Probabilistic sampling-based testing for accelerated reliability assessment

Roberto Pietrantuono
Università degli Studi di Napoli Federico II
Via Claudio 21, 80125 Naples, Italy
Email: roberto.pietrantuono@unina.it

Stefano Russo
Università degli Studi di Napoli Federico II
Via Claudio 21, 80125 Naples, Italy
Email: stefano.russo@unina.it

***Abstract*—A relevant objective of software reliability assessment is to get unbiased estimates with an acceptable trade-off between the number of tests required and the variance of the estimate. A low variance is desirable to increase the confidence in the estimate, but too many tests may be required by conventional reliability assessment testing techniques based solely on the operational profile.**

This article presents *probabilistic sampling-based testing*, a new technique using unequal probability sampling to exploit auxiliary information about the software under test so as to assess reliability unbiasedly and efficiently. The technique expedites the assessment process assuming the availability of some prior belief about input regions failure proneness. The evaluation by simulation and experimentally shows promising results in terms of estimate accuracy and efficiency.

***Index Terms*—Reliability assessment, SRGM, random testing, operational testing, reliability testing, debug testing.**

I. Introduction

In several contexts where software is of primary concern, being able to confidently know its reliability is a key to decision making. Reliability provides a measure of the user-perceived quality in operation, thus its faithful assessment is strategic in business-critical systems and even vital in mission- or safety-critical ones.

Testing for reliability assessment exercises the software with the goal of getting an estimate of expected reliability before delivery. A common usage is during acceptance testing, where the reliability estimate is adopted as criterion to accept/reject a product for release. A conventional technique is operational testing, where tests try to mimic the real usage in operation and failure data are used to estimate reliability unbiasedly [1]. The hard challenge of this approach is that too many tests are usually required to get an estimate with an acceptable confidence [2], [3]. In other words, the efficiency of the estimator (i.e., a low estimator's variance under a given number of tests) is a big hurdle to the conventional operational testing adoption. This is especially true for high-reliability systems, where, at acceptance testing stage, there are few and low-occurrence residual failing inputs, which operational testing, naturally targeting higher-occurrence inputs, will hardly expose [3].

Attempts have been done to overcome this limitation, e.g., by means of adaptivity [4], [5], with improvements over conventional operational testing. In our previous works [5], [6], we argued that the selection of test cases is not strictly required to adhere to the operational profile, since when a high reliability is achieved by operational testing it is convenient to look for low-occurrence failing inputs, as high-occurrence ones are already exposed (namely, operational testing achieves its saturation). In this sense, operational and debug testing (i.e., testing oriented to detect bugs regardless their expected occurrence probability) should be combined. Ideally, to be efficient, the *testing profile* should favour those inputs with the *highest expected unreliability contribution*, that are those inputs with a high product between the operational occurrence probability and the probability that the selected input is failing [7]. This should be done while preserving the estimation unbiasedness.

In this work, we propose *Probabilistic Sampling-based* Testing (PST), a technique to enable an unbiased and efficient assessment of reliability. We advocate the use of *probabilistic sampling* with unequal selection probabilities to define testing profiles deliberately different from the operational profile, and conceived to provide a more efficient (and unbiased) estimate of reliability. The idea is to combine the benefit of testing profiles aimed at maximising the exposure of failures (hence, finding many faults) with the benefit of the operational testing profile, aimed at selecting "representative" test cases to have an unbiased reliability estimate. This approach expedites the assessment process (namely, it requires fewer tests to get a desired variance, hence confidence).

Probabilistic testing with unequal selection probabilities is grounded on the concept that, at acceptance testing time, a tester usually has some auxiliary information about the relative unreliability of portions of the input space (e.g., components, modules, partitions), which can be very useful to minimise the variance of the estimation. For instance, in partition-based testing, the simple belief about equivalence classes (e.g., boundary values are expected to fail more often than in-range value classes) is used to select test cases. This is a kind of information that is worth to be exploited for reliability assessment too. Quantitative evidence can be also built from failure data at component-level collected during development testing or from previous releases. Additional evidence collected during module-level or integration testing (e.g., achieved coverage, defect detection/correction trend) are further examples of information contributing to form the tester's belief, like discussed in many papers proposing

Bayesian inference to formalise it [2], [8], [9]. In operational testing, this information is ignored, and the selection is done solely based on the expected operational profile. The idea of PST is to use the auxiliary information in the test selection procedure, combined with the operational profile expectation, to sample tests contributing more to assess unreliability at minimal variance. We formulate PST as a sampling problem, wherein several sampling strategies can be implemented depending on the input space model and the auxiliary information being available. The output is an estimate of reliability and of its variance much more efficient than testing schemes based only on the operational profile.

Evaluation is performed by comparing the assessment *accuracy* and *efficiency* against two competing techniques. Evaluation is conducted both by means of a simulation study, and by means of experimentation on a widely-used large software application, namely the MySQL DBMS. As auxiliary information gained before acceptance testing, we have tested the integration of PST with the usage of software reliability growth models (SRGMs), applied on MySQL components' historical failure data. For the selected case study, we instantiated the technique by a *without-replacement sampling algorithm*, which is known to be more efficient than the with-replacement counterpart, exploiting the auxiliary information by SRGMs via *probability-proportional-to-size* (PPS) sampling. Results show that PST outperforms competing techniques, both under the assumption of uniform belief about components reliability (i.e., ignorance of any auxiliary information) and with the SRGM-based beliefs. The latter is, expectedly, shown to be more performant.

The rest of the paper is organised as follows: Section II surveys the research related to reliability assessment via testing; Section III defines the PST strategy; Section IV presents the evaluation design and setup; Sections V and VI report the results of the simulation and experimentation study, respectively; Section VII discusses some threats to validity, and Section VIII closes the paper.

II. Related work

Software reliability assessment via testing is a wide research area. The problem of how to provide faithful reliability estimates with an acceptable testing effort before release is known to be a big challenge for both business- and safety-critical systems. Operational testing, where a testing profile is derived in accordance with the expected operational profile, has always been a reference technique for software reliability engineering practitioners. Since the eighties, operational testing is seen as a means to certify the software against a given mean time to failure (MTTF), e.g., in the context of Cleanroom software engineering [10], finding a noticeable popularity [11], [12], [13], [14]. Few years later, Musa proposed the well-known Software Reliability Engineering Test process [1], in which operational testing was a core element. He also found very relevant results in favor of operational testing at AT&T, e.g., claiming a reduction of a factor-of-10 in customer-reported problems [15]. Further evidence in favor of operational testing was reported in [3], [16].

Nonetheless, operational testing was also hardly criticised. For instance, Beizer strongly argued that the Cleanroom process, wherein operational testing is a key technique, was never compared fairly with other approaches, and that abandoning debug testing in favour of operational testing would be a wrong choice [17]. A raised criticism was that operational testing cannot deal with low-occurrence failures, and thus it cannot replace at all debug testing. Indeed, once operational testing exposes the higher-occurrence failures (for which it is particularly suitable), then it would expose few further failures (i.e., it achieves a saturation), making it hard to get a tighter-variance reliability estimate because of few failing samples.

More recent work tried to improve this negative aspect of operational testing, through a partition-based approach and especially through adaptation. Cai et *al.* published several papers on *Adaptive Testing*, based still on operational profile but foreseeing adaptation in the assignment of test cases to input partitions [4], [18], [19]. The authors formulate testing as an adaptive control problem using controlled Markov chains, with the goal of minimising the variance of reliability estimator. In [20], it is used along with a gradient descent method to the same aim, while in [21], it exploits confidence intervals as driving criterion to select tests adaptively. In our previous work [6], we claim that operational and debug testing are not necessarily in contrast with each other, since they are two different ways of getting to the same objective, and may be suitably combined in their respective advantages: trying to expose higher-occurrence failures (operational testing), and trying to expose as many failures as possible (debug testing). We pursued that direction, and proposed a testing strategy, named RELAI (RELiability Assessment and Improvement) [5], where adaptiveness is exploited to have the benefit of operational testing at the beginning (when high-occurrence failures can still be found) and progressively moving toward low-occurrence failures. In those works, we have explored the possibility to use a testing profile different from the operational profile, while still providing an unbiased estimate but in a faster way (i.e., with fewer test cases).

Besides improving the estimates by exposing more failures, further problems regard the sampling and estimation technique itself. In fact, test selection is typically done either according to the operational profile to assure an unbiased estimate (with the discussed limitations in terms of failures exposure and thus large-variance estimate), or foresees a partition-level profile coupled with the basic simple random sampling with replacement (SRSWR) within partitions. Few approaches go beyond this. In [22], authors adopt stratified sampling, by stratifying executions by cluster analysis applied to execution profiles, and then sampling within strata without replacement, which is known to be more efficient than the with-replacement counterpart. In [23], stratified sampling is still proposed combined with symbolic execution to stratify profiles. In our last recent work [24], we stressed this point, and presented a family of sampling-based algorithms to be applied at subdomains

level depending on knowledge (or belief) that testers gain about the subdomain characteristics during development. A theoretical analysis was conducted to enable the usage of more complex sampling strategies than SRSWR. Exploiting a prior belief is a point also made by the theory of software reliability corroboration [2], [8], where a Bayesian approach is used that considers operational testing as a means to corroborate (rather than to assess) reliability, by complementing evidences already gained in previous phases – a problem particularly felt in ultra-reliable systems, where no failures are observed during testing, making operational testing not able, by itself, to give confidence about reliability.

The work presented here aims at exploiting the same belief that tester would use for defining a "debug testing" profile and/or any quantitative evidence about failure proneness of input subdomains in order to drive test selection toward exposing failures most impacting reliability. As specific instance, an algorithm is developed that considers the whole domain-level input space, rather than combining subdomain-level estimates, and quantitative evidence from past failure data. A promising combination with software reliability growth models (SRGMs) as source of auxiliary information is implemented and tested. It is showed how a source of (un)reliability evidence gained before the final testing stage, even though loosely correlated with reliability, can expedite the assessment. As consequence, approaches to quantify reliability before acceptance testing, such as Bayesian inference, architecture-based, early stage assessment techniques [25], [9], [26], [27], [28], [29], [30], [31], [32], can be seamlessly integrated with what proposed hereafter.

III. Probabilistic Sampling-based Testing

In the following, we formulate the problem and describe the idea of exploiting unequal sampling techniques and estimators. Then, we instantiate the technique with one specific sampling design, deriving the reliability estimator, its real variance and the estimate of the variance.

A. Overview

Let us denote with D the input space, namely the set of all input points that can be given to the system under test. Assume the system can be decomposed into M *independently testable units*, each with its own input domain $D_1, \dots D_m$. These units can be thought in several ways: a subsystem, a component, a module, or a partition in partition-based testing. In the following, we refer to them as *subdomains*. The following assumptions, typical of reliability assessment testing studies (e.g., [5] [18]-[21]), are made:

1) A test case leads to failure or success; we are able to determine when it is successful or not (perfect oracle).
2) The code is not modified during testing (i.e., it is frozen). Code can be modified and detected faults can be removed after the assessment.
3) Test case runs are independent; i.e., all the non-executed test cases are admissible each time. The execution of a test case is not constrained by the execution of some other test case before. This affects the way in which a "test case" is defined, since, if the assumption is not met, a set of tasks can be grouped together in a single test case, so that at the end of the test case the system goes back to the initial state [20].
4) The output of a test case is independent of the history of testing; in other words, a failing test case is always such, independently from the previously run test cases.
5) The operational profile P can be described as a probability distribution over the input domain D. With respect to the knowledge of P, PST can consider each inputs either singularly or grouped by classes with similar characteristics (e.g., all inputs of D_i assumed with the same occurrence probability because of lack of such a fine-grain knowledge) – namely, it does not make assumption on the level of granularity of the operational profile description, although the accuracy of such knowledge impacts the results. Profile, for the purpose of this study, is assumed to be known, like in most related literature [4], [18]-[21]: we dealt with partial knowledge of the profile in our previous work [5], whose Montecarlo-based approach can be integrated in what presented here.

Given a fixed number of test cases T provided as budget, the PST aims at unbiasedly estimate reliability R. Reliability is defined as in [7]:

$$R = 1 - \Phi = 1 - \sum_{t \in D} p_t z_t \tag{1}$$

where Φ denotes the failure probability, z_t is 1 if the input t is a failure point, 0 otherwise, and p_t is the probability of selecting the input t at runtime. An unbiased estimate of R is pursued by any reliability assessment testing strategy: the main challenge is to provide an *efficient* estimate (i.e., with low variance), while preserving unbiasedness.

PST is based on the idea that at reliability assessment stage – usually the last stage before release, such as at acceptance testing – information about the relative failure proneness of subdomains is available either as quantitative evidence or at least as tester's belief. For instance:

- If the tested units corresponds to partitions in partition-based testing, the partitioning criterion is itself an example of belief of tester, who judges some ranges of values more prone to failure while others are deemed correct. It is constitutive of partitioning to assume that inputs within a partition have a homogeneous failing behaviour, and the partitioning criterion establishes this assignment. For instance, in traditional equivalence partitioning boundary values are usually expected to fail more often than in-range values. A similar concept applies for defining the "choices" within categories in category-partition testing [33]. The idea of PST is to exploit such a belief not only for fault detection during development-time testing, but also for reliability assessment during acceptance testing.
- If the tested units corresponding to D_i are components in a component-based system, then the observed *failure data* during development testing, or during the operational

phase of previous releases, are a source of knowledge to exploit. In particular, inter-failure times can be used to build *software reliability growth models* (SRGMs) for the components under test. Using SRGMs in combination with reliability assessment testing is a practice also foreseen in the well-known SRET process outlined by Musa [1] in 1996[1]. Such an approach allows quantifying the failure-proneness belief by means of the estimation of the failure intensity (consequently, of expected failure probability) of each component.

- When the tested units are software modules, then results of module-level testing (e.g., detected/corrected defects, level of coverage, amount of testing or, generally, V&V effort) are informative about their quality.
- Other examples of information contributing to form the tester's belief are discussed in several papers proposing Bayesian inference to formalise and quantify the belief [2], [8], [9], such as code characteristics (e.g. complexity metrics are often used as predictor for defect proneness by machine learning [34]), domain expert opinion, characteristics of the testing and of development process.

PST uses this auxiliary information combined with the operational profile expectation in an unequal probability sampling design to select tests most impacting reliability. The *sampling design* establishes which (combination of) sampling techniques, within the family of probabilistic sampling, is better to use for the particular input domain of interest. Thus, the specific PST algorithm will depend on: *i)* the input space (inputs are modelled as 0/1 values, denoting correct/failing inputs, respectively), and on *ii)* the information available about failure proneness and profile. For instance, as for the input space: if the input domain can be easily split in homogeneous subdomains (i.e., with low intra-group variance) and so that the variance between subdomains (i.e., inter-group variance) is high, then *stratification with unequal sampling probability of strata* and *with replacement* (to allow multiple tests for each subdomain) is a good sampling strategy. Instead, if stratification is not advisable, *unequal probability sampling* of single inputs is preferred [35]. In such a case, *without-replacement selection* is better, even though its mathematical treatment is more complex, because it is known to be more efficient than with-replacement schemes. Generally, *unequal probability sampling* is the required underlying framework in all the cases, as it allows having selection probabilities deviating from the operational profile (hence, integrating any testing profile in the sampling strategy) while preserving unbiasedness and improving efficiency.

Regarding the available information, its granularity is of interest for defining the auxiliary variable that supports unequal probability sampling. Indeed, rarely the failure proneness belief and the operational profile estimate are available at single-input level; realistically, they are at subdomain level (partition, module, component). Hence, the auxiliary variable will be more often defined using subdomain-level information, and ignorance within subdomains is modelled assuming all inputs being equal in terms of both failure proneness belief and operational usage probability. Different choices are possible if information is at single-input level, or at (multiple) class-level within subdomains (a case suitable for *multi-stage sampling* strategies).

[1]In SRET, the SRGM are foreseen during the so-called development testing, when bugs are detected and also removed and, consequently, reliability grows; this is then complemented by reliability assessment testing to certify achieved reliability before shipping, through conventional operational testing

B. PST with SRGMs and PPS

In the following, we describe the PST algorithm used in the rest of this work – which is a specific instance suitable for component-based systems. As sampling design, the algorithm implements a *probability-proportional-to-size* (PPS) and *without-replacement* sampling of single inputs of an input domain D. As auxiliary information, it assumes the availability of failure data (e.g., stored by an issue tracking system) collected during development-time testing or operational usage of previous versions. This information is exploited to build an SRGM for each component, whose output is informative about the expected components' failure probabilities. There are many types of SRGMs available in the literature. The common goal of SRGMs is to predict reliability and this is usually done by means a *failure intensity* function λ(t) = $\frac{dm(t)}{dt}$, with $m(t)$ being the cumulative number of detected (or corrected) faults estimated by the SRGM. Although no assumption is made about which one to use, a debug-aware model is preferred, since it captures the real growth of reliability as consequence of a fault correction action, and does not assume an unrealistic immediate debugging [36]. Given the $m(t)$ or $\lambda(t)$ function at testing time t, both *testing reliability* (i.e., assuming to keep on testing, with $\lambda(t)$ still changing over time) and *operational reliability* (i.e., assuming the software is released at time t, hence $\lambda(t)$ assumed to be constant) can be estimated [37][38]. In a typical scenario, reliability assessment is needed after debug testing, namely at acceptance testing stage just before release. Therefore, we are interested in the latter case, and take the $\lambda(t_{end})$ = λ, with t_{end} representing the last detection (or correction) time of the debug testing stage. This represents the 'most recent" estimate of failure intensity before acceptance testing and release. Operational reliability is: $R(\tau|t_{end}) = exp(-\lambda\tau)$, with τ being the operational execution time. Hence, failure probability estimated by the SRGM is: $\vartheta(\tau|t_{end}) = 1 - R(\tau|t_{end})$. What actually discriminates the more or less reliable components is the failure intensity of components. Therefore λ_i values ($i = 1$ to n represents one out of n components) or a function thereof (e.g., R_i, ϑ_i, $MTTF_i$) can be seamlessly used as auxiliary information to drive test selection as explained hereafter. Without loss of generality, let us assume to use $\vartheta_i = \vartheta_i(\tau|t_{end})$, which represents explicitly the failure probability, computed under the same amount of τ time units for all components; e.g., for simplicity: $\tau = 1$ and $\vartheta_i = 1 - exp(-\lambda_i))$.

Now, consider the quantity to estimate: $\Phi = \sum_{t\in D} p_t z_t$, namely the total failure probability. The following two steps are distinguished:

1) In the first step we are concerned with the construction of a testing profile, namely with the assignment of probabilities of input selection during testing (denoted as π_t). The goal of this step is to assign higher selection probability to those inputs whose expected contribution to the total unreliability in operation will be higher. This *unreliability contribution* is given by the product of the probability of selection in operation, p_t, and the expected probability of failing once selected, Pr(z_t=1).
2) Once defined the testing profile, the second step is to implement a test selection algorithm by a sampling scheme that selects inputs according to the defined testing profile (namely, according to π_t values), and that allows an unbiased and efficient estimate of Φ.

The first step is supported by ϑ_i. It is defined the auxiliary variable x associated with each input t such that: $x_{i,t} = p_t \vartheta_i$, where p_t captures the probability of selecting the input t from D_i, and ϑ_i captures the probability of failing conditioned on that selection. The auxiliary variable is used to define the *testing profile*: the probability of selecting t as test case is given by: $\pi_t = \frac{x_{i,t}}{\sum_t x_{i,t}}$.

Note that π_t values can be selected in any different way: if, for instance, the tester does not have failure proneness information, but has reasons to believe that some subdomains are more impacting on reliability than others, he can neglect the construction of the auxiliary variable x and define directly his own testing profile. The approach still works.

As for the second step, a sampling algorithm for unequal sampling is implemented. We tailor the Rao, Hartley and Cochran (RHC) sampling procedure [39], as it is a *without-replacement sampling* scheme (more efficient than with-replacement case) and is a popular sampling method adopted in numerous contexts for its simplicity and practicability. The algorithm steps are:

1) Given the T test cases to execute, divide randomly the $N = |D|$ units of the population into T groups, by selecting G_1 inputs with a *Simple Random Sampling Without Replacement* (SRSWOR) for the first group, then G_2 inputs out of the remaining $(N - G_1)$ for the second, and so on. This will lead to g groups of size $G_1, G_2, \ldots, G_g$ with $\sum_{r=1}^{g} G_r = N$. The group size is arbitrary, but we select $G_1 = G_2 = \cdots = G_g = N/T$, as this minimizes the variance.
2) One test case is then drawn by taking an input t in each of these g groups independently and with a probability proportional to size – in our case, according to π_t values.
3) Denote with $\pi_{t,r}$ the probability associated with the t-th unit in the r-th group, and with $q_r = \sum_{t \in G_r} \pi_{t,r}$ the sum in the r-th group. An unbiased estimator of the failure probability Φ is:

$$\hat{\Phi} = \sum_{r=1}^{g} \frac{p_r z_r}{\pi_r / q_r} \tag{2}$$

where the suffixes $1, 2, \ldots, r$ denote the g test cases selected from the g groups separately.

The estimator is unbiased since $E[\hat{\Phi}] = E_1 E_2[\hat{\Phi}] = E_1[\Phi] = \Phi$, where E_2 is the expectation for a given split and E_1 the expectation over all possible splits into T groups of the chosen sizes. Hence:

$$\hat{R} = 1 - \hat{\Phi} \tag{3}$$

Following [39], the variance of $\hat{\Phi}$ is derived by observing that, under unbiasedness, $V(\hat{\Phi}) = E_1 V_2(\hat{\Phi})$, where V_2 is the variance within a split:

$$V(\hat{\Phi}) = \frac{\sum_r G_r^2 - N}{N(N-1)} \left(\sum_{t=1}^{N} \frac{(p_t z_t)^2}{\pi_t} - \Phi^2 \right) \tag{4}$$

with $\sum_r$ denoting the sum over the $g = T$ groups, and:

$$V(\hat{R}) = V(\hat{\Phi}) \tag{5}$$

Expectedly, the variance decreases as more test cases are devoted to the subdomain. Finally, its unbiased estimator is derived:

$$\hat{V}(\hat{\Phi}) = \frac{\sum_r G_r^2 - N}{N^2 - \sum_r G_r^2} \left(\sum_{r=1}^{g} q_r (\frac{p_r z_r}{\pi_r} - \hat{\Phi})^2 \right). \tag{6}$$

Choosing $G_1 = G_2 = \cdots = G_g = N/T$ simplifies the above expressions:

$$V(\hat{\Phi}) = \frac{1}{T} \frac{(N-T)}{(N-1)} \left(\sum_{t=1}^{N} \frac{(p_t z_t)^2}{\pi_t} - \Phi^2 \right) \tag{7}$$

and:

$$\hat{V}(\hat{\Phi}) = \frac{1}{N} \frac{(N-T)}{(T-1)} \left(\sum_{r=1}^{g} q_r (\frac{p_r z_r}{\pi_r} - \hat{\Phi})^2 \right). \tag{8}$$

Unequal sampling procedure like the presented one is known to be more efficient (hence lower variance, given the same sample size) than Simple Random Sampling (SRS) (which is adopted in conventional partition-based operational testing), because it can exploit the auxiliary information to direct sampling on those units that can contribute more to variance reduction. Additionally, a further improvement is given by the without-replacement case, which is known to be more efficient than the corresponding with-replacement case [35], [40], [39].

For PST to outperform conventional operational testing that ignores any prior belief to select tests, it is sufficient a positive correlation between the auxiliary variable and the variable to estimate. This is shown analytically in our previous work [24] and experimentally confirmed in this work. In other words, PST could work worse if the correlation between the auxiliary information and failure probability is negative: this is a worse situation than a complete absence of knowledge about more or less failure-prone subdomains, because it means that available knowledge is even misleading. In practice, an even partial knowledge (e.g.. inputs from boundary-value regions more likely to fail than others) can be sufficient to distinguish the more (relative) failure-prone subdomains. Without such a knowledge, tester should renounce to any form of partition or component-based testing, as the partitioning criterion can work worse than a uniform testing. In the next Section, we

evaluate both the assessment accuracy and efficiency of this scheme against other techniques, as well as with and without some form of auxiliary knowledge.

IV. Evaluation

A. Objective and Evaluation Method

The primary objective of the evaluation is to assess how much the reliability estimate provided by PST is close to the true reliability, and how efficient it is compared to other testing techniques for reliability assessment. Compared techniques are: *i)* PST; *ii)* the without-replacement version of operational testing (OP), that is arguably the state-of-the-practice technique in reliability testing; *iii)* the newer technique named adaptive testing with gradient-descent method (AT-GD, simply AT in the following) [20], demonstrated to provide better results than operational testing in terms of variance minimisation. We already analysed PST by the analytical perspective in terms of variance it is expected to deliver; here we use both simulation and experimentation as evaluation methods to assess performance under different perspectives.

1) Simulation: Simulation assesses performance under several configurations to see the impact of factors of interest on results without the burden of experimentation. Hypothesising 10 subdomains, performance is studied with respect to two sizes of the input domain: $|D| = 1.0\text{E+4}$ and $|D| = 1.0\text{E+5}$ and an increasing number of test cases ranging from $T = |D|/1.0\text{E+2}$ to $T = |D|/1.0\text{E+1}$, with a step equal to $|D|/1.0\text{E+2}$. Hence, the evaluation is under different ratios of available test cases over the input domain size. The failure points proportion in each subdomain is regulated by a probability P of an input to be a failure point (i.e.: an input t is marked as failing point with probability P or as correct point with probability $(1-P)$). To include the cases of both low, medium, and high reliability systems, the value of P is allowed to vary in three ranges [1.0E-2; 1.0E-3]; [1.0E-3; 1.0E-4]; [1.0E-4; 1.0E-5]. A simulation scenario j consists of a 4-way combination: $<$*technique*, *domain size*, *#test cases*, P *range* $>$. Since sampling-based testing techniques are randomized algorithm, each scenario is repeated 100 times to draw statistically valid conclusions. For each repetition, a different operational profile is generated randomly and kept fixed for all the 100 repetitions[2]. For simulation, we assume the conservative case of "no exploitable knowledge" by PST: namely, all subdomains are given the same failure probability, approximating the case in which a tester has a uniform belief about the failure proneness of each subdomain (i.e., the assumption that better represents ignorance). The number of runs is: (3 techniques x 3 domain sizes x 10 test case points x 3 ranges of P) x 100 repetitions = 27,000 runs. Results are in Section V.

[2]Profile is generated by randomly assigning a [0; 1] number q_i to each subdomain i, according to a uniform distribution, then normalising the output to sum up to 1: $p_i = \frac{q_i}{\sum_i q_i}$. Inputs within subdomains are selected with a uniform distribution with probability $p_t = p_i/|D_i|$, where $|D_i|$ is the size of subdomain i.

2) Experimentation: Simulation by itself is not sufficient to make claims on real systems, as the latter introduce further variability due to uncontrollable (and often unknown) factors. We assess performance of PST also experimentally on a real-scale case study. We test reliability of MySQL DBMS v5.6.35 deployed on an $iMac$ machine (3.5 GHz Intel Core i7, 32 GB DD3 RAM, 3.2 TB Hard Disk) with the same techniques listed above. Reliability of version 5.6.35 is computed by running the test suite of the subsequent version 5.7.7, so that real bugs are expected to be found. For experimentation purpose, the input space is assumed to be equivalent to the test space given by the available test suite. The actual size of the test suite depends on DBMS configuration and target machine: in our case it amounted to 4,632 test cases, out of which 1,108 are marked as *skipped* or *disabled* by MySQL Test Framework (e.g., because they require a specific storage engine, or GTID based logging active, or because are temporarily disabled by developers) – thus the final test space is of 3,524 tests. Test cases are organised in test suites, which group together logically close tests, e.g.. *innodb*, *optimizer*, *partitions*. We treat the test suites, or sets of similar test suites, as our subdomains, since they basically correspond to the MySQL logical architectural components. In some cases, some manual work to merge test suites into one logical component and/or to map sparse test cases was required to get a cleaner mapping – the final set of components is in the first column of Table V.

Two versions of PST are tested, one under a uniform belief about relative failure proneness of components (like in the simulation case), one with the SRGM-based auxiliary information – see Section III-B. SRGMs are built at component level over the previous versions' failure data. Number of test cases is allowed to vary between 100 and 1,000, with a step of 100 test cases. Three operational profiles are generated randomly with the same procedure as in the simulation case. An experimental scenario j is made up of these configurations: $<$*technique*, *#test cases*, *profile* $>$ where the techniques are, in this case, 4: OP, AT, PST with uniform auxiliary information (PST_U) and SRGM-based PST (PST_{SRGM}) – getting to (4 techniques x 10 test case points x 3 profiles) x 100 repetitions = 12,000 runs. Results of experimentation are in Section VI.

B. Evaluation criteria

If we focus exclusively on the analytical derivation as reported in Section III, we could consider only a variance analysis: indeed, since the estimator is unbiased, its *expected value* exactly estimates the true reliability – so, what makes the difference in terms of expectation is the variance of the estimates, namely its efficiency. However, for both simulation and experimentation, the technique needs also to be assessed in terms of estimation *accuracy*, besides efficiency, since the number of tests is limited (hence, there is actually an offset between the point estimate and true reliability, despite the estimate expected value is equal to the true value). Therefore, we focus on both estimation *accuracy* and *efficiency*. An experimented scenario j is repeated 100 times; denote with r one of such repetitions. At the end of each repetition,

the reliability estimate $\hat{R}_{r,j}$ is computed by the technique under assessment as well as the *true* reliability R_j. In the simulation case, we know in advance which input t is a failure point (hence, $R_j = \sum_{t \in D} p_t z_t$, where z_t is 1 if the input is a failure point and 0 otherwise). In the experimentation case, we had to preliminary run all the test cases, and mark failing and passing test cases, so as to get the z_t labels for each input – these are stored in a file and then used by the experiment support code just like in the simulation case to compute true reliability. Profile is generated randomly by the simulation/experimentation code at each repetition as explained in the previous Section.

For each scenario j and for each technique, we compute the sample mean (denoted as M), sample variance (S) and mean squared error (MSE):

$$M(\hat{R}_j) = \frac{1}{100} \sum_{r=1}^{100} \hat{R}_{r.j}$$

$$S(\hat{R}_j) = \frac{1}{100-1} \sum_{r=1}^{100} (\hat{R}_{r.j} - M(\hat{R}_j))^2 \quad (9)$$

$$MSE(\hat{R}_j) = \frac{1}{100} \sum_{r=1}^{100} (\hat{R}_{r.j} - R_j))^2$$

Comparison of estimation accuracy is done by looking at the MSE. Comparison of efficiency is done by the sample variance S.

V. Simulation Results

Simulation results are in Figure 1. Each graph reports the MSE over the 100 repetitions of the three compared *techniques* with all the 10 values of *#test cases*, under a fixed domain size and failure point probability ranges P. Figure 2 reports the sample variances S.

What can be seen graphically is that PST has in most cases lower MSE and variance. The difference with the other techniques is particularly evident in configurations 3 and 6, namely in those configurations characterized by very few failure points, with a very high reliability (in those cases, the true reliability was always higher than 0.999, the exact value depending on the operational profile that change in each of the ten scenarios). The difference is still relevant in configuration 2 and 5, where there is a medium range of failure points probability (true reliability higher than 0.99 lower than 0.999), but more pronounced in configuration 2 (characterized by $|D|$=1.0E+4) than configuration 5 ($|D|$=1.0E+5). When reliability is of the order 0.9 (lower than 0.99), the difference between the techniques is not so relevant. Detailed results for each configuration are in Tables I and II, where it is possible to read the best values reached by PST (as well as by AT) that cannot be read by the graph because too low. For instance, in configuration 6, the MSE of PST is an order of magnitude lower than the others.

It is interesting to note the performance of AT: if we look at configuration 3 and 6 (the ultra-high reliability cases), we can note that it has a very low sample variance, comparable with PST or sometimes even better, but in the same configurations it has a bad MSE, close to OP testing – in other words it achieved a very precise estimate of reliability but not much accurate, with a relatively large bias. Moreover, while it works well in low reliability configurations (1 and 4), it seems to loose efficiency in configurations 2 and 5 especially in the cases with a low number of tests (points 1-3 on the *x*-axis).

Configurations	*Techniques*	**Median**	**Mean**	**Min**	**Max**
Configuration 1	*PST*	1.03E-04	1.64E-04	3.99E-05	5.19E-04
	OP	1.04E-04	2.16E-04	6.17E-05	8.72E-04
	AT	1.03E-04	1.87E-04	5.44E-05	7.24E-04
Configuration 2	*PST*	1.37E-05	2.74E-05	4.71E-06	9.66E-05
	OP	9.97E-05	5.93E-04	1.43E-05	2.40E-03
	AT	1.16E-04	5.28E-04	1.69E-05	3.16E-03
Configuration 3	*PST*	6.90E-07	2.98E-06	2.26E-07	2.41E-05
	OP	2.59E-04	4.49E-04	8.67E-05	1.33E-03
	AT	1.85E-04	4.13E-04	6.17E-05	1.26E-03
Configuration 4	*PST*	7.97E-06	1.41E-05	4.33E-06	4.35E-05
	OP	9.25E-06	1.67E-05	6.27E-06	5.64E-05
	AT	9.07E-06	1.53E-05	4.52E-06	5.17E-05
Configuration 5	*PST*	8.49E-07	1.40E-06	2.24E-07	5.35E-06
	OP	1.15E-06	1.66E-06	4.96E-07	5.89E-06
	AT	1.15E-06	1.66E-06	4.96E-07	5.89E-06
Configuration 6	*PST*	9.72E-08	2.37E-07	2.19E-08	1.36E-06
	OP	1.16E-06	1.11E-05	4.04E-07	6.94E-05
	AT	1.00E-06	7.94E-06	3.37E-07	5.65E-05

TABLE I: Statistics of testing Configurations. **MSE**.

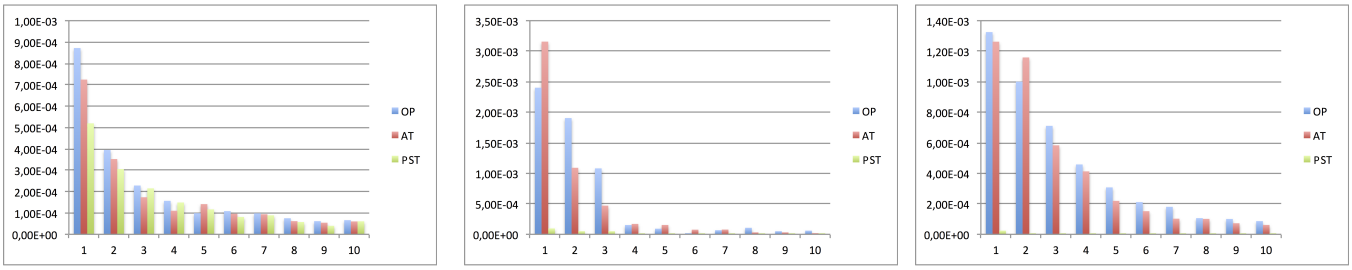
Configurations	*Techniques*	**Median**	**Mean**	**Min**	**Max**
Configuration 1	*PST*	9.56E-05	1.26E-04	3.97E-05	5.23E-04
	OP	1.01E-04	1.65E-04	6.20E-05	5.76E-04
	AT	1.03E-04	1.46E-04	4.67E-05	5.56E-04
Configuration 2	*PST*	3.70E-06	6.76E-06	1.09E-06	1.68E-05
	OP	1.24E-05	1.60E-05	4.98E-06	4.77E-05
	AT	5.79E-06	2.14E-05	1.10E-06	9.17E-05
Configuration 3	*PST*	5.68E-08	2.84E-07	1.33E-08	2.40E-06
	OP	1.05E-06	1.07E-06	1.70E-07	2.36E-06
	AT	1.04E-07	3.02E-07	1.62E-10	2.33E-06
Configuration 4	*PST*	7.94E-06	1.38E-05	4.38E-06	4.29E-05
	OP	9.29E-06	1.44E-05	6.32E-06	3.66E-05
	AT	8.25E-06	1.33E-05	4.10E-06	3.62E-05
Configuration 5	*PST*	7.63E-07	1.29E-06	4.10E-07	5.39E-06
	OP	9.14E-07	1.58E-06	4.99E-07	5.94E-06
	AT	7.92E-07	1.37E-06	3.37E-07	6.10E-06
Configuration 6	*PST*	8.49E-07	5.69E-08	1.12E-08	1.68E-07
	OP	1.15E-06	1.67E-07	4.08E-08	6.90E-07
	AT	1.01E-06	9.08E-08	1.83E-08	3.05E-07

TABLE II: Statistics of testing Configurations. **Sample Variance**.

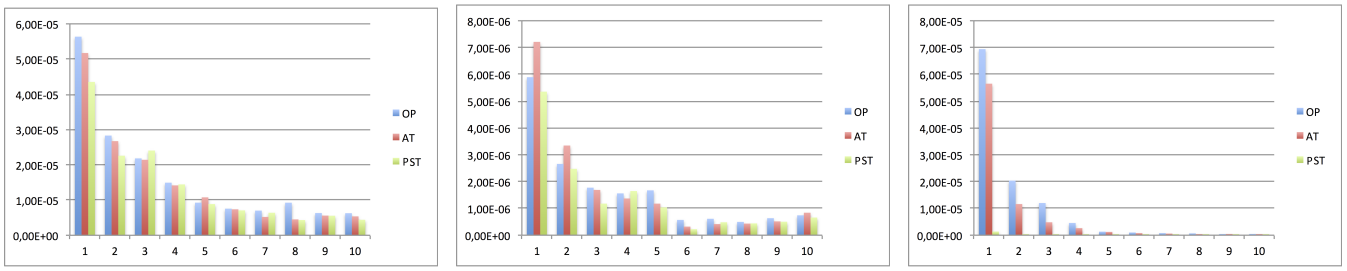
In Tables IVa and Va we report the overall median and mean values of MSE and sample variance, confirming the superiority of PST for both criteria. Tables IVb and Vb confirm the rejection of the hypothesis that samples from compared techniques (row *vs* column) come from the same distribution. Since we are interested in comparing all the techniques with each other, we have adopted the Nemenyi test for pairwise comparison of a multi-level experimental factor. The test uses the *critical difference* (CD): two levels of a compared factor are significantly different if the corresponding average ranks differ by at least CD = $q_\alpha \sqrt{k(k+1)/6N}$. where q_α values are the Studentized range statistic divided by $\sqrt{2}$, and adjusted according to the number of comparisons[3], k is the number of levels compared, N is the sample size [41]. Results confirm that all the observed differences are statistically significant.

It is worth noting that these results use the assumption of uniform belief of failure probability of partitions, thus not

[3]As the family-wise error rate is already controlled by considering q_α, no other multiple comparison protection procedure is needed.

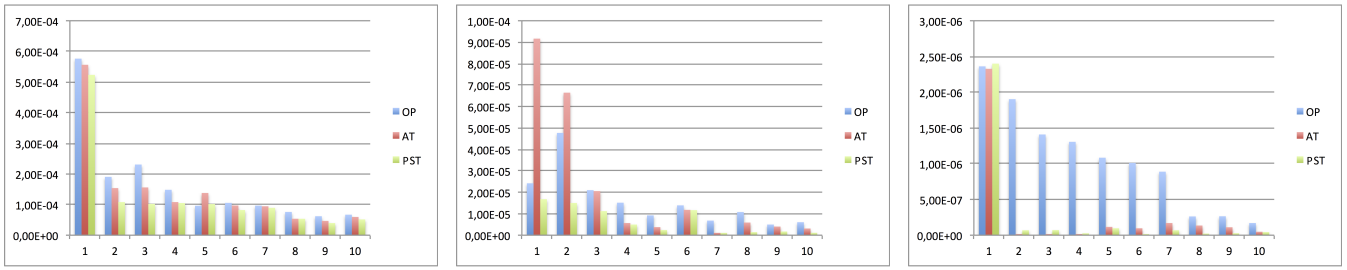


(a) Configuration 1 (b) Configuration 2 (c) Configuration 3

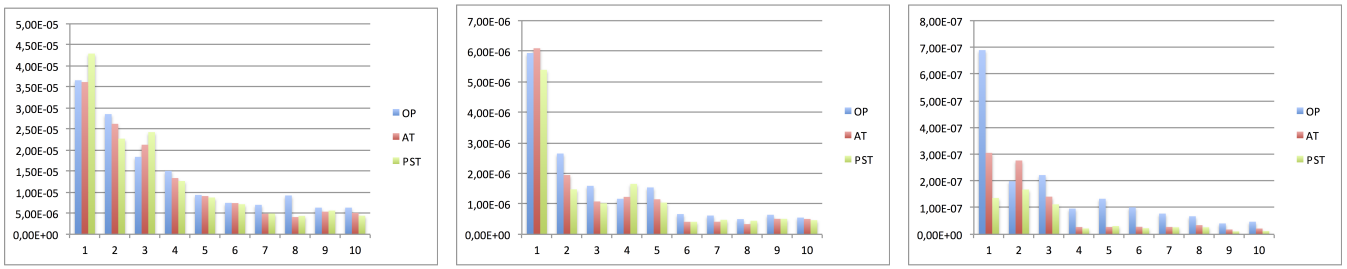


(d) Configuration 4 (e) Configuration 5 (f) Configuration 6

Fig. 1: MSE of reliability estimate. Legend: Configuration 1: P= [1.0E-2;1.0E-3]; |D| = 1.0E+4. Configuration 2: P = [1.0E-3;1.0E-4]; |D| = 1.0E+4. Configuration 3: P= [1.0E-4;1.0E-5]; |D| = 1.0E+4. Configuration 4: P = [1.0E-2;1.0E-3]; |D| = 1.0E+5. Configuration 5: P = [1.0E-3;1.0E-4]; |D| = 1.0E+5. Configuration 6: P = [1.0E-4;1.0E-5]; |D| = 1.0E+5



(a) Configuration 1 (b) Configuration 2 (c) Configuration 3



(d) Configuration 4 (e) Configuration 5 (f) Configuration 6

Fig. 2: Sample variance of reliability estimate. Legend: Configuration 1: P= [1.0E-2;1.0E-3]; |D| = 1.0E+4. Configuration 2: P = [1.0E-3;1.0E-4]; |D| = 1.0E+4. Configuration 3: P= [1.0E-4;1.0E-5]; |D| = 1.0E+4. Configuration 4: P = [1.0E-2;1.0E-3]; |D| = 1.0E+5. Configuration 5: P = [1.0E-3;1.0E-4]; |D| = 1.0E+5. Configuration 6: P = [1.0E-4;1.0E-5]; |D| = 1.0E+5

exploiting the full potential of PST; better results are expected if we use auxiliary information.

VI. Experimentation Results

A. Setup of PST_{SRGM}

We consider the case when SRGMs can be built from historical data and inform about components' expected reliability. The steps to build the SRGMs for MySQL components are:

Overall median and mean values		
	Median	*Mean*
PST	4.55E-06	3.49E-05
OP	5.36E-05	2.15E-4
AT	3.19E-05	1.92E-4

(a)

Pairwise Comparison: *p-values*		
	AT	*PST*
OP	**2.42E-5**	**<1.0E-15**
AT	-	**5.17E-4**

(b)

TABLE III: Comparison for MSE

Overall median and mean values		
	Median	*Mean*
PST	1.56E-06	2.46E-05
OP	5.45E-06	3.30E-05
AT	2.72E-06	3.04E-05

(a)

Pairwise Comparison: *p-values*		
	AT	*PST*
OP	**4.25E-6**	**<1.0E-15**
AT	-	**2.91E-3**

(b)

TABLE IV: Comparison for Sample Variance **S**.

- *Failure data extraction*. We extracted bug reports from the MySQL bug repository[4]. All the closed bugs referring to a component (derived by the category field), spanning from 2003 to 2017, are taken. We consider closed bugs, since only closed bugs actually indicate the growth of reliability[5], and non-closed bugs might turn out to be not a bug, duplicate, feature requests, etc..
- *Data fitting*: we used the following set of SRGMs, applied to each component: Exponential, Log Normal, Truncated Normal, Log Logistic, Truncated Logistic, Truncated Extreme-Value Min, Truncated Extreme-Value Max. The SRGM with the best Akaike Information Criterion (AIC) value is selected, like in [42], [43], [44].
- *Auxiliary variable computation*. For selected SRGMs, we take the failure intensity at the last correction time T: $\lambda_i(T) = \lambda_i$ from which the SRGM-based failure probability estimate ϑ_i and then the auxiliary variable $x_{i,t}$ are computed as described in Section III.

Figure 3 reports two examples of SRGM of relevant components (*replication* and *MyISAM*). Table V reports the results of applying the SRGM at component level. For each component, it is reported: the number of defects closed; the total number of operational hours since the first closed bug; the best fitting SRGM; the failure intensity λ_i; the SRGM-based failure probability estimate ϑ_i.

B. Results

Figure 4 reports the MSE values and sample variances reached by the four compared techniques in the 10 points (from 100 to 1000 test cases), for the three profiles.

In terms of MSE, PST clearly outperforms the others in the three cases. AT, in line with what observed in the simulation case, shows a higher MSE, often higher than OP testing. In fact, this is a scenario characterized by a test space of 3,524 points and a number of failure points, derived by failed test cases, equal to 26; the true reliability, is in the medium range

[4]*http://bugs.mysql.com*

[5]MySQL repository has not the resolution field, hence closed bugs are also "solved" bugs

MySQL Component	*Defects closed*	*operational time (h)*	*Applied SRGM*	*Failure intensity*	*Failure prob.*
Replication	1259	123313	TEVM	4.89E-3	4.77E-2
InnoDB	1154	123161	TL	6.67E-6	6.66E-05
Optimizer	1010	105156	TEVm	6.77E-3	6.54E-2
Tests	63	80149	EXP	2.54E-4	2.53E-3
Partitions	374	99956	TEVM	4.63E-4	4.61E-3
Charsets	193	94782	TL	1.34E-6	1.33E-05
Perf. Schema	147	64560	EXP	1.01E-3	1.00E-2
Parser	101	91897	EXP	4.50E-4	4.48E-3
Federated	47	72611	EXP	1.84E-5	1.83E-4
GIS	74	79986	TEVM	1.51E-3	1.49E-2
Logging	117	93350	TEVm	2.57E-3	2.53E-2
SysSchema	14	10212	EXP	9.95E-4	9.9E-3
DDL	193	82550	TEVm	4.79E-3	4.67E-2
DML	262	81791	EXP	2.38E-3	2.35E-2
MyISAM	315	110226	TEVM	1.48E-4	1.47E-3
JSON	38	29229	EXP	5.16E-4	5.14E-3
Security	137	91981	TEVM	2.05E-3	2.02E-2
Query cache	29	76705	EXP	1.01E-4	1.00E-3
Memcached	22	22230	TRN	8.44E-5	8.43E-4
Connection	9	14319	EXP	1.48E-3	1.46E-2
Doc store	16	8115	EXP	1.74E-3	1.72E-2

TABLE V: MySQL components SRGMs characterization

[0.99; 0.999], the exact value depending on the profile. In terms of sample variance, AT is indeed better than OP and with values similar to PST_U. The usage of auxiliary information given by the SRGM remarkably improves performance of PST, both in terms of MSE and sample variance. Average results per profile are given in Tables VI and VII, showing that PST_{SRGM} has often values of an order of magnitude lower than the others.

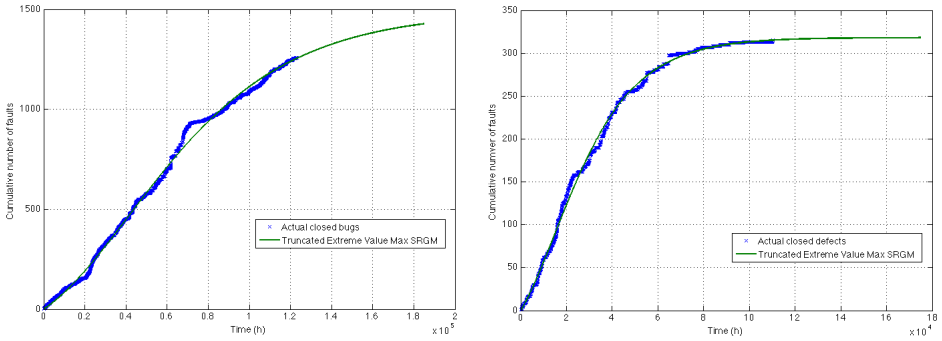
Profile	*Techniques*	**Median**	**Mean**	**Min**	**Max**
Profile 1	PST_{SRGM}	2.55E-05	3.51E-05	1.17E-05	1.01E-04
	PST_U	3.00E-05	8.56E-05	9.78E-06	5.12E-04
	OP	9.09E-05	1.63E-04	1.59E-05	6.16E-04
	AT	8.75E-05	2.48E-04	2.40E-05	1.09E-03
Profile 2	PST_{SRGM}	9.07E-06	2.13E-05	4.34E-06	1.02E-04
	PST_U	2.59E-05	5.52E-05	1.07E-05	1.91E-04
	OP	9.99E-05	1.68E-04	6.23E-05	5.98E-04
	AT	8.65E-05	2.46E-04	2.37E-05	1.07E-03
Profile 3	PST_{SRGM}	1.08E-05	1.92E-05	5.30E-06	7.76E-05
	PST_U	2.64E-05	5.08E-05	1.38E-05	2.42E-04
	OP	1.05E-04	2.05E-04	4.94E-05	7.49E-04
	AT	8.75E-05	2.21E-04	2.40E-05	8.19E-04

TABLE VI: MSE Statistics

Profile	*Techniques*	**Median**	**Mean**	**Min**	**Max**
Profile 1	PST_{SRGM}	2.20E-05	2.21E-05	8.13E-06	3.29E-05
	PST_U	2.57E-05	2.80E-05	9.61E-06	5.12E-05
	OP	3.36E-05	3.64E-05	2.49E-05	6.22E-05
	AT	2.69E-05	2.89E-05	1.68E-05	4.74E-05
Profile 2	PST_{SRGM}	2.04E-06	4.72E-06	1.52E-06	9.80E-06
	PST_U	1.00E-05	1.67E-05	6.21E-06	5.94E-05
	OP	2.61E-05	4.19E-05	1.06E-05	1.09E-04
	AT	1.06E-05	1.24E-05	2.05E-06	4.18E-05
Profile 3	PST_{SRGM}	2.09E-06	6.32E-06	9.79E-07	1.96E-05
	PST_U	1.04E-05	2.05E-05	4.99E-06	7.48E-05
	OP	2.66E-05	5.11E-05	1.38E-05	2.44E-04
	AT	1.67E-05	2.45E-05	1.47E-06	8.08E-05

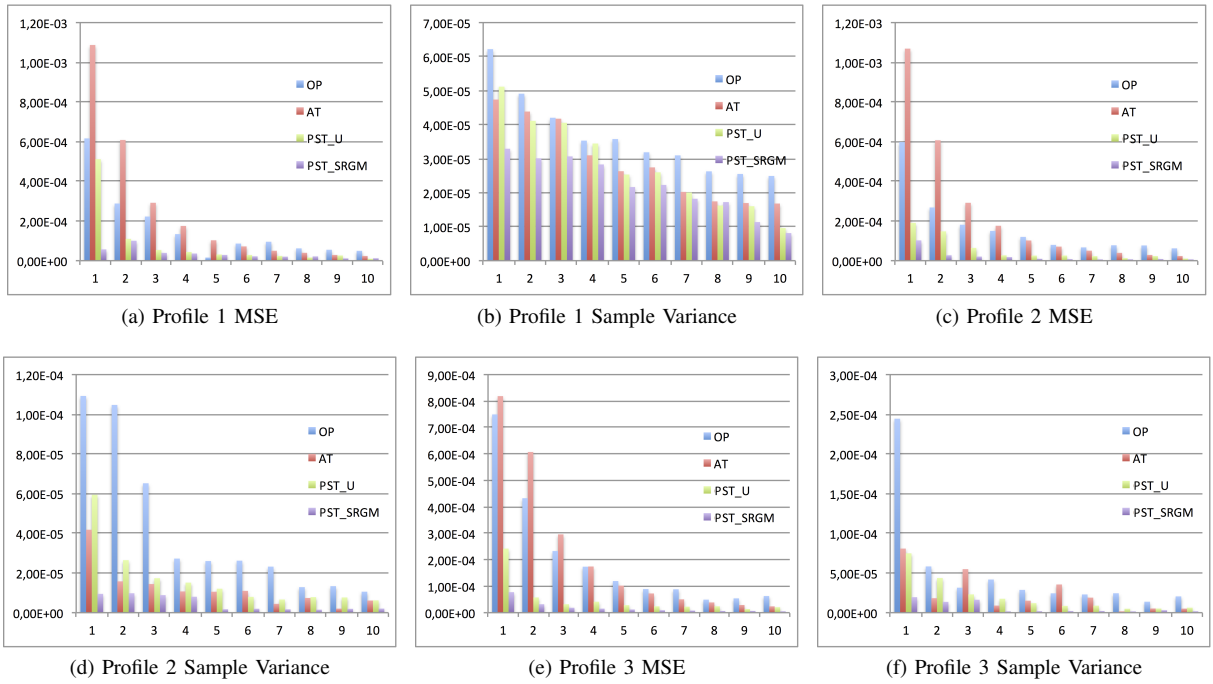
TABLE VII: Sample Variance statistics

Statistical analysis corroborates the previous observations; it is worth noting that the sample variance difference between AT and PST_U is not significant (p-value = 0.29), as can be seen by the similar values in the graphs as well as in Table



(a) Replication (b) MyISAM

Fig. 3: SRGMs built for *Replication* and *MyISAM* components



(a) Profile 1 MSE (b) Profile 1 Sample Variance (c) Profile 2 MSE

(d) Profile 2 Sample Variance (e) Profile 3 MSE (f) Profile 3 Sample Variance

Fig. 4: MSE values and Sample Variances of reliability estimate

8. The other non-significant difference is between OP and AT in terms of MSE (AT performed better for larger number of test cases but worse for small number of tests) – Table IXb.

To validate a posteriori the goodness of choosing the SRGM as information source for components' failure probability estimate, we have computed the Pearson and Spearman correlation between the failure probability values ϑ_i assigned to each component and the corresponding number of failing tests actually observed on those components. Values are in Table X; a value for each experimented scenario (3 profiles x 10 values for "number of test cases"). The positive values indeed helped in improving performance of PST$_{SRGM}$ over the others.

Overall median and mean values		
	Median	*Mean*
PST_{SRGM}	1.68E-05	2.51E-05
PST_U	2.68E-05	6.38E-05
OP	9.24E-05	1.78E-4
AT	8.73E-05	2.38E-4

(a)

Pairwise Comparison: *p-values*			
	AT	PST_{SRGM}	PST_U
OP	2.05E-1	**<1.0E-15**	**1.75E-6**
AT	-	**<1.0E-15**	**4.98E-5**
PST_{SRGM}	-	**-**	**2.6E-2**

(b)

TABLE VIII: Comparison for MSE

Overall median and mean values		
	Median	*Mean*
PST_{SRGM}	8.48E-06	1.10E-05
PST_U	1.61E-05	2.17E-05
OP	2.79E-05	4.31E-05
AT	1.68E-05	2.19E-05

(a)

Pairwise Comparison: *p-values*			
	AT	PST_{SRGM}	PST_U
OP	**8.54E-5**	**<1.0E-15**	**8.4E-5**
AT	-	**4.03-5**	2.94E-1
PST_{SRGM}	-	**-**	**8.0E-5**

(b)

TABLE IX: Comparison for Sample Variance **S**.

#Tests	Profile 1		Profile 2		Profile 3	
	Pearson	Spearman	Pearson	Spearman	Pearson	Spearman
100	0.44	0.23	0.41	0.28	0.39	0.22
200	0.48	0.26	0.49	0.37	0.44	0.26
300	0.56	0.38	0.55	0.42	0.49	0.29
400	0.58	0.42	0.56	0.45	0.54	0.36
500	0.62	0.47	0.64	0.47	0.55	0.38
600	0.64	0.48	0.65	0.48	0.65	0.49
700	0.70	0.52	0.67	0.51	0.68	0.51
800	.070	0.51	0.72	0.54	0.68	0.52
900	0.71	0.54	0.71	0.55	0.73	0.55
1000	0.72	0.55	0.75	0.56	0.72	0.56

TABLE X: Correlation between ϑ and number of failing test cases in each component

VII. Threats to validity

Simulation results are useful to understand performances of different algorithms. The accuracy of the results depends on how closely the simulation represents a realistic scenario. From the test results point of view, the size of the input space, the number of failure points and their distribution over the input space, the occurrence probability of each input are all factors that have an impact on the final result. With respect to them, we have varied those factors to cover more scenarios, replicated 100 times each scenario, and generated profiles randomly to reduce the effect of the variability due to these factors. However, the difference with real system can be impactful: in a real system, further variability is likely to be introduced because of uncontrollable/unknown factors, such as the impact of the execution environment, the interaction with tester, the effect of an imperfect partitioning – all factors that can undermine the assumptions made in Section III. For this reason, we have complemented simulation by an experimental analysis on real-scale system, where results account for the impact of possible assumptions' violation.

With respect to the experimentation, we warn the reader against the following threats. With respect to the identification of components, we have derived it from the already grouped test suites, combining it with the logical architecture of MySQL, and applied SRGMs to them. An SRGM describes the detection/correction trend of faults, but the availability of failure data from the bug repository is influenced by how much each component is used by the community. Therefore, few failure reports might indistinguishably indicate a good reliability of the component or a scarse usage. Under the same testing effort, the SRGMs could be different. Also, bug reports quality may impact (e.g., bugs put in the wrong category). On the other hand, we claimed that the belief acquired before acceptance testing can be even loosely coupled with actual reliability – paradoxically, if we had certainty about components' reliability we would not need to carry out any assessment. Considering also the absence of any belief (i.e., the uniform case), the results demonstrate that the algorithm is robust even to bad quality of preliminary reliability beliefs. We contrasted two cases (uniform belief represents absence of knowledge and SRGM represents a generally good quantitative belief) and performance was satisfactory in both cases. Integrating further evidences is left to future research. Finally, external validity is mined by the single system used. Indeed, while the simulation study can help improve statistical generalizability, we cannot claim that experimental results are valid for other real systems with different characteristics with respect to MySQL.

VIII. Conclusion

This article presented Probabilistic Sampling-based Testing (PST), a new testing technique for reliability assessment. The key idea is to enable testing profiles different from the operational one to be used for assessment purpose. There are many strategies that can stress the bug-finding ability regardless the occurrence probability of selected inputs; while they are usually considered as pursuing a separate objective, they can still be used for reliability assessment, to expedite the assessment, by counterbalancing the different-from-operation probability of test cases selection with proper weights in the estimation. This approach eases the integration with other sources of reliability assessment gained before acceptance testing, which can be used as auxiliary information to expedite the assessment.

In this work, we have neglected the role of *adaptivity*. As we experimented in our recent work, adaptivity can further improve the test selection by changing the testing profile online, as testing results are observed [5], [24], [45]. Therefore, in the next future we aim at combining adaptivity with the presented scheme. Moreover, in this work we have instantiated one specific sampling procedure and estimators for unequal sampling; a wider exploitation of complex survey design [35] is envisioned to look for formulations that can exploit most of the available, yet currently neglected, information.

Acknowledgements

This work has been supported by the GAUSS national research project, which has been funded by MIUR under the PRIN 2015 program (grant n. 2015KWREMX_002).

References

[1] J. Musa, Software reliability-engineered testing, Computer 29 (11) (1996) 61–68.

[2] H. Singh, V. Cortellessa, B. Cukic, E. Gunel, V. Bharadwaj, A bayesian approach to reliability prediction and assessment of component based systems, in: Proceedings 12th International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2001, pp. 12–21.

[3] L. Strigini, B. Littlewood, Guidelines for statistical testing, Tech. Rep. PASCON/WO6-CCN2/TN12, ESA/ESTEC project PASCON (1997).

[4] K.-Y. Cai, Y.-C. Li, K. Liu, Optimal and adaptive testing for software reliability assessment, Information and Software Technology 46 (15) (2004) 989–1000.

[5] D. Cotroneo, R. Pietrantuono, S. Russo, RELAI Testing: A Technique to Assess and Improve Software Reliability, IEEE Trans. on Software Engineering 42 (5) (2016) 452–475.

[6] D. Cotroneo, R. Pietrantuono, S. Russo, Combining Operational and Debug Testing for Improving Reliability, IEEE Trans. on Reliability 62 (2) (2013) 408–423.

[7] P. Frankl, D. Hamlet, B. Littlewood, L. Strigini, Evaluating Testing Methods by Delivered Reliability, IEEE Trans. on Software Engineering 24 (8) (1998) 586–601.

[8] C. Smidts, B. Cukic, E. Gunel, M. Li, H. Singh, Software reliability corroboration, in: Proceedings 27th Annual NASA Goddard/IEEE Software Engineering Workshop, IEEE, 2002, pp. 82–87.

[9] M. Neil, N. Fenton, L. Nielson, Building Large-scale Bayesian Networks, Knowl. Eng. Rev. 15 (3) (2000) 257–284.

[10] H. Mills, M. Dyer, R. Linger, Cleanroom software engineering, IEEE Software 4 (55) (1987) 19–24.

[11] P. Currit, M. Dyer, H. Mills, Certifying the reliability of software, IEEE Trans. on Software Engineering SE-12 (1) (1986) 3–11.

[12] R. Cobb, H. Mills, Engineering software under statistical quality control, IEEE Software 7 (6) (1990) 45–54.

[13] R. Linger, H. Mills, A case study in cleanroom software engineering: the ibm cobol structuring facility, in: Proceedings 12th Int. Computer Software and Applications Conference (COMPSAC), IEEE, 1988, pp. 10–17.

[14] J. Poore, A case study using cleanroom with box structures ADL, Tech. rep., CDRL 1880, Software Engineering Technology, Vero Beach, Fla. (1990).

[15] M. R. Lyu (Ed.), Handbook of Software Reliability Engineering, McGraw-Hill, Inc., Hightstown, NJ, USA, 1996.

[16] L. Madani, C. Oriat, I. Parissis, J. Bouchet, L. Nigay, Synchronous testing of multimodal systems: an operational profile-based approach, in: 16th IEEE Int. Symposium on Software Reliability Engineering (ISSRE), 2005, pp. 10 pp.–334.

[17] B. Beizer, Cleanroom process model: a critical examination, IEEE Software 14 (2) (1997) 14–16.

[18] K.-Y. Cai, C.-H. Jiang, H. Hu, C.-G. Bai, An experimental study of adaptive testing for software reliability assessment, Journal of Systems and Software 81 (8) (2008) 1406–1429.

[19] K.-Y. Cai, Optimal software testing and adaptive software testing in the context of software cybernetics, Information and Software Technology 44 (14) (2002) 841–855.

[20] J. Lv, B.-B. Yin, K.-Y. Cai, On the Asymptotic Behavior of Adaptive Testing Strategy for Software Reliability Assessment, IEEE Trans. on Software Engineering 40 (4) (2014) 396–412.

[21] J. Lv, B.-B. Yin, K.-Y. Cai, Estimating confidence interval of software reliability with adaptive testing strategy, Journal of Systems and Software 97 (2014) 192–206.

[22] A. Podgurski, W. Masri, Y. McCleese, F. Wolff, C. Yang, Estimation of software reliability by stratified sampling, ACM Transactions on Software Engineering and Methodology 8 (3) (1999) 263–283.

[23] F. Omri, Weighted statistical white-box testing with proportional-optimal stratification, in: Proc. 19th International Doctoral Symposium on Components and Architecture, WCOP'14, ACM, 2014, pp. 19–24.

[24] R. Pietrantuono, S. Russo, On Adaptive Sampling-Based Testing for Software Reliability Assessment, in: Proceedings 27th International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2016, pp. 1–11.

[25] C. Smidts, M. Stutzke, R. W. Stoddard, Software reliability modeling: an approach to early reliability prediction, IEEE Transactions on Reliability 47 (3) (1998) 268–278.

[26] D. S. Herrmann, Sample implementation of the littlewood holistic model for assessing software quality, safety and reliability, in: Annual Reliability and Maintainability Symposium. 1998 Proceedings. International Symposium on Product Quality and Integrity, 1998, pp. 138–148. doi:10.1109/RAMS.1998.653698.

[27] K. Goševa-Popstojanova, K. S. Trivedi, Architecture-based Approach to Reliability Assessment of Software Systems, Performance Evaluation 45 (2-3) (2001) 179–204.

[28] P. Popov, Proc. 21st int. conference on computer safety, reliability and security, SAFECOMP, Springer, 2002, pp. 139–150. doi:10.1007/3-540-45732-1_15.

[29] I. Gashi, P. Popov, V. Stankovic, Uncertainty explicit assessment of off-the-shelf software: A bayesian approach, Information and Software Technology 51 (2) (2009) 497–511.

[30] K. Kanoun, M. Kaaniche, J. P. Laprie, Qualitative and quantitative reliability assessment, IEEE Software 14 (2) (1997) 77–87.

[31] L. Strigini, D. Wright, Bounds on survival probability given mean probability of failure per demand; and the paradoxical advantages of uncertainty, Reliability Engineering & System Safety 128 (2014) 66–83.

[32] K.-Y. Cai, L. Cai, W.-D. Wang, Z.-Y. Yu, D. Zhang, On the neural network approach in software reliability modeling, Journal of Systems and Software 58 (1) (2001) 47 – 62.

[33] T. J. Ostrand, M. J. Balcer, The category-partition method for specifying and generating fuctional tests, Commun. ACM 31 (6) (1988) 676–686. doi:10.1145/62959.62964.

[34] C. Catal, B. Diri, A systematic review of software fault prediction studies, Expert Systems with Applications 36 (4) (2009) 7346–7354. doi:10.1016/j.eswa.2008.10.027.

[35] S. L. Lohr, Sampling Design and Analysis, Duxbury Press; 2 edition, 2009.

[36] M. Cinque, D. Cotroneo, A. Pecchia, R. Pietrantuono, S. Russo, Debugging-workflow-aware software reliability growth analysis, Software Testing, Verification and Reliability 27 (7).

[37] B. Yang, M. Xie, A study of operational and testing reliability in software reliability analysis, Reliability Engineering & System Safety 70 (3) (2000) 323 – 329.

[38] R. Pietrantuono, S. Russo, K. Trivedi, Software Reliability and Testing Time Allocation: An Architecture-Based Approach, IEEE Trans. on Software Engineering 36 (3) (2010) 323–337.

[39] J. Rao, H. Hartley, W. Cochran, On a simple procedure of unequal probability sampling without replacement, Journal of the Royal Statistical Society. Series B (Methodological) 24 (2) (1962) 482–491.

[40] A. Chaudhuri, Survey Sampling Theory and Methods, Chapman & Hall/CRC, Second Edition, Taylor & Francis Group, 2005.

[41] J. Demšar, Statistical Comparisons of Classifiers over Multiple Data Sets, Journal of Machine Learning Research 7 (2006) 1–30.

[42] G. Carrozza, R. Pietrantuono, S. Russo, Dynamic test planning: a study in an industrial context, International Journal on Software Tools for Technology Transfer 16 (5) (2014) 593–607.

[43] R. Pietrantuono, P. Potena, A. Pecchia, D. Rodriguez, S. Russo, L. Fernández-Sanz, Multiobjective testing resource allocation under uncertainty, IEEE Transactions on Evolutionary Computation 22 (3) (2018) 347–362.

[44] K. Ohishi, H. Okamura, T. Dohi, Gompertz software reliability model: Estimation algorithm and empirical validation, Journal of Systems and Software 82 (3) (2009) 535–543.

[45] D. Cotroneo, R. Pietrantuono, S. Russo, A Learning-based Method for Combining Testing Techniques, in: Proc. 35th Int. Conference on Software Engineering (ICSE), IEEE, 2013, pp. 142–151.