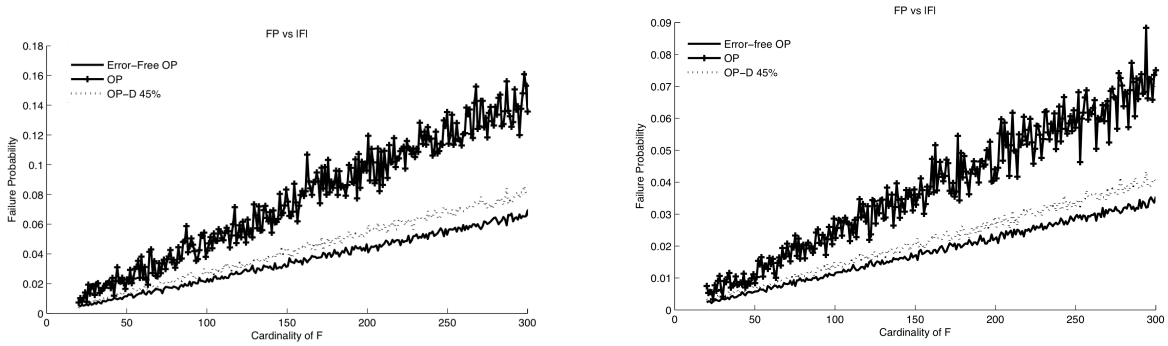
The observed behavior shown in Fig. 5 is similar to a bathtub curve, as also showed by the example in Section V. The starting value 0 represents the OP testing. The failure probability decreases suddenly as T_2 increases, and outperforms the *Error-free* OP testing (represented by the constant line) in a range, starting at about 10%-20% to 75%-85% of T. Then, besides that value, OP-D performance gets worse. The best T_2 values for Fig. 5(b) are between about 320 and 520 out of 900 test cases, i.e., between 35% and 57% of T; while in Fig. 5(a), with negative Υ, the interval is between 370, 41%, and about 600, 66%. In the case of the example in Fig. 2, the comparison between OP-D and *Error-free* OP testing suggested a T_2 value between 170 and 190 out of 500 (i.e., between



(a) $T = 1000$, Υ negative.

(b) $T = 1000$, Υ positive.

Fig. 5. Experiment 3.



(a) $T = 1000$.

(b) $T = 2000$.

Fig. 6. Experiment 4.

34% and 38% of T) for the difference $E[\varphi] - E[\delta]$ to be the maximum, confirming that the central part of the curve provides the best performance. In general, the differences in such percentages depend on the value of Υ; to compute the best T_2 given an Υ, (20) must be used.

E. Experiment 4

This experiment shows the variation of performance with respect to the number of failure regions, $|F|$, with $T = 1000$, and then $T = 2000$, also with $|D| = 5000$. The performance, shown in Fig. 6, gets worse as the number of failure regions increases; the increasing for all the techniques is approximately linear. OP testing increases more rapidly than the others; the OP-D combination performs better also in this case. The behavior of OP-D 45% testing is confirmed to be the best also in this case.

Overall, the conducted simulation suggests that the OP-D solution with $T_2 = 45\%$ performs better than the others. The experiments confirm this outcome with several parameter configurations: in Experiment 1, OP-D at 45% was evidently the most stable solution with respect to the error Υ (it is almost constant); in Experiment 2 and

Experiment 4,

the OP-D at 45% had the lowest failure probability value in all the experimented cases; Experiment 3 showed that the lowest point in the bathtub curve is in the central part, thus close to 45%.

VII. A Case Study Example

The case study chosen for illustrative purpose is an application developed for the European Space Agency called Space. It is a program to provide the user interface for the configuration of an array of antennas, and it functions as an interpreter for an array definition language (ADL). Its purpose is to prepare a data file according to a predefined format, given the array antenna configuration described in ADL. If the input ADL file is correct, Space outputs an array data file containing a list of array elements, positions, and excitations; otherwise, it outputs an error message. The program consists of 9564 lines of C code (6218 executable), and has already been widely used in several previous studies on reliability analysis, evaluation, and software testing (e.g., in [13], [9], [14], [15]).

The program is provided with 33 faulty versions containing faults discovered and recorded during testing and operation (the five additional faults found by [14] are not considered), and with 13585 test cases (each test case is an ADL file). The goal of our experiment is to compare OP-D testing against OP testing. OP-D at 45% has been chosen among the compared ones, as it showed the best performance previously.

TABLE II
Total Detected Faults and Final Reliability

Profile	Technique	Number of the test case that exposed the failure, from which a fault is removed												Total No. of Detected Faults	Rel.
$P2$ - $D2$	**OP**	1	2	3	5	8	9	10	12	15	29	64	126	**17**	**0.9904**
		177	302	488	604	932	-	-	-	-	-	-	-		
	OP-D	1	2	3	5	8	12	15	16	18	31	82	103	**21**	**0.9939**
		212	254	437	722	803	916	1115	1299	1342	-	-	-		
$P3$ - $D3$	**OP**	1	2	3	4	5	8	10	18	19	20	32	109	**16**	**0.9878**
		188	324	511	922	-	-	-	-	-	-	-	-		
	OP-D	1	2	3	4	5	8	10	11	18	19	25	48	**22**	**0.9978**
		71	198	383	627	704	712	880	945	1124	1181	-	-		
$P4$ - $D4$	**OP**	1	2	3	4	5	6	8	13	32	56	112	203	**16**	**0.9869**
		488	702	784	1009	-	-	-	-	-	-	-	-		
	OP-D	1	2	3	4	5	6	8	13	38	74	162	280	**19**	**0.9935**
		537	682	723	797	918	1226	1412	-	-	-	-			
$P5$ - $D5$	**OP**	1	2	3	4	6	7	9	23	40	71	159	190	**17**	**0.9917**
		247	301	746	-	-	-	-	-	-	-	-	-		
	OP-D	1	2	3	4	6	7	8	9	26	62	181	236	**22**	**0.9961**
		340	522	664	714	805	881	923	1060	1246	1355	-	-		

The following steps have been taken to perform the experiment.

1) We first assigned a value between 0 and 1 to each single available test case,[8]

[8]Test cases are not partitioned into equivalence classes.

representing the probability with which the considered input case would actually occur in operation. Each value is generated by a uniform random number generation in (0,1), and then normalized over the sum of all values so their sum is 1. The so-built distribution is assumed to represent the actual operational profile, denoted as $P1$.

2) Four further operational profiles have then been generated randomly in the same way, named $P2$, $P3$, $P4$, and $P5$. They are supposed to be the testing profiles for operational testing (meaning that test cases for OP testing are sampled from such profiles). From these profiles, we used (1) to create testing profiles for debug testing, named respectively $D2$, $D3$, $D4$, and $D5$.
3) After generating profiles, we run eight testing sessions: four with OP, and four with OP-D testing techniques. Specifically, we used 1500 out of the total number of test cases as the testing budget T for each testing session; we first run the 1500 test cases with the OP testing method according to operational profiles $P2$, $P3$, $P4$, and $P5$. Then, we run OP-D testing, using the same P profiles for the operational testing part (i.e., for 55% of the T = 825 test cases picked up from P profiles), followed by the corresponding debug testing profile (45% of the T = 675 test cases from D profiles).
4) During the testing sessions, failed runs have been detected by using the original Space program as the test oracle, i.e., applying test cases to the original and to the faulty Space program, revealing differences between the corresponding outputs. Because the same test case may trigger several faults, the fault removal has been performed as in [18], [13], assuming a one-to-one correspondence between failure and defect; when a software failure is observed, one and only one failure-causing defect is removed without new defects being introduced. Table II reports the number of the test case that exposed a failure, from which a (seeded) fault is removed.
5) After this phase, the reliability of the program with residual defects has been estimated by running the software under the real operational profile $P1$, by executing 2300 runs (without removing defects) selected according to the actual profile $P1$. Reliability is estimated as in [13], [16]: $R = 1 - (K/N)$, with K being the number of failed executions, and N the number of total executions, selected according to $P1$.

The final reliability is in the last row of Table II, which confirms the superiority of the OP-D approach as compared to OP testing, for the considered example.

VIII. Summary, and findings

In the previous sections, we examined operational testing by simulation, and via a case study; we attempted to address the issues hindering its adoption by practitioners. To get closer to the realistic application of this technique, several factors are included in the analysis, ranging from the number of test cases to the profile features. Most importantly, given that it is realistically difficult to avoid an error in estimating the operational profile when the technique is applied in practice, the impact of such an error on the result is formalized, and then evaluated. Software developers often distrust operational testing because they believe that a actual profile is not a meaningful concept. The introduction of the profile error measure and its relation with the number of test cases is essential to address the issue of inaccurate operational profile estimation.

Besides the analysis of operational (OP) testing, the paper proposed a new approach to testing software with the goal of improving its reliability. The combination of operational testing with a *debug testing* technique, namely the anti-profile debug testing, is presented; and its performance is analyzed with respect to OP testing. The main findings of these analyses are the following.

- OP testing affected by error in the profile estimation is different from *Error-free* OP testing, as expected. But it may perform better than the error-free case.
- The new conceived approach, OP-D testing, is probabilistically better than OP testing; numerically, and by simulation, it has been shown to behave better in most of cases.
- It has been possible to establish the analytical conditions in which actual operational testing (namely, OP testing with error) is better than *Error-free* OP testing, and in which OP-D testing is better than OP testing, depending on the error on the profile, and on the parameters related to the number of test cases (T, T_2).
- In general, the exact amount of test cases in the OP approach, and of operational or debug test case proportion in the OP-D approach must be decided by means of the analytical formulas from Section IV and V, depending on the expected, or the tolerable, error on the profile, on the desired performance, and on the budget in terms of test cases.
- By simulation, we found that the OP-D solution is better than OP testing in most of the tried settings, and that OP-D with 45% of the debug test cases is the best approach among those tried (15%, 45%, 75 %); it is more robust to operational profile estimation error, and delivers always the least failure probability at the end of testing. Such results are a confirmation of analytical findings.
- By simulation, OP-D testing shows a lower variability in the results than OP testing (and OP-D testing with 45 % of T_2 shows the lowest one),

  probably because combining operational and debug test cases assures, in the average, to address always both failures with high occurrence frequency at operational time, and failures with low-occurrence frequency. Variability of performance is also regulated by the cardinality of the input domain (the larger it is, the more stable the results).
- Experimentally, the application to a case study confirmed the good results of 45% OP-D as compared to OP testing in terms of delivered reliability. However, care must be taken in generalizing the observed behavior. In fact, *i)* the number of trials (i.e., of testing sessions) is not sufficient to claim a statistically significant superiority, and *ii)* choosing a case study implies that results are not generalizable a priori, so different results may be observed applying the approach to software programs with different features, application targets, size, and complexity. Moreover, in general, it may not be true that each test failure deterministically causes one fault to be removed, and the perfect oracle assumption (i.e., all test failures are noticed) may not hold. The experiment must therefore be viewed more as an example to confirm both analytical and simulation results on a concrete case study rather than as a statistically valid comparison between the techniques generalizable to other programs.

IX. Conclusion

Evaluating the impact of software testing techniques on delivered reliability is not common. There are indeed difficulties in this evaluation, related to nature of these attributes. This difficulty explains also why reliability is rarely adopted as pilot criterion to drive verification activities, or to select testing techniques. Operational testing, in spite of the encountered difficulties and its limits, is still the technique generally considered as oriented toward reliability, in contrast to debug testing which aims to maximize the number of detected bugs. Starting from operational testing, from its advantages and shortcomings, the paper tries to address again the topic of *reliability-driven* testing, i.e., it tries to figure out how to conceive testing approaches oriented toward a reliability improvement objective.

Besides the obtained results, it is important to remark that, different from the past, our aim has not been to claim the superiority of operational over debug testing (or the opposite), because, as several studies showed so far, it depends on the context (e.g., it depends on operational profile and testing profile characteristics, on the profile estimation error, and on the occurrence rate of each failure). Our attempt has been to act on the limits of operational testing for high-reliable demands, and combine debug and operational testing for getting high delivered reliability. The objective of the proposed OP-D technique is to improve reliability with respect to applying only operational testing, while uncovering, at the same time, as many bugs as possible, as pursued by debug-testing. From this perspective, we believe that the view of operational testing compared against debug testing may in some sense be overcome. The final goal of any quality assurance activity is to deliver a program not failing in operation, i.e., highly reliable. From an operational tester perspective, the program may also contain bugs, but these must not be activated at runtime.

However, for assuring this goal, operational testing should work perfectly (under the oracle of exact estimate of operational profile). On the other hand, from a debug tester perspective, the same goal may be achieved by trying to expose failures and remove as many failure regions as possible. But not considering the actual occurrence frequency of failures at operational time may bring very little reliability improvement. These two views seem counter to each other, but they aim at the same objective, and may be suitably combined in their respective advantages: *trying to expose higher-occurrence failure regions*, and *trying to expose as many failure regions as possible*. For sure, for ultra-reliable systems, testers need to do both; thus, combining them makes sense. We intend to pursue this combination, further promoting the concept and practice of reliability-driven testing.

Acknowledgements

This work has been supported by MIUR under Project PON02_00485_3487758 “SVEVIA” of the public-private laboratory “COSMIC” (PON02_00669). The work of Dr. Pietrantuono is supported by the project Embedded Systems in Critical Domains (CUP B25B09000100007) within the framework of POR Campania FSE 2007-2013.

References

[1] A. Wood, “Software reliability growth models, ” Tandem Computers Inc. Technical Report 96.1, 1996.

[2] V. Almering, M. Van Genuchten, G. Cloudt, and P.J.M. Sonnemans, “Using software reliability growth models in practice,” *IEEE Software*, vol. 24, no. 6, pp. 82-88, 2007.

[3] E. N. Adams, “Optimizing preventive service of software products, ” *IBM J. Research and Development*, vol. 28, 1, pp. 2-14, 1984.

[4] P.G. Frankl, R.G. Hamlet, B. Littlewood, and L. Strigini, “Evaluating testing methods by delivered reliability, ” *IEEE Transactions on Software Engineering*, vol. 24, no. 8, pp. 586-601, 1998.

[5] B. Beizer, “The Cleanroom process model: a critical examination, ” *IEEE Software*, vol. 14, no. 2, pp. 14-16, 1997.

[6] R. Chillarege, I.S. Bhandari, J.K. Chaar, M.J. Halliday, D.S. Moebus, B.K. Ray, and M.Y. Wong, “Orthogonal defect classification-a concept for in-process measurements, ” *IEEE Transactions on Software Engineering*, vo. 18, no. 11, pp. 943-956, 1992.

[7] B. Zachariah, and R. N. Rattihalli, “Failure size proportional models and an analysis of failure detection abilities of software testing strategies, ” *IEEE Transactions on Reliability*, vol. 56, no.2, pp. 246-253, 2007.

[8] J. D. Musa, “Operational profiles in software-reliability engineering, ” *IEEE Software*, vol. 10, no. 2, pp.14-32, 1993.

[9] C.G. Bai, C.H. Jiang, and K.Y. Cai, “A reliability improvement predictive approach to software testing with bayesian method, ” *Proc. of the Chinese Control Conference*, pp. 6031-6036, 2010.

[10] P. Frankl, D. Hamlet, B. Littlewood, and L. Strigini, “Choosing a Testing Method to Deliver Reliability, ” *Proc. of the 19th International Conference on Software Engineering*, pp. 68-78, 1997.

[11] K.Y. Cai, “Towards a conceptual framework of software run reliability modeling, ” *Information Sciences–Informatics and Computer Science: An International Journal*, vol. 126, no. 1-4, pp. 137-163, 2010.

[12] K.S. Trivedi, “Probability and statistics with reliability, queuing and computer science applications (2nd edition ed.), ” John Wiley and Sons Ltd., Chichester, UK., 2001.

[13] A. N. Crespo, P. Matrella, and A. Pasquini, “Sensitivity of reliability growth models to operational profile errors, ” *Proc. of the 7th International Symposium on Software Reliability Engineering*, pp. 35-44, 1996.

[14] G. Rothermel, R.H. Untch, C. Chu, and M.J. Harrold., “Prioritizing test cases for regression testing, ” *IEEE Transactions on Software Engineering*, vol. 27, no. 10, pp. 929–948, 2001.

[15] F. I. Vokolos, and P.G. Frankl, “Empirical evaluation of the textual differencing regression testing technique, ” *Proc. of the International Conference on Software Maintenance*, pp. 44–53, 1998.

[16] R. Pietrantuono, S. Russo, K.S. Trivedi, “Software Reliability and Testing Time Allocation: An Architecture-Based Approach”, *IEEE Transactions on Software Engineering*, vol. 36, no. 3, pp. 323-337, 2010.

[17] K.Y Cai, C.H. Jiang, H. Hu, and C. Bai, “An experimental study of adaptive testing for software reliability assessment”, *Journal of Systems and Software*, vol. 81, no. 8, pp. 1406-1429, 2008.

[18] K.Y. Cai, D. B. Hu, C. Bai, H. Hu, and T. Jing, “Does software reliability growth behavior follow a non-homogeneous Poisson process, ” *Information & Software Technology*, vol. 50, no. 12, pp. 1232-1247, 2008.

[19] E. Weyuker, “Comparing the Effectiveness of Testing Techniques, ” *Formal Methods and Testing, Robert M. Hierons, Jonathan P. Bowen, and Mark Harman (Eds.). Lecture Notes In Computer Science*, vol. 4949, 271-291, 2008.

[20] J.W. Duran, and S.C. Ntafos, “An evaluation of random testing, ” *IEEE Transactions on Software Engineering*, vol. 10, no. 7, pp. 438-444, 1984.

[21] D. Hamlet, and R. Taylor, “Partition testing does not inspire confidence,” *IEEE Transactions on Software Engineering*, vol. 16, no. 12, pp. 1402-1411, 1990.

[22] P.G. Frankl, and E.J. Weyuker, “A formal analysis of the fault detecting ability of testing methods, ” *IEEE Transactions on Software Engineering*, vol 19, no. 3, pp. 202-213,1993.

[23] D. Hamlet, “Are We Testing for True Reliability?, ” *IEEE Software*, vol. 9, no. 4, pp. 21-27, 1992.

[24] M. Chen, A.P. Mathur, and V.J. Rego, “Effect of testing techniques on software reliability estimates obtained using a time-domain model, ” *IEEE Transactions on Reliability*, vol. 44, no. 1, pp. 97-103, 1995.

[25] P. Frankl, and Y. Deng, “Comparison of delivered reliability of branch, data flow and operational testing: a case study, ” *Proc. of the International Conference on Software Testing and Analysis*, pp. 124-134, 2000.

[26] S. Poulding, and J.A. Clark, “Efficient Software Verification: Statistical Testing Using Automated Search, ” *IEEE Transactions on Software Engineering*, vol. 36, no. 6, pp. 763-777, 2010.

[27] J.A., Whittaker, and M.G. Thomason, “A Markov chain model for statistical software testing, ” *IEEE Transactions on Software Engineering*, vol. 20, no. 10, 1994.

[28] C. Kallepalli, and J. Tian, “Measuring and modeling usage and reliability for statistical Web testing, ” *IEEE Transactions on Software Engineering*, vol. 27, no. 11, pp. 1023-1036, 2001.

[29] P. Thevenod-Fosse, and H. Waeselynck, “An investigation of statistical software testing, ” *Software Testing, Verification and Reliability*, vol. 1, no. 2, pp. 5-26, 1991.

[30] J. D. Musa, “Software-reliability-engineered testing, ” *Computer*, vol. 29, no. 11, pp. 61-68, 1996.

[31] H.D. Mills, M. Dyer, and R.C. Linger, “Cleanroom software engineering, ” *IEEE Software*, vol. 4, no.5, pp.19-24, 1987.

[32] R.W. Selby, V.R. Basili, and F.T. Baker, “Cleanroom software development: an empirical evaluation, ” *lEEE Transactions on Software Engineering*, vol. 13, no. 9, pp. 1027-1037, 1987.

[33] P. A. Currit, M. Dyer, and H. D. Mills, “Certifying the reliability of software, ” *IEEE Transactions on Software Engineering*, vol. SE-12, no. 1, pp. 3-11, 1986.

[34] R.H. Cobb, and H.D. Mills, “Engineering software under statistical quality control, ” *IEEE Software*, vol. 7, no. 6, pp. 44-54, 1990.

[35] R.C. Linger, and H.D. Mills, “A case study in cleanroom software engineering: The IBM Cobol structuring facility, ” *Proc. of the Computer Software and Application Conference*, pp. 10-17, 1988.

[36] J.H. Poore et al., “A case study using cleanroom with box structures ADL, ” Software Engineering Technology Technical Report CDRL 1880, 1990.

[37] M. R. Lyu (Ed.), “Handbook of software reliability engineering, ” IEEE Computer Society Press and McGraw-Hill, 1996.

[38] L. Strigini, and B. Littlewood, “Guidelines for statistical testing, ” (Report No. PASCON/WO6-CCN2/TN12). ESA/ESTEC project PASCON, 1997.

[39] L. Madani, C. Oriat, I. Parissis, J. Bouchet, and L. Nigay, “Synchronous testing of multimodal systems: an operational profile-based approach, ” *Proc. of the16th IEEE International Symposium on Software Reliability Engineering*, pp. 325-334, 2005.

[40] T. Y. Chen, F.-C. Kuo, and H. Liu, “Application of a failure driven test profile in random testing, ” *IEEE Transactions on Reliability*, vol. 58, no. 1, pp. 179-192, 2009.

[41] T. Y. Chen, F.-C. Kuo, and H. Liu, “Distributing test cases more evenly in adaptive random testing, ” *The Journal of Systems and Software*, vol. 81, no. 12, pp. 2146-2162, 2008.

[42] K.Y. Cai., “Optimal software testing and adaptive software testing in the context of software cybernetics, ” *Information and Software Technology*, vol 44, no. 14, pp. 841-855, 2002.

[43] K.Y. Cai, B. Gu, H. Hu, and Y.C. Li, “Adaptive software testing with fixed-memory feedback, ” *Journal of Systems and Software*, vol. 80, no. 8, pp. 1328-1348, 2007.

[44] K.Y. Cai, Y.C. Li, and K. Liu, “Optimal and adaptive testing for software reliability assessment, ” *Information & Software Technology*, vol. 46, no. 15, pp. 989-1000, 2004.

[45] G. J. Myers, “The art of software testing, ” Wiley, New York, 1979.

[46] C. Sarbu, A. Johansson, N. Suri, and N. Nagappan, “Profiling the operational behavior of OS device drivers, ” *Proc. of the 19th IEEE International Symposium on Software Reliability Engineering*, pp. 127-136, 2008.

[47] C. Trammell, “Quantifying the reliability of software: statistical testing based on a usage model, ” *Proc. of the 2nd IEEE International Software Engineering Standards Symposium*, pp. 208-218, 1995.

[48] M. Riebisch, I. Philippow, M. Gotze, “UML-Based statistical test case generation, ” *Objects, Components, Architectures, Services, and Applications for a Networked World*, vol. 2591, pp. 394-411, 2003.

[49] J. D. Musa, “Sensitivity of field failure intensity to operational profile errors, ” *Proc. of the Fifth Int. Symposium on Software Reliability Engineering*, pp. 330-333, 1994.

[50] O. J. Silva, A. N. Crespo, M. L. Chaim, and M. Jino, “Sensitivity of two coverage- based software reliability models to variations in the operational profile”, *Proc. of the 4th International Conference on Secure Software Integration and Reliability Improvement*, pp. 113-120, 2010.

[51] M. H. Chen, A. P. Mathur, e V. Rego, “A case study to investigate sensitivity of reliability estimates to errors in operational profile, ” *Proc. of the Fifth Int. Symposium on Software Reliability Engineering*, pp. 276-281, 1994.

[52] C.Y. Huang, and M. R. Lyu, “Optimal testing resource allocation, and sensitivity analysis in software development”, *IEEE Transactions on Reliability*, vol. 54, pp. 592-603, 2005.

Domenico Cotroeno received his Ph.D. in 2001 from the Department of Computer Science and System Engineering at the University of Naples "Federico II". He is currently associate professor at the same university. His main interests include dependability assessment techniques, software fault injection, and field-based measurement techniques. Domenico Cotroneo has served as Program committee member in a number of scientific conferences on dependability topics, including DSN, EDCC, ISSRE, SRDS, and LADC; and he is involved in several national and European projects in the context of dependable systems.

Roberto Pietrantuono, Ph.D., IEEE Member, is currently a post-doc at University of Naples "Federico II", Italy. He received his Ph.D. degree (2009) in Computer and Automation Engineering from the same university. He collaborates with several companies of the Finmeccanica group, in the field of critical software system development. His research interests are in the area of software reliability engineering, particularly in software verification of critical systems, software testing, and software reliability analysis.

Stefano Russo is a Professor and the Deputy Head at the Department of Computer and Systems Engineering, Federico II University of Naples. He is the Chairman of the Curriculum in Computer Engineering, and Director of the "C. Savy" Laboratory of the National Inter-Universities Consortium for Informatics (CINI). His research interests are in the areas of distributed software engineering, middleware technologies, and dependable software systems. He coordinates the national project DOTS-LCCI on software dependability for critical infrastructures. He has (co-)authored more than 120 scientific papers.