Low-Variance Software Reliability Estimation Using Statistical Testing

Fouad ben Nasr Omri, Safa Omri and Ralf Reussner

Chair for Software Design and Quality
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: {fouad.omri, safa.omri, ralf.reussner}@kit.edu

***Abstract*—Software statistical testing establishes a basis for statistical inference about the expected field quality of software based on an expected operational profile. The standard statistical testing approach draws randomly test cases from the expected operational profile according to the statistical distribution on the expected inputs. Standard Statistical testing is in the most of the cases impractical. The number of required test cases to reach a target reliability with a given confidence is too large. In this paper, we present a test selection approach to minimize the variance of reliability estimator and reduce the overhead of estimating reliability. The presented approach combines the idea of statistical testing with the concepts of stratified sampling. Experiments are conducted to validate the efficiency of our approach.**

Keywords–Software reliability testing, reliability estimation, statistical testing, stratified sampling

I. Introduction

Statistical testing draws test cases from the expected Operational Profile (OP) according to the statistical distribution on the expected inputs.

Reliability assessment using statistical testing can be grouped in three different categories: (i) reliability growth models, (ii) fault seeding models and (iii) sampling models. Reliability growth models are making assumptions about the number of faults removed at each testing step by trying to extrapolate the future behavior of the software based on its past behavior. The assumptions made by reliability growth models are difficult to justify [1][2]. Fault seeding models are also making assumptions about the distribution of faults remaining in the program after testing. Such assumptions cannot be rigorously justified [3].

One class of reliability assessment approaches using statistical testing are sampling models. These models are theoretically sound [4], but they suffer from several practical problems. Sampling models require a large number of test cases [5]. In addition, a major concern is how to choose a proper estimator for the reliability that provides accurate and stable reliability estimate. The goodness of an estimator is judged based on the following four properties: (i) unbiased, (ii) minimum variance, (iii) consistent and (iv) sufficient. The main focus when selecting an estimator is the minimum variance of the estimator. The other three properties are in most of the cases satisfied by most of the estimators. The variance of an estimator describes the closeness of the future estimate to the previous estimate when rerunning the estimation with the same setting. An estimator with low variance increases the confidence on the predicted estimate. In fact, a low variance usually implies tighter confidence interval for the estimate. Consequently, we can improve the accuracy of the reliability estimation by providing or choosing an unbiased estimator that has a minimum variance. It is also important to note that the more tests are executed the more will the variance of the estimator decrease. Consequently, an estimator with low variance can find an accurate estimation with fewer test cases.

This paper presents a test selection strategy based on adaptive constrained sampling of the OP to deliver an unbiased reliability estimator which is both efficient and accurate (i.e., needs less test cases than standard approaches to find an accurate estimate). We call our approach Adaptive Constrained Test Selection (ACTS).

The rest of the paper is organized as follows. Section II formulates the problem of reliability estimation variance reduction and identifies the adaptive optimal test cases allocation over the operational profile sub-domains as a solution. The main steps of our approach are described in detail, in Section III. Experiments are set up to validate the performance of the proposed approach in Section IV. We give an overview on related work in Section V. Section VI concludes this paper and proposes future research direction.

II. Problem Formulation

The target of this paper is to present a reliability estimation approach that minimizes the variance of the reliability estimator for discrete-time domain software. For discrete-time domain systems, one is interested in the probability of success of a single usage of the software.

A. Software Statistical Testing

Software statistical testing is testing based on an operational profile.

Operational Profile Definition: As defined by Musa [6] ”an Operational Profile (OP) is a quantitative characterization of how a (software) system will be used”. It consists of a set of partitions of the software input domain (sub-domains) and their probabilities of occurrence.

In most of the cases, the OP describes also the distribution of the input variables of a program. We use in this work the abstract OP representation defined by Musa [6], which we introduce in Section II-B.

B. Standard Tests Selection Approach

Statistical Testing as proposed by Musa [6] generates by random sampling test cases according to the OP.

The OP is used to divide the input domain $\mathcal{D}$ of the software to test in L disjoint sub-domains: $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_L$. Each sub-domain represents a possible operational use and has a probability of occurrence according the OP. Let p_i be the probability of occurrence of sub-domain $\mathcal{D}_i$. The OP can be represented as $OP = \{(\mathcal{D}_i, p_i) | i = 1, 2, \ldots, L\}$.

Let $\mathcal{A}$ a sequence defined as follows: $\mathcal{A} = \{\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_L\}$, $|\mathcal{A}| = L + 1$, where $\mathcal{A}_i = \sum_{k=1}^{i} p_i$ for $i = 1, \ldots, L$, and $\mathcal{A}_0 = 0$.

The generation of the test cases is then as follows:

1) Generate an uniformly distributed random number $\zeta \in (0, 1)$, if $\zeta \in [\mathcal{A}_i, \mathcal{A}_{i+1}]$, then the sub-domain $\mathcal{D}_{i+1}$ will be randomly sampled since $\mathcal{A}_{i+1} - \mathcal{A}_i = p_{i+1}$, where p_{i+1} the probability of occurrence of sub-domain $\mathcal{D}_{i+1}$.
2) Generate input variables from the sub-domain $\mathcal{D}_{i+1}$ based on the provided input distributions, and execute the test case.
3) Repeat the above steps until a stopping criteria is reached (e.g, target reliability value reached, target confidence on the estimated reliability reached, required test time reached, etc.)

C. Discussion

The test selection approach proposed by Musa [6] is a random selection process without replacement. The selection is controlled by the uniformly distributed random variable $\zeta \in (0, 1)$. The main idea behind this approach is to ensure that when the testing process is terminated because of (for example) imperative software project constraints, then the most used operations will have received the most testing effort. Musa also claims that "the reliability level will be the maximum that is practically achievable for the given test time" [6]. One key assumption here is that the sample of selected test cases represents the expected software execution according to the OP and delivers the maximum achievable reliability level. However, this assumption is not always valid. It would be ideal if we could separate successful program execution from the failing ones. However, this is not likely, because failures are often caused by small faults in a large program.

A software fault is a hidden programming error in one or more program statements. A program consists of a set of statements. A program execution is a program path executed with an input value from the program's input domain. A program path is a sequence of statements. Each program path has an input and an executed output which usually depends on the input. Consequently, a program execution is considered as a failure if the corresponding executed program path deviates from the expected output.

Two program execution are similar if they execute the same program path with different input value. If the same input value is used then the two executions are equal.

Two similar program executions may differ only in regard to executing a particular fault, with the result that one execution fails while the other does not. Conversely, two dissimilar program execution may both fail because they execute the same faulty program statement. Consequently, we may not group the failing program executions together even if they have the same causing fault.

Hence, it is realistic to assume that the reliability estimate across the test sub-domains have different statistical properties (i.e., mean and variance). In this case, we refer to the sub-domains as heterogeneous sub-domains. Using conventional proportional random sampling to select test cases from heterogeneous sub-domains does not guarantee that a statistically sufficient number of test cases will be selected from every sub-domain. Hence, the statistical quality of the samples may be compromised for some sub-domains. This may lead to inaccurate statistical estimate.

Stratified sampling is designed to cluster a population made of heterogeneous groups into disjoint strata and then randomly sampling each strata. This paper addresses the problem of heterogeneity of the OP sub-domains by using optimal stratified sampling. The goal is to formulate the statistical testing approach as an optimal stratified random sampling process and provide a reliability estimator which should reduce the number of required test cases for the estimation while delivering accurate reliability estimates.

D. Stratified Sampling Variance Reduction

Stratified sampling is based on the idea of iterated expectations [7]. Let Y be a discrete random variable taking values $y_1, y_2, ..., y_L$ with probabilities $p_1, p_2, ..., p_L$. Let X be a discrete random variable. Then, $E[X] = E[E[X|Y]] = \sum_{l=1}^{L} E[X|Y = y_l]p_l$. Suppose that the population can be divided into $L > 1$ groups, known as **strata**. Suppose that a stratum l contains N_l units from the population ($\sum_{l=1}^{L} N_L = N$), and the value for the units in stratum l are $x_{1l}, x_{2l}, ..., x_{N_l l}$.

Let $W_l = \frac{N_l}{N}$ and $\mu_l = \frac{1}{N_l}\sum_{i=1}^{N_l} x_{il}$, then it follows that the population mean is $\mu = \frac{1}{N}\sum_{l=1}^{L}\sum_{i=1}^{N_l} x_{il} = \frac{1}{N}\sum_{l=1}^{L} N_l \mu_l = \sum_{l=1}^{L} W_l \mu_l$.

Then, instead of taking a simple random sample (SRS) of n units from the total population, we can take a SRS of size n_l from each stratum ($\sum_{l=1}^{L} n_l = n$). Here $\mu_l = E[X|stratum\ l]$ and $W_l = P[Stratum\ l]$, so the overall mean satisfies the setup of an iterated expectation.

Let $X_{1l}, X_{2l}, ..., X_{n_l l}$ be a sequence of independent and identically distributed random variables samples from stratum l, $\bar{X}_l = \frac{1}{n_l}\sum_{i=1}^{n_l} X_{il}$ be the sample mean, and ${S_l}^2 = \frac{1}{n_l - 1}\sum_{i=1}^{n_l}(X_{il} - \bar{X}_l)^2$ be the sample variance. Then, an estimate of the population mean μ is: $\bar{X}_S = \sum_{l=1}^{L} \frac{N_l}{N}\bar{X}_l = \sum_{l=1}^{L} W_l \bar{X}_l = \sum_{l=1}^{L} W_l \frac{1}{n_l}\sum_{i=1}^{n_l} X_{il}$. Since the random variables X_l are independent, then:
$\text{var}(\bar{X_S}) = \sum_{l=1}^{L} {W_l}^2 Var(\bar{X_l}) = \sum_{l=1}^{L} {W_l}^2 \frac{1}{n_l}(1 - \frac{n_l - 1}{N_l - 1}){\sigma_l}^2$, where $\sigma_l^2 = \frac{1}{N_l}\sum_{i=1}^{N_l}(x_{il} - \mu_l)^2$ is the variance of stratum l.

If we assume that $n_l \ll N_l$ for each stratum l so that the finite population factor $FPC = 1 - \frac{n_l - 1}{N_l - 1} \approx 1$ can be ignored, then:

$$\text{var}(\bar{X_S}) = \sum_{l=1}^{L} {W_l}^2 \frac{1}{n_l}{\sigma_l}^2 = \frac{1}{N}\sum_{l=1}^{L} {W_l}^2 \frac{{\sigma_l}^2}{a_l} \qquad (1)$$

where $a_l = n_l / N$ indicates the fraction of samples drawn from the stratum l.

This variance is controllable through the allocation ratio a_l. For example, the proportional allocation, where $a_l = W_l.N/N = W_l$, yields the variance $\text{var}(\bar{X_S}) = \frac{1}{N}\sum_{l=1}^{L} W_l {\sigma_l}^2$.

By Lagrange multiplier method, the optimal allocation $a^* := (a_1^*, \ldots, a_L^*)$ is derived in closed form

$$a_k^* = \frac{W_k.\sigma_k}{\sum_{l=1}^{L} W_l.\sigma_l} \tag{2}$$

achieving the minimal variance $\text{var}(\bar{X_S}) = \frac{1}{N}\sum_{l=1}^{L} {W_l}^2 \frac{{\sigma_l}^2}{a_l^*} = \frac{1}{N}(\sum_{l=1}^{L} W_l\sigma_l)^2$, [7].

Moreover, due to the mutual independence of samples across the strata, the empirical mean $\bar{X_S}$ is asymptotically normal [7].

E. Assumptions

In order to formulate the concerned research goal, some assumptions on the software are presented.

1) The software is frozen when estimating the reliability, since reliability estimation aims at testing the current status of the software. The software will not be modified during the estimation process. The software can be modified after the estimation process.
2) The output of each test is independent of the testing history. In some cases, it is possible that a test case is judged to be failure free although it actually leads to some faults which cannot be observed due to limited test oracles. We consider such test cases to be failure free. However, such unobserved faulty program states can cause the failure of some following test cases. Consequently, the latter test cases can be mistakenly considered as faulty test cases. This leads to an error in the reliability estimation. However, this is not a reliability estimation approach concern rather is a test oracle problem.
3) Each test case either fails or succeeds. A test oracle is used to verify the behavior of the software under test.
4) We assume that a proper test oracle is available, since this work focuses on the effectiveness and efficiency of reliability estimation.
5) In each operational use represented by a sub-domain $\mathcal{D}_i$, all possible software operations and possible inputs are equally likely to arise.
6) We assume that an OP is provided for the tested software.

III. Adaptive Constrained Test Selection

The $OP = \{(D_i, p_i) | i \in \{1, ..., L\}, \sum_{i=1}^{L} p_i = 1\}$ defines the expected input domain of the program's input variables. Each partition (D_l, p_l) is a subset of the OP, and $p_l \geq 0$ is the probability that a program input belongs to sub-domain D_l. The OP is a natural definition of the strata for stratified random sampling. Each stratum l corresponds to the sub-domain $\mathcal{D}_l$ and has a weight $W_i = p_l$.

A test case either fails or not. Consequently, each test case execution is a Bernoulli trial. Let X_{il} be the outcome of test case i from stratum l, i.e., from sub-domain $\mathcal{D}_l$, then:

$$X_{il} = \begin{cases} 1, & \text{if the test case fails} \\ 0, & \text{if the test case does not fail} \end{cases}.$$

Let μ_i defined as $P(\text{test cases from sub-domain } \mathcal{D}_i \text{ fail}) = \mu_i$, where $i = \{1, 2, \ldots, L\}$ and $\mu_i \in [0, 1]$.

Based on assumption 2, $\{X_{il}\}$ are independent random variables, and since $\sum_{i=1}^{L} p_i = 1$, then it can inferred that $P(X_{il} = 1) = \mu_i$ (i.e., the probability that test case i from sub-domain $\mathcal{D}_l$ fails). Each test case will lead the software under test to failure or success. And in each sub-domain the probability of failure of each test case is equal for all test cases in the sub-domain. Hence, the distribution of X_{il} is binomial distribution with μ_i.

Consequently, the sample mean of stratum l, $\bar{X}_l = \frac{1}{n_l}\sum_{i=1}^{n_l} X_{il}$ is an unbiased point estimator of μ_i.

The reliability of the tested software can be defined as the weighted sum of the reliability of the sampled OP sub-domains $\mathcal{D}_{i,i=\{1,\ldots,L\}} : R = \sum_{i=1}^{L} p_i(1 - \mu_i)$. An unbiased estimator of the reliability is then defined as:

$$\widehat{R} = 1 - \sum_{i=1}^{L} p_i \bar{X}_i = 1 - \sum_{l=1}^{L} \frac{1}{n_l}.p_l \sum_{i=1}^{n_l} X_{il} \tag{3}$$

with $E[\widehat{R}] = 1 - \sum_{i=1}^{L} p_i\mu_i$ and $\text{var}[\widehat{R}] = \sum_{i=1}^{L} {p_i}^2 \frac{\mu_i.(1-\mu_i)}{n_i} = \sum_{i=1}^{L} {p_i}^2 \frac{{\sigma_i}^2}{n_i}$, since the distribution of X_{il} is a binomial distribution with μ_i.

A. Optimal Test Cases Selection

The Problem of selecting the test cases optimally from the OP sub-domains is an adaptive optimization problem formulated as follows. Given the OP, we want to select a total number n of test cases, where (i) n_i test cases are selected from each sub-domains $\mathcal{D}_{i,i\in\{1,\ldots,L\}}$ and (ii) $\sum_{i=1}^{L} n_i = n$, with the goal to minimize $\text{var}[\widehat{R}]$. For mathematical tractability, we assume in this section that the total number of required test case n as well as the sub-domains failure rates and consequently their variances are known. These assumptions will be relaxed in the next sections. According to Section II-D:

$$n_i = n\frac{p_i\sigma_i}{\sum_{k=1}^{L} p_k\sigma_k} \tag{4}$$

Note that the larger the variance ${\sigma_i}^2$ of the failure rate of the software when executed with inputs from the sub-domain $\mathcal{D}_i$, the more test cases should be selected from that sub-domain. This makes sense, since the sub-domain with higher estimated/observed failure rate variability should require more testing to attain the same degree of precision as those with lower variability. If the variances of all sub-domains are all equal, the optimal allocation is proportional allocation.

B. Constrained optimal allocation

The intuition behind statistical testing is that the highest the probability of occurrence of a sub-domain, the larger the number of test cases executed from that sub-domain.

To account for this, the optimal allocation introduced in the previous section is formulated as a constrained optimization to a utility cost function c^*. Let $c_i = 1 - p_i$ the cost of selecting

```
1: if $\mathcal{SC}(\mathcal{T}_{OP}) \neq 1 \wedge \mathcal{SC}_{min} = \mathcal{SC}(\mathcal{T}_{(\mathcal{D}_k,p_k)}) < 0$ then
       // $\mathcal{T}_{(\mathcal{D}_k,p_k)}$ is over-proportional sampled
2:     $n = \lceil \frac{n_k}{p_k} \rceil$
3:     for $\mathcal{T}_{(\mathcal{D}_i,p_i)} \in \mathcal{T} \wedge \mathcal{T}_{(\mathcal{D}_i,p_i)} \neq \mathcal{T}_{(\mathcal{D}_k,p_k)}$ do
4:         $n_i = \lceil n.p_i \rceil$
5: //select extra $(\lceil n.p_i \rceil - n_i)$ test cases
6:     end for
7: else
8:     for $\mathcal{T}_{(\mathcal{D}_i,p_i)} \in \mathcal{T}$ do
9:         $n_i = \lceil n.p_i \rceil$
10:    end for
11: end if
```

Figure 1. Adjust to Proportional Sampling

a test case from a sub-domain $\mathcal{D}_i$ that has a probability of occurrence p_i, Then

$$n_i = c^*.\frac{p_i\sigma_i/\sqrt{c_i}}{\sum_{k=1}^{L} p_k\sigma_k/\sqrt{c_k}} \tag{5}$$

The cost function c^* is defined in Section III-D.

Note that the higher the cost c_i of selecting a test case from sub-domain $\mathcal{D}_i$, the smaller the sub-domain sample size n_i.

Since the cost function c_i is defined as $c_i = 1 - p_i$, then (5) means: the smaller the probability of occurrence of a sub-domain $\mathcal{D}_i$, the smaller the sample size n_i.

C. Similarity Confidence

When testing a software according to an OP, the goal is to simulate the expected software execution as described by the OP. Consequently, it is interesting to quantify the similarity of the total set of selected test cases to the expected OP. It is also interesting to control the testing process toward a 100% similarity to the OP.

Let $\mathcal{T}_{(\mathcal{D}_i,p_i)}$ be the set of test cases selected from the sub-domain $(\mathcal{D}_i,p_i)_{\{i\in 1,\dots,L\}}$. Let $|\mathcal{T}_{(\mathcal{D}_i,p_i)}| = n_i$, i.e., the set $\mathcal{T}_{(\mathcal{D}_i,p_i)}$ contains n_i different test cases selected from the sub-domain $(\mathcal{D}_i,p_i)_{\{i\in 1,\dots,L\}}$. Let $\mathcal{T}_{OP} = \{\mathcal{T}_{(\mathcal{D}_i,p_i)} | (\mathcal{D}_i,p_i) \in OP = \{(\mathcal{D}_i,p_i) | i \in \{1,\dots,L\}, \sum_{i=1}^{L} p_i = 1\}\}$ the set of selected test cases from the OP. The similarity of $\mathcal{T}_{(\mathcal{D}_i,p_i)}$ to the OP when a total number $n = |\bigcup_{(\mathcal{D}_i,p_i)\in OP} \mathcal{T}_{(\mathcal{D}_i,p_i)}| = |\mathcal{T}_{OP}|$ of test cases is selected from the OP sub-domains, is defined as follows:

$$\mathcal{SC}(\mathcal{T}_{(\mathcal{D}_i,p_i)}) = \begin{cases} \frac{n_i}{\lceil p_i.n \rceil}, & \text{if } n_i \leq \lceil p_i.n \rceil \\ -\frac{n_i}{\lceil p_i.n \rceil}, & \text{if } n_i > \lceil p_i.n \rceil \end{cases} \tag{6}$$

The similarity confidence of the total selected test cases is consequently defined as follows: $\mathcal{SC}(\bigcup_{(\mathcal{D}_i,p_i)\in OP} \mathcal{T}_{(\mathcal{D}_i,p_i)}) = \frac{\sum_{i=1}^{L} \mathcal{SC}(\mathcal{T}_{(\mathcal{D}_i,p_i)})}{L}$

Let $\mathcal{SC}_{min} = min\{\mathcal{SC}(\mathcal{T}_{(\mathcal{D}_i,p_i)}) | i \in \{1,\dots,L\}\} = \mathcal{SC}(\mathcal{T}_{(\mathcal{D}_k,p_k)})_{k\in\{1,\dots,L\}}$, the minimum computed similarity to the OP.

Algorithm 1, adjusts the allocation of the test cases from each sub-domain $(\mathcal{D}_i,p_i)_{\{i\in 1,\dots,L\}}$ to reach a similarity confidence of 100%. The steps of the algorithm are as follows. If the selected tested cases $\mathcal{T}_{OP}$ is not similar to the OP and if $\mathcal{SC}_{min} = SC((D_k,p_k))$ is negative (line 1), then it means that the sub-domain $(\mathcal{D}_k,p_k)$ is over proportionally sampled. In this case, the total number of test case n is updated proportionally to n_k (line 3), and for each sub-domain except the sub-domain $(\mathcal{D}_k,p_k)$, extra $(\lceil n.p_i \rceil - n_i)$ test case are selected (lines 4-6).

Otherwise, the sub-domains are under proportionally sampled, and for each sub-domain $(\mathcal{D}_i,p_i)$, extra $(\lceil n.p_i \rceil - n_i)$ test case are selected (lines 8-9).

D. Stopping Criteria

We define a test stopping criteria based on the tester required (i) maximal error of the reliability estimate d, and (ii) confidence level $(1-\alpha)$. The goal of reliability testing is then to estimate the reliability $\hat{R}$ to within d with $100(1-\alpha)\%$ probability.

The total required number of test cases depends on the allocation of the selected test cases to the sub-domains. Let a_l (as defined in Section II-D) be the allocation ratio for the sub-domain $\mathcal{D}_l$, with $n_l = n.a_l$. Also, let z be the upper $\alpha/2$ critical point of the standard normal distribution. Then, we want to find n such that $z[\text{var}[\widehat{R}]]^{1/2} = d$ (margin of error equation), where $\text{var}[\widehat{R}] = \sum_{i=1}^{L} {p_i}^2 \frac{{\sigma_i}^2}{n_i} = \frac{1}{n}.\sum_{i=1}^{L} {p_i}^2 \frac{{\sigma_i}^2}{a_l}$.

Solving the margin of error equation for n leads to: $n = \frac{z^2}{d^2} \sum_{i=1}^{L} {p_i}^2 \frac{{\sigma_i}^2}{a_l}$.

From (5), we can compute the total cost c^* required to achieve the desired level of accuracy as follows [7]:

$$a_l = \frac{p_l.\sigma_l/\sqrt{c_l}}{\sum_{i=1}^{L} p_i.\sigma_i.\sqrt{c_i}} \text{ and } c^* = \frac{z^2}{d^2}\left[\sum_{i=1}^{L} p_i.\sigma_i.\sqrt{c_i}\right]^2. \tag{7}$$

From here, we can compute $n_l = c^*.a_l$, and then, ultimately n.

E. Adaptive Constrained Test Selection Algorithm

Based on the discussions above, the adaptive constrained test selection algorithm works as described in Algorithm 2. In the intialization phase of the algorithm (lines 6-7), $|\mathcal{T}_{(\mathcal{D}_i,p_i)}|$ test case are selected from each sub-domain $(\mathcal{D}_i,p_i)$ based on a given initial number of test case n_{start}. $\mathcal{T}_{(\mathcal{D}_i,p_i)}$ represents the set of test cases selected from sub-domain $(\mathcal{D}_i,p_i)$. In the sampling phase (lines 10-27), the algorithm computes for each sub-domain the optimal required number of test cases to be select based on the stopping criteria formula in equation 7 (line 12-13). Extra test cases are then selected if required (lines 15-16). Otherwise, test cases have been optimally selected from that sub-domain (line 18). The algorithm computes the variance of the observed failure rate for each sub-domain after each sampling phase (line 24), and adjust the test allocation toward 100% similarity to the OP. The algorithm stops and returns the estimated reliability if (i) a maximal allowed test time interval Δ has passed or (ii) for all sub-domains the optimal required number of test cases has been selected and the total selected test cases are 100% similar to the OP (line 21).

```
Require: $OP = \{(\mathcal{D}_i, p_i) | i \in \{1, ..., L\}, \sum_{i=1}^{L} p_i = 1\}$
        $\mathcal{T}_{OP} = \{\mathcal{T}_{(\mathcal{D}_i,p_i)} | (\mathcal{D}_i, p_i) \in OP\}$
        $\Delta$ : maximal allowed test time
        $n_{\text{start}}$ : initial number of test cases to start
        $1 - \alpha$ : confidence level
        $d$ : margin of error
for $(\mathcal{D}_i, p_i) \in OP$ do // 1. Initialization
    $|\mathcal{T}_{(\mathcal{D}_i,p_i)}| \leftarrow \lceil n_{\text{start}}.p_i \rceil$

end for
    //2. Adaptive optimal constrained stratification
while true do
    for $(\mathcal{D}_i, p_i) \in OP$ do
        $c^* = \frac{z^2}{d^2} \left[ \sum_{i=1}^{L} p_i.\sigma_i.\sqrt{(1 - p_i)} \right]^2$
        $a_i = \frac{p_i.\sigma_i/\sqrt{(1-p_i)}}{\sum_{k=1}^{L} p_k.\sigma_k.\sqrt{(1-p_k)}}$
        $n_i^o = \lceil c^* a_i \rceil$
        if $|\mathcal{T}_{(\mathcal{D}_i,p_i)}| < n_i^o$ then
      //select extra $(n_i^o - |\mathcal{T}_{(\mathcal{D}_i,p_i)}|)$ test cases from $(\mathcal{D}_i, p_i)$
            $|\mathcal{T}_{(\mathcal{D}_i,p_i)}| \leftarrow n_i^o$
        else
            opt = opt + 1
        end if
    end for
    if $\Delta$ passed or $(\text{opt} = L \wedge \mathcal{SC}(\mathcal{T}_{OP}) = 100\%)$ then
        break;
    end if
    update statistics for all $(\mathcal{D}_i, p_i)$
    Adjust to proportional sampling: call Algorithm 1
    opt = 1
end while
return $\widehat{R} = \sum_{i=1}^{L} p_i.\widehat{R}_i$
```

Figure 2. Adaptive constrained test cases selection

IV. Experimental Evaluation

We conduct a set of experiments on a real subject program to evaluate the performance of the Adaptive Constrained Test Selection (ACTS) algorithm against the standard proportional test selection approach as proposed by Musa [6] (PS), and the theoretical optimal test selection approach (OS) with respect to the estimated reliability accuracy and precision. For (OS), we assume that we know the failure rates in advance, and we sample accordingly.

A. Experiment Design and Setup

1) Subject Program and Operational Profiles: Space: a language-oriented user interface developed by the European Space Agency. It allows the user to describe the configuration of an array of antennas with a high level language. The correct version as well as the 38 faulty versions and a test suite of $13,585$ test cases are downloaded from the software-artifact infrastructure repository [8]. In these experiments, three faulty versions are not used because we did not find test cases that failed on these faulty versions. Space is 9126 LOCs big.

A failure of an execution is determined by comparing the outputs of the faulty version and the correct version of the program. A failure is a deviation from the expected output. The failure rates for both studied programs are empirically computed by executing all the available test cases against each faulty version of a program and recording the number of failed test cases.

Operational profiles for Space are not available. We create operational profiles for Space as follows. We assume that in each sub-domain $\mathcal{D}_i$, all possible inputs are equally likely to arise. Hence, it follows that the number of sub-domains (greater or equal to two sub-domains) as well as the number of inputs in each sub-domain may not bias the statistical properties (i.e., variance and mean) of the estimated reliability. The estimated reliability is influenced by the probability of occurrence of the sub-domains, as well as the true failure rate of the tested software when executed with inputs from each sub-domain. In that sense, we partition the test cases of Space in six disjoint sub-domains. All six sub-domains contain the same number of test cases except for rounding issues. For each sub-domain, test cases are randomly selected without replacement from the pool of test cases. The $13,585$ test cases of Space are partitioned into six disjoint classes: 2264, 2264, 2264, 2264, 2264 and 2265. In order to minimize possible bias due to the choice of the test cases in each sub-domain, we repeat the allocation of the test cases of each subject program into the six sub-domains twice. This results into 2 possible allocations of the test cases to sub-domains $\mathcal{D}_i$ for each subject program.

Due to time and space limitation, not all possible operational profiles can be adopted in the experiments. We define two different profiles for the probability of occurrence of the sub-domains: (i) uniform profile: the probability of occurrence of each sub-domain is the same except for rounding error and (ii) optimal profile: the probability of occurrence of each sub-domain is proportional to the number of test cases allocated to each sub-domain using optimal allocation

These two profiles are some typical or extreme profiles and cannot represent all usage scenarios in field use. Consequently, for each subject program, 4 different operational profiles are created.

2) Performance Metrics: ACTS, PS and OS are randomized test selection strategies. For statistical significance, we conduct 200 independent repetitions of each experiment for each test selection strategy.

We compare the performances of ACTS, PS and OS by comparing the accuracy and precision of the estimated reliability by each approach. The accuracy of an estimate is a measure of how close the estimated value is to its true value. The precision of an estimate is a measure of how close the estimates measured from different samples are to another, when the samples are taken from the same data set. We use the sample variance as metric for the reliability estimation accuracy. The sample variance is an unbiased estimator of the variance. We use the *root mean squared error* (RMSE) to quantify the estimate precision.

Based on assumption 5 in Section II-E, the reliability estimates delivered by ACTS, PS, and OS are unbiased. Consequently, we can compare the relative efficiency of the estimates using the sample variance. For each experiment $\mathcal{E}$ we define the mean value of the reliability estimate ($\overline{\widehat{R}}$), its sample variance ($S_{199}^2(\widehat{R})$), its root mean squared error ($RMSE(\widehat{R})$), and the relative efficiency of the reliability estimator using ACTS to PS and OS as follows:

$$\overline{R} = \frac{1}{200}\sum_{i=1}^{200}\widehat{R_i}, \quad S^2_{199}(\hat{R}) = \frac{1}{199}\sum_{i=1}^{200}(\widehat{R_i}-\overline{R})^2, \quad RMSE(\hat{R}) = \sqrt{\frac{1}{200}\sum_{i=1}^{200}(\widehat{R_i}-R)^2}$$

$$\text{eff}(\hat{R}_{ACTS}, \hat{R}_{PS}) = \frac{RMSE(\hat{R}_{PS})}{RMSE(\hat{R}_{ACTS})}, \quad \text{eff}(\hat{R}_{ACTS}, \hat{R}_{OS}) = \frac{RMSE(\hat{R}_{OS})}{RMSE(\hat{R}_{ACTS})}$$

where R is the true reliability calculated based on the true failure rates, $\widehat{R_i}$ the reliability estimate in repetition i of the experiment, $\widehat{R}_{ACTS}$ the reliability estimate using ACTS, $\widehat{R}_{PS}$ the reliability estimate using PS and $\widehat{R}_{OS}$ the reliability estimate using OS.

The differences in reliability mean values between the different test selection strategies is confirmed using the non-parametric Mann-Whitney U test [9]. The differences between the sample variances are tested using the Brown-Forsythe test [9].

For each experiment and for each test selection strategy, we compute the reliability estimate at seven checkpoints: $200, 250, 350, \ldots, 500$. After 200 repetitions of the experiment, we compute the mean value, sample variance and the root mean square error of the reliability estimates for each test selection strategy. Note that the more test cases are executed the more will the variance of the estimator decrease. In addition, the experimental dataset is selected randomly from the population and the selection is repeated 200 times. Consequently, the selected dataset do not affect the efficiency and the generalizability of ACTS.

B. *Experimental Results*

The goal of this set of experiments is to assess the efficiency and precision of our reliability estimation approach.

Figure 3 presents the sample means and sample variances for Space. The dashed lines are the true reliability values for the subject programs.

According to the experimental results, the means as well as the sample variances of the reliability estimates of ACTS are closer to the true values than those of PS and OS. This is confirmed by the statistical tests Mann-Whitney U test and Brown-Forsythe test in table I. The table confirms that ACTS significantly deliver more accurate reliability estimate that PS and OS.

The computed mean of the relative efficiency of the reliability estimator using ACTS compared to the one using PS for the Space experiments was $1,57$. This means that PS will yield a reliability estimate as accurate as ACTS only if 57% more test cases are selected.

The computed mean of the relative efficiency of the reliability estimator using ACTS compared to the one using OS for the Space experiments was $1,32$. This means that OS will yield a reliability estimate as accurate as ACTS only if 32% more test cases are selected.

V. Related Work

Stratified sampling is linked to the idea of partition testing or sub-domain testing of a software. Techniques to estimate software reliability using partition testing are proposed by Brown and Lipow [10] and Duramn and Wiorkowski [11], for example. They introduced the idea of sampling to reliability estimation but did not specify a sampling design. Podgurski et al.'s [12] work of is the most related work to our research. However, they only used the idea of equal stratification using clustering to estimate the software reliability from software execution profiles collected by capture/replay tools. Failure rates have been extensively used in the area of adaptive random testing (for example Cangussu et al.'s [13] and Chen et al.'s [14]). Adaptive random testing aims to distribute the selected test cases as spaced out as possible to increase the chance of hitting the failure patterns. The intuition behind adaptive random sampling can be added in a future work to our approach to probably further enhance the efficiency of the reliability estimator. Cangussu et al.'s [13] and Chen et al.'s [14] do not address the problem of reliability estimator efficiency.

Thévenod-Fosse and Waeselynck [15] present the usage of probabilistic test generation for fault detection. They generate automatically tests to address different behavioral and structural test criteria. Apparently, Thévenod-Fosse and Waeselynck [15] view the evaluation of tests as *inexpensive*. They call their approach "statistical testing" although it dos not involve reliability estimation. In contrast to Thévenod-Fosse and Waeselynck [15], we think that evaluating test is an expensive process. Our approach aims to reduce the variance of a reliability estimator and consequently reduce the required number of executed and evaluated test cases to reach a target reliability confidence. A recent work on adaptive testing by Junpeng and Cai [16], allocates test cases using a gradient search method based on the variance variation of the failure rate. However, their approach introduces bias resulting from the use of the gradient method: it is possible that all test cases are selected from the sub-domain that first reveals a failure. They avoid such situations by introducing a biased estimator using Bayesian estimation.

TABLE I. Mann-Whitney U and Brown-Forsythe test results for the sample means and variances for Space

		Variance		Mean	
Scenarios		ACTS	OS	ACTS	OS
Space profile1	PS	0/7	4/7	6/7	0/7
	OS	0/7	-	7/7	-
Space profile2	PS	0/7	1/7	7/7	7/7
	OS	1/7	-	7/7	-
Space profile3	PS	1/7	1/7	7/7	5/7
	OS	1/7	-	5/7	-
Space profile4	PS	0/7	1/7	5/7	1/7
	OS	2/7	-	6/7	-

VI. Conclusions and Future Work

Statistical testing is in the most of the cases impractical due to the large number of test cases required to reach a target reliability. In this paper, we presented an approach to automatically select test cases from an operational profile sub-domains with the goal to reduce the variance of the reliability estimator. Our initial experimental results are promising and shows that our approach ACTS outperforms PS and OS.

We plan to conduct further experiments to validate the

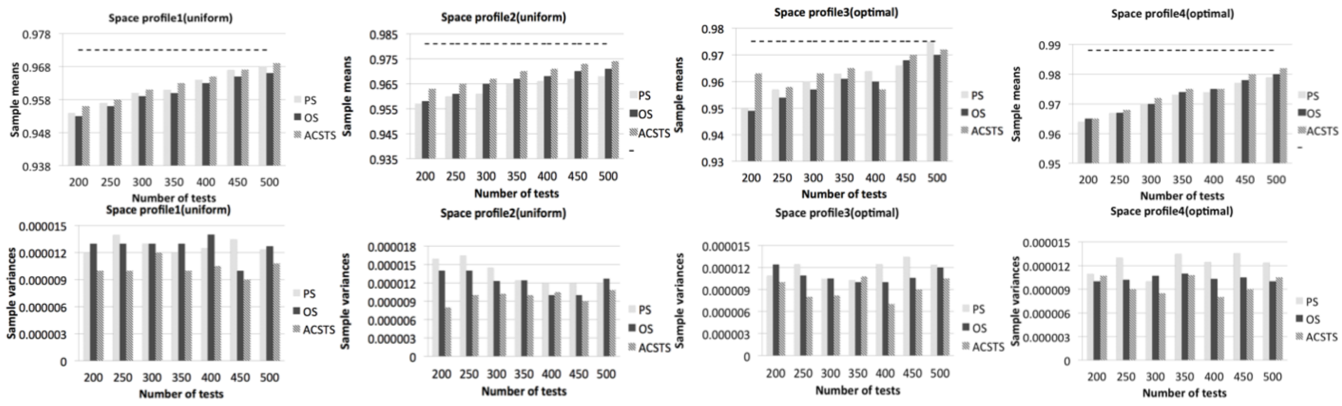

Figure 3. Sample means and sample variances of the reliability estimates for Space

effectiveness of ACTS on software with real specified operational profiles. We also plan to further investigate the efficiency of our approach for ultra-high software reliability scenarios.

Acknowledgment

This work was partially supported by the German Federal Ministry of Economics and Energy (BMWI), grant No. 01MD11005 (PeerEnergyCloud).

References

[1] M.-H. Chen, M. Lyu, and W. Wong, “Effect of code coverage on software reliability measurement,” Reliability, IEEE Transactions on, vol. 50, no. 2, 2001, pp. 165–170.

[2] P. K. Kapur, D. N. Goswami, and A. Bardhan, “A general software reliability growth model with testing effort dependent learning process,” Int. J. Model. Simul., vol. 27, no. 4, Sep. 2007, pp. 340–346.

[3] C. V. Ramamoorthy and F. B. Bastani, “Software reliability status and perspectives,” IEEE Trans. Softw. Eng., vol. 8, no. 4, Jul. 1982, pp. 354–371.

[4] T. A. Thayer, M. Lipow, and E. C. Nelson, Software reliability : a study of large project reality, ser. TRW Series of software technology; 2. Amsterdam: North-Holland, 1978.

[5] R. W. Butler and G. B. Finelli, “The infeasibility of quantifying the reliability of life-critical real-time software,” IEEE Trans. Softw. Eng., vol. 19, no. 1, Jan. 1993, pp. 3–12.

[6] J. D. Musa, “Operational profiles in software-reliability engineering,” IEEE Softw., vol. 10, no. 2, Mar. 1993, pp. 14–32.

[7] W. Cochran, Sampling techniques, ser. Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley, 1977.

[8] “The software-artifact infrastructure repository,” http://sir.unl.edu, accessed: 2014-08-30.

[9] S. Wilks, Mathematical Statistics. Read Books, 2008.

[10] J. R. Brown and M. Lipow, “Testing for software reliability,” SIGPLAN Not., vol. 10, no. 6, Apr. 1975, pp. 518–527.

[11] J. W. Duran and J. J. Wiorkowski, “Quantifying software validity by sampling,” Reliability, IEEE Transactions on, vol. R-29, no. 2, June 1980, pp. 141–144.

[12] A. Podgurski, W. Masri, Y. McCleese, F. G. Wolff, and C. Yang, “Estimation of software reliability by stratified sampling,” 1999.

[13] J. W. Cangussu, K. Cooper, and W. E. Wong, “A segment based approach for the reduction of the number of test cases for performance evaluation of components,” International Journal of Software Engineering and Knowledge Engineering, vol. 19, no. 04, 2009, pp. 481–505.

[14] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T. H. Tse, “Adaptive random testing: The art of test case diversity,” J. Syst. Softw., vol. 83, no. 1, Jan. 2010, pp. 60–66.

[15] P. Thévenod-Fosse and H. Waeselynck, “Statemate applied to statistical software testing,” in Proceedings of the 1993 ACM SIGSOFT International Symposium on Software Testing and Analysis, ser. ISSTA ’93. New York, NY, USA: ACM, 1993, pp. 99–109.

[16] B.-B. y. Junpeng Lv and K. yuan Cai, “On the asymptotic behavior of adaptive testing strategy for software reliability assessment,” Transaction on software Engineering, vol. 40, no. 4, April 2014, pp. 396–412.