

SEQUENTIAL SAMPLING DESIGNS FOR ESTIMATING
SOFTWARE RELIABILITY

A DISSERTATION
IN
Mathematics
and
Physics

Presented to the Faculty of the University
of Missouri–Kansas City in partial fulfillment of
the requirements for the degree

DOCTOR OF PHILOSOPHY

by
BADER ALANAZI

B.S. Al Qassim University, 2010
M.S. University of Missouri-Kansas City, 2014

Kansas City, Missouri
2020

© 2020
BADER ALANAZI
ALL RIGHTS RESERVED

SEQUENTIAL SAMPLING DESIGNS FOR ESTIMATING SOFTWARE RELIABILITY

BADER ALANAZI, Candidate for the Doctor of Philosophy Degree
University of Missouri–Kansas City, 2020

ABSTRACT

For any non-trivial system, it is impossible to reach the exact reliability of software due to the complexity, cost, and time required to complete the testing. Instead, a sample of test cases can be used to estimate the overall software reliability. Our objective is to obtain the most accurate estimate of software reliability by allocating test cases among partitions.

In the traditional approach, the method of allocating test cases among partitions is determined before reliability testing begins. By allocating test cases in advance, there is no opportunity to take advantage of the errors in choosing the distributions of test cases that may occur during the testing of the software. The inability to use these errors to adjust the estimate during testing is a shortcoming of a fixed sampling scheme.

We applied sequential sampling schemes to make allocation decisions dynamically throughout the testing process. Under these sampling schemes, we

can refine the allocation of test cases sequentially based on the information gained as the testing proceeds. Using theoretical results and Monte Carlo simulation, we have shown that the proposed sequential sampling scheme performs at least as well as the balanced sampling scheme.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the College of Graduate Studies, have examined a dissertation titled “ “Sequential Sampling Designs for Estimating Software Reliability,” presented by Bader Alanazi, candidate for the Doctor of Philosophy degree, and certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Kamel Rekab, Ph.D., Committee Chair
Department of Mathematics and Statistics

Paul Rulis, Ph.D., Co-discipline Advisor
Department of Physics and Astronomy

Majid Bani-Yaghoub, Ph.D.
Department of Mathematics and Statistics

Rhee Noah, Ph.D.
Department of Mathematics and Statistics

Da-Ming Zhu, Ph.D.
Department of Physics and Astronomy

Contents

ABSTRACT	iii
TABLES	viii
ACKNOWLEDGEMENTS	ix
Chapter	
1 INTRODUCTION	1
2 A FULLY SEQUENTIAL SAMPLING SCHEME FOR SOFTWARE RELIABILITY ESTIMATION	6
2.1 Introduction	6
2.2 Estimating Software Reliability	7
2.3 Optimal Sampling Scheme	13
2.4 Sequential Sampling Scheme	16
2.5 Fully Sequential Sampling Scheme	19
2.6 Comparison	21
2.7 Theoretical Result	22
2.8 Monte Carlo Simulations	24
3 A Two-Stage Sampling Scheme for Software Reliability Estimation .	27
3.1 Introduction	27
3.2 Sequential Sampling Scheme	28

3.3	Two Stage Sampling Scheme	30
3.4	Theoretical Result	33
3.5	Monte Carlo Simulations	35
4	Summary and Conclusion	37
A	Tables	40
Appendix		
VITA	62

TABLES

Tables		Page
1	Comparison of Optimal , Fully, and Balanced Allocation Model	41
2	Comparison of Fully Allocation, and Balanced Allocation Model	42
3	Comparison of Optimal , Two Stage, and Balanced Allocation Model	43
4	Comparison of Fully Allocation, Two-Stage Allocation Model .	44
5	Comparison of Two Stage Allocation, and Balanced Allocation Model	45
6	Comparison of Optimal , Fully, Two-Stage, and Balanced Model	46
7	Comparison of Highly Reliability	47
8	Comparison of Fully Allocation, and Balanced Allocation Model with the percentage of n_1	48
9	Comparison of Two Stage Allocation, and Balanced Allocation Model with the percentage of n_1	49
10	Comparison of Optimal and Sequential Allocation Model . . .	50

ACKNOWLEDGEMENTS

I would like to show my greatest appreciation to Dr. Rekab. Without his guidance and persistent help, this dissertation would not have been possible. Dr. Rekab not only was a supervisor in my academic life, he also supported me through the most difficult circumstances. That he was beside me in the hospital when my son had open-heart surgery sums up the difference and impact he has had on my life. It is my pleasure to be under his supervision.

I would like to express my gratitude to Dr. Paul Rules, Dr. Majid Bani-Yaghoub, Dr. Rhee Noah, and Dr. Da-Ming Zhu for their kindly agreed to be on my doctoral committee.

I cannot express my thanks to my mom for her love, prayers, and continuing support over my life. All thanks also go to my brothers, my sisters, and my cousins. Special thanks are for my brother Ahmed, my cousin Khalaf, and my friend Abdulelah for all their support and encouragement.

I give my deepest thanks and gratitude to my wife and sons, Abdulelah and Azam for being in my life and for unending patience as I completed this achievement.

CHAPTER 1

INTRODUCTION

Software reliability plays an essential role in building confidence in software quality assurance. Software reliability is an essential parameter of evaluating systems and determining whether a program meets specified requirements of the system. In critical systems, the level of reliability must be high; that is, the level of failure should not exceed the order of 10^{-3} [11]. Software is most often tested late in the product development cycle despite that a product should not be launched on the market before ensuring reliable functioning. The outcomes from measuring software reliability have significant impacts on decision-makers who must determine whether and when to release a product.

Highly reliable software benefits some industries and academic organizations, such as universities and information technology firms, as well as in the medical and health organizations. To meet the demands of technology development and organizations' goals, many organizations depend critically on software to perform tasks effectively and efficiently; thus, enabling these entities to carry out their missions with confidence. Unreliable software could lead to disastrous consequences, especially in situations where software failure costs lives. For example, unreliable software in medical devices can neg-

actively impact patient safety [30]. Thus, accurate estimations of software reliability can support significant reductions in risk and costs. The purpose of this research is to present sequential sampling methods for testing software reliability.

Researchers have investigated the testing of software using many distinct testing strategies, such as functional testing [9], mutation testing [5], partition analysis [26], and among others [5, 18]. These testing strategies are based on forms of sub-domain testing. In sub-domain testing, the input domain of the program is divided into sub-domains that may overlap in some strategies [7]. Some strategies require that the input domain must be divided into non-overlapping sub-domains which is referred to as partition testings [32, 33, 16]. Thus, partition testing may be viewed as stratified random sampling [4, 15].

In the approach presented for this study, we focus on a partition testing strategy in which the input domain of all possible test cases is broken into $K > 2$ partitions. The methods of creating partitions are not the focus of this research; however, several researchers have studied these methods [8, 28, 33]. The partitions $\mathbf{D}_1, \mathbf{D}_2, \dots, \mathbf{D}_k$ must be mutually disjoint and cover the whole domain \mathbf{D} such that:

$$(i) \mathbf{D} = \bigcup_{i=1}^k \mathbf{D}_i$$

$$(ii) \mathbf{D}_i \cap \mathbf{D}_j = \emptyset, i \neq j$$

This form of partitioning is known as non-overlapping where we can test each partition independently of the others. The independence here is that if test case j is drawn from partition i , then all other partitions selected will not contain test case j . Also, there should be an equal chance to select each test case from partition i . The goal is to allocate the test cases among partitions to improve the estimated reliability of the software.

The operational profile is a crucial factor in determining how to allocate test cases among partitions. Operational profile is defined as the quantitative characterization of how a system will be used [12]. When using an operational profile, attention is directed toward the operations that are used more frequently by the software. In this approach, larger fractions of the test cases are allocated to the most frequently used functions of a system. We can test software reliability efficiently by using the operation profile $, p_i$, where the values of p_i are specified for each sub-domain [13, 14]. In our method, We assume that the parameters of the operational profile p_1, p_2, \dots, p_k are known. There are several approaches for estimating the probabilities of an operational profile, such as the usage models and Markov chain models [31, 34]; however, it is not the goal of this research to focus on computing the probabilities of operational profile.

Software reliability is defined as the probability that the software will give the correct result for a single randomly chosen use [17]. In our method, this

selected use will be according to the operational profile. Thus, the overall software reliability can be represented by:

$$\mathbf{R} = \sum_{i=1}^k P_i R_i$$

see [1, 28, 22]. Based on the definition, the reliability of a software can be determined by two categories:

- (i) The number of successes and faults in each sub-domain R_i .
- (ii) The operational profile P_i for each partition.

Because the reliability of the software for each partition, R_i , is unknown, the optimal sampling scheme is theoretical and not practical. Thus, we aim to find a model that is as close as possible to the optimal model.

In software testing, exhaustive testing refers to the testing of all possible types of inputs and paths. Unfortunately, exhaustive testing to reach a precise level of software reliability is not possible for any non-trivial system due to the complexity, cost, and time required to complete [10]. Instead, a sample of test cases can be executed to estimate the overall software reliability. We have one constraint for the sample size in the proposed method. We assume that the total number of test cases is fixed. Mathematically, $n_1 + n_2 + \dots + n_k = N$ where n_1, n_2, \dots, n_k are sample sizes that are drawn from sub-domains D_1, D_2, \dots, D_k , respectively. The approach differs from oth-

ers [11, 3] in which the goal is to meet a specified level of reliability. Under the approach requiring a specific reliability level, the testing halts if the system executes n test cases and achieves the required goal, otherwise testing continues. Thus, the sample size is not fixed in methods using a specific goal for reliability. Alternatively, our objective is to reach the most accurate estimate by sampling a fixed number of test cases from the software.

In a fixed sampling scheme, the method of allocating test cases among partitions is determined before reliability testing begins. By allocating test cases in advance, there is no opportunity to take advantage of the errors that occur throughout the testing of the software. The inability to use these errors to adjust the estimate during testing is a shortcoming of a fixed sampling scheme. Therefore, we will use sequential sampling schemes to estimate reliability as we test software. Under these sampling schemes, we can refine the allocation of test cases sequentially based on the information gained as the testing proceeds. As the process continues, the remaining test cases are distributed based on previous testing results. We aim to allocate these test cases among the partitions of the software to minimize the variance of the reliability estimate. Through Monte Carlo simulation and theoretical results, we will test whether the proposed sampling scheme performs significantly better than the fixed sampling scheme.

CHAPTER 2

A FULLY SEQUENTIAL SAMPLING SCHEME FOR SOFTWARE RELIABILITY ESTIMATION

2.1 Introduction

In this section, we introduce a sequential sampling scheme to test software by estimating software reliability. We applied a method that permitted refinement of the process of allocating test cases among partitions of the software. This method of sequential allocation supports the development of allocation decisions that are carried throughout the test process. The process aimed to minimize the variance of the overall estimated reliability of the software system via distributing the total number of test cases N into K partitions. The expectation is that a fully sequential sampling scheme performs better than balanced sampling in which the total number of test cases are allocated in advance. We present the theoretical results and Monte Carlo simulations to show that the proposed sampling scheme performs better than the balanced sampling scheme.

2.2 Estimating Software Reliability

In the context of software testing, reliability provides a measure of the probability that the software runs without failure; thus, reliability always takes the value of 0 or 1. We modeled the outcome of a functional unit as 1, while the outcome of a non-function unit as 0, as seen by

$$\mathbf{X}_{ij} = \begin{cases} 1 & \text{,if the } j^{th} \text{ test case of } i^{th} \text{ partition succeeds.} \\ 0 & \text{,if the } j^{th} \text{ test case of } i^{th} \text{ partition fails.} \end{cases}$$

, where X_{ij} is the outcome of the j^{th} test case of i^{th} .

Because the outcome is binary, it is obvious that X_{ij} follows Bernoulli distribution with parameter R_i , which is the software reliability estimator for partition i . After allocating n_i test cases to partition i , we define \hat{R}_i as the estimator for R_i . The maximum likelihood estimator of each partition R_i with n_i test cases is determined as follow:

$$\mathbf{f}(\mathbf{X}_{ij}) = \begin{cases} R_i & \text{,if } X_{ij} = 1 \\ 1 - R_i & \text{,if } X_{ij} = 0 \end{cases}$$

The reliability mass function of each X_{ij} is

$$f(X_{ij}) = R_i^{X_{ij}} (1 - R_i)^{1 - X_{ij}} \quad (2.1)$$

The likelihood function $L(R_i)$ is given by :

$$\begin{aligned} L(R_i, X_{i1}, \dots, X_{in_i}) &= \prod_{j=1}^{n_i} f(X_{ij}) \\ &= R_i^{X_{i1}} (1 - R_i)^{1 - X_{i1}} \dots R_i^{X_{in_i}} (1 - R_i)^{1 - X_{in_i}} \end{aligned} \quad (2.2)$$

We can simplify (2.2) by summing up the exponents :

$$L(R_i, X_{i1}, \dots, X_{in_i}) = R_i^{\sum_{j=1}^{n_i} X_{ij}} (1 - R_i)^{1 - \sum_{j=1}^{n_i} X_{ij}} \quad (2.3)$$

To maximize (2.3), we will take the natural algorithm of likelihood function

:

$$\log(L(R_i, X_{i1}, \dots, X_{in_i})) = \sum_{j=1}^{n_i} X_{ij} \log(R_i) + (n_i - \sum_{j=1}^{n_i} X_{ij}) \log(1 - R_i) \quad (2.4)$$

Then the partial derivative of the natural logarithm with respect to R_I will

be set to zero:

$$\frac{\partial \log (L(R_i, X_{i1}, \dots, X_{in_i}))}{\partial R_i} = \frac{\sum_{j=1}^{n_i} X_{ij}}{R_i} - \frac{(n_i - \sum_{j=1}^{n_i} X_{ij})}{1 - R_i} \quad (2.5)$$

$$= \sum_{j=1}^{n_i} X_{ij}(1 - R_i) - (n_i - \sum_{j=1}^{n_i} X_{ij})R_i$$

$$= \sum_{j=1}^{n_i} X_{ij} - n_i R_i = 0 \quad (2.6)$$

$$R_i = \frac{\sum_{j=1}^{n_i} X_{ij}}{n_i}$$

Hence, the MLE estimator of each partition R_i is given by :

$$\hat{R}_i = \frac{\sum_{j=1}^{n_i} X_{ij}}{n_i} \quad (2.7)$$

, where n_i is the number of test cases that are allocated to partition i .

After we have determined the estimated reliability of each partition with n_i test cases, we aim now to find the estimation of overall software reliability.

Based on the definition, our estimator \hat{R} for overall software reliability , R , can be obtained by :

$$\hat{R} = \sum_{i=1}^k p_i R_i \quad (2.8)$$

Several strategies are used to allocate test cases among partitions and find the most accurate estimator of software reliability, such as [21, 25, 24, 19]. Our goal is to allocate the total number of test cases, N , among the partitions, K , using a strategy that achieves the most accurate estimate possible for software reliability. We are using a classical approach by aiming to minimize the variance of overall estimated software reliability incurred by the maximum likelihood estimator. To find the variance of the overall estimated reliability of \hat{R} , we will use two properties. First, operational profile is fixed, so we will prove that $Var(P_i R_i) = P_i^2 Var(R_i)$. Also, we assume that we test each partition independently of the others; thus, we need to prove that the sum of the variance equals the variance of the sum.

Theorem 2.2.1 *If R_i is any random variable and p_i is any constant, then*

$$Var(p_i R_i) = p_i^2 Var(R_i)$$

Proof 2.2.1

$$Var(p_i R_i) = E[(p_i R_i - E[p_i R_i])^2]$$

By the linearity of the expected value, we obtain

$$\begin{aligned}
\text{Var}(p_i R_i) &= \text{E} [(p_i R_i - p_i \text{E}[R_i])^2] \\
&= \text{E} [p_i^2 (R_i - \text{E}[R_i])^2] \\
&= p_i^2 \text{E} [(R_i - \text{E}[R_i])^2] \\
&= p_i^2 \text{Var}(R_i)
\end{aligned} \tag{2.9}$$

Now it is useful to prove that the sum of the variance is equal to the variance of the sum for independent random variables.

Theorem 2.2.2 *if R_i and R_ℓ are two independent random variables, then*

$$\text{Var}(R_i + R_\ell) = \text{Var}(R_i) + \text{Var}(R_\ell)$$

Proof 2.2.2

$$\begin{aligned}
\text{Var}(R_i + R_\ell) &= \text{E} [(R_i + R_\ell)^2] - (\text{E}[R_i + R_\ell])^2 \\
&= \text{E} [R_i^2 + R_\ell^2 + 2R_i R_\ell] - (\text{E}[R_i] + \text{E}[R_\ell])^2 \\
&= \text{E} [R_i^2] + 2\text{E}[R_i R_\ell] + \text{E} [R_\ell^2] - [\text{E}[R_i]^2 + 2\text{E}[R_i]\text{E}[R_\ell] + \text{E}[R_\ell]^2] \\
&= \text{E} [R_i^2] - \text{E}[R_i]^2 + \text{E} [R_\ell^2] - \text{E}[R_\ell]^2 + 2\text{E}[R_i R_\ell] - 2\text{E}[R_i]\text{E}[R_\ell] \\
&= \text{Var}(R_i) + \text{Var}(R_\ell) + 2\text{E}[R_i R_\ell] - 2\text{E}[R_i]\text{E}[R_\ell]
\end{aligned} \tag{2.10}$$

Since R_i and R_ℓ are independent, then $E[R_i R_\ell] = E[R_i] E[R_\ell]$

$$\begin{aligned} \text{Var}(R_i + R_\ell) &= \text{Var}(R_i) + \text{Var}(R_\ell) + 2E[R_i R_\ell] - 2E[R_i]E[R_\ell] \\ &= \text{Var}(R_i) + \text{Var}(R_\ell) \end{aligned} \tag{2.11}$$

We can use the properties in (2.9) and (2.10) to simplify the variance of overall software reliability estimator such that:

$$\begin{aligned} \text{Var}(\hat{\mathbf{R}}) &= \text{Var}\left(\sum_{i=1}^k p_i \hat{R}_i\right) \\ &= \sum_{i=1}^k p_i^2 \text{Var}(\hat{R}_i) \\ &= \sum_{i=1}^k \frac{p_i^2 R_i(1 - R_i)}{n_i} \end{aligned} \tag{2.12}$$

Because the aim to distribute the total number of test cases N among partitions is to obtain the most accurate estimator, we seek the best strategy to select n_1, n_2, \dots, n_k in (2.12) where the variance become as small as possible.

There are several potential choices to select the sample size for each partition i . In a sequential sampling scheme, test cases are determined during the testing process. In contrast, a balanced sampling scheme determines how to allocate test cases among partitions before testing starts. Our primary mission is to derive a near-optimal strategy that minimizes the variance of our estimated reliability model.

2.3 Optimal Sampling Scheme

Exhaustive testing for any non-trivial system is impossible; therefore, finding an exact value for software reliability is also not possible. To circumvent the problem, we have used sampling methods to estimate software reliability. In this section, we will consider the optimal allocation of the total number of test cases N to different partitions. Using optimal allocation, we can work to obtain the most accurate reliability estimator where each partition has the ideal number of test cases. We use Lagrange multipliers as a method to solve optimization problems with one or more constraints [27]. In our method, we want to minimize:

$$\text{Var}(\hat{R}) = \sum_{i=1}^k \frac{p_i^2 R_i(1 - R_i)}{n_i} \quad (2.13)$$

We have only one constrain that is the total number of test cases is fixed:

$$n_1 + n_2 + \dots + n_k = N$$

Theorem 2.3.1 *The optimal allocation of the sample size N to minimize the variance of overall estimated reliability subject to the constraint $N = \sum_{i=1}^k n_i$ is given by:*

$$n_i = \frac{N p_i \sqrt{R_i(1 - R_i)}}{\sum_{i=1}^k p_i \sqrt{R_i(1 - R_i)}}$$

Proof 2.3.1 *To use Lagrange multiplier, we want to illustrate two parts:*

1- the function that we need to minimize that is

$$f(n_1, n_2, \dots, n_k, \lambda) = \sum_{i=1}^k \frac{p_i^2 R_i (1 - R_i)}{n_i} \quad (2.14)$$

2- the constrain that we have

$$g(n_1, n_2, \dots, n_k) = \sum_{i=1}^k n_i - N \quad (2.15)$$

So we can now use lgrange function which is:

$$\begin{aligned} L(n_1, n_2, \dots, n_k, \lambda) &= f(n_1, n_2, \dots, n_k, \lambda) + \lambda g(n_1, n_2, \dots, n_k) \\ &= \sum_{i=1}^k \frac{p_i^2 R_i (1 - R_i)}{n_i} + \lambda \left(\sum_{i=1}^k n_i - N \right) \end{aligned} \quad (2.16)$$

Differentiating (2.16) with respect to n_i , and setting to Zero:

$$\frac{\partial L(n_1, n_2, \dots, n_k, \lambda)}{\partial n_i} = \frac{-p_i^2 R_i (1 - R_i)}{n_i^2} + \lambda = 0 \quad (2.17)$$

Thus,

$$\begin{aligned} n_i &= \sqrt{\frac{p_i^2 R_i (1 - R_i)}{\lambda}} \\ &= \frac{p_i \sqrt{R_i (1 - R_i)}}{\sqrt{\lambda}} \end{aligned} \quad (2.18)$$

Hence,

$$N = \sum_{i=1}^k n_i = \frac{\sum_{i=1}^k p_i^2 \sqrt{R_i(1-R_i)}}{\sqrt{\lambda}} \quad (2.19)$$

and

$$\lambda^{\frac{1}{2}} = \frac{1}{N} \sum_{i=1}^k p_i^2 \sqrt{R_i(1-R_i)} \quad (2.20)$$

Hence, we substitute (2.20) into the expression of n_i in (2.18), we obtain

$$n_i = \frac{N p_i^2 \sqrt{R_i(1-R_i)}}{\sum_{i=1}^k p_i^2 \sqrt{R_i(1-R_i)}} \quad (2.21)$$

From the previous theorem 2.3.1, we determined the optimal allocation for each partition given the total number of test cases, the operational profile, and software reliability for each partition. Because the software reliability for each partition R_i is unknown, the optimal sampling scheme is theoretical but not practical. Unfortunately, this limitation prevents the use of the optimal allocation strategy to obtain the best software reliability estimator. Due to the shortcoming, we were motivated to explore a sequential allocation scheme, as described in the next section.

2.4 Sequential Sampling Scheme

We consider the problem of allocating the total number of test cases among partitions such that the variance is minimized. Because optimal allocation is unattainable, the aim is to find a nearly optimal variance of the software reliability estimator. In contrast to a balanced sampling scheme where the allocation of test cases is determined in advance, a sequential sampling scheme relies on distribution test cases among partitions dynamically during the testing process. Evidence supports that the fully sequential scheme performs more efficiently than balanced sample allocation because we can refine our allocation for test cases among partitions as information accrues about the system. The variance of overall software reliability estimator is given by:

$$\text{Var}(\hat{R}) = \sum_{i=1}^k \frac{p_i^2 R_i(1 - R_i)}{n_i} \quad (2.22)$$

Assume that we have only two partitions, so the equation (2.22) will be written such that:

$$\text{Var}(\hat{R}) = \frac{p_1^2 R_1(1 - R_1)}{n_1} + \frac{p_2^2 R_2(1 - R_2)}{n_2} \quad (2.23)$$

We want to simplify (2.23) to facilitate finding how we can allocate test cases among partitions.

Here we show the algebraic steps to simplify (2.23):

$$\begin{aligned}
Var(\hat{R}) &= \frac{p_1^2 R_1(1-R_1)}{n_1} + \frac{p_2^2 R_2(1-R_2)}{n_2} \\
&= \frac{n_2 p_1^2 R_1(1-R_1) + n_1 p_2^2 R_2(1-R_2)}{n_1 n_2} \\
&= \frac{N(n_2 p_1^2 R_1(1-R_1) + n_1 p_2^2 R_2(1-R_2))}{N n_1 n_2} \\
&= \frac{(n_1 + n_2)(n_2 p_1^2 R_1(1-R_1) + n_1 p_2^2 R_2(1-R_2))}{(n_1 + n_2) n_1 n_2} \\
&= \frac{(n_1 n_2 + n_2^2) p_1^2 R_1(1-R_1) + (n_1^2 + n_1 n_2) p_2^2 R_2(1-R_2)}{(n_1 + n_2) n_1 n_2} \\
&= \frac{n_1 n_2 p_1^2 R_1(1-R_1) + n_2^2 p_1^2 R_1(1-R_1) + n_1^2 p_2^2 R_2(1-R_2) + n_1 n_2 p_2^2 R_2(1-R_2)}{(n_1 + n_2) n_1 n_2} \\
&= \frac{n_1 n_2 p_1^2 R_1(1-R_1) + n_2^2 p_1^2 R_1(1-R_1) + n_1^2 p_2^2 R_2(1-R_2) + n_1 n_2 p_2^2 R_2(1-R_2)}{(n_1 + n_2) n_1 n_2} \\
&\quad + \frac{2p_1 p_2 R_1(1-R_1)R_2(1-R_2) - 2p_1 p_2 R_1(1-R_1)R_2(1-R_2)}{(n_1 + n_2) n_1 n_2} \\
&= \frac{n_1 n_2 p_1^2 R_1(1-R_1) + n_1 n_2 p_2^2 R_2(1-R_2) + 2n_1 n_2 p_1 p_2 R_1(1-R_1)R_2(1-R_2)}{(n_1 + n_2) n_1 n_2} \\
&\quad + \frac{n_2^2 p_1^2 R_1(1-R_1) + n_1^2 p_2^2 R_2(1-R_2) - 2n_1 n_2 p_1 p_2 R_1(1-R_1)R_2(1-R_2)}{(n_1 + n_2) n_1 n_2} \\
&= \frac{\left(p_1 \sqrt{R_1(1-R_1)} + p_2 \sqrt{R_2(1-R_2)}\right)^2}{(n_1 + n_2)} + \frac{\left(n_2 p_1 \sqrt{R_1(1-R_1)} - n_1 p_2 \sqrt{R_2(1-R_2)}\right)^2}{(n_1 + n_2) n_1 n_2} \\
&= \frac{\left(p_1 \sqrt{R_1(1-R_1)} + p_2 \sqrt{R_2(1-R_2)}\right)^2}{N} + \frac{\left(n_2 p_1 \sqrt{R_1(1-R_1)} - n_1 p_2 \sqrt{R_2(1-R_2)}\right)^2}{N n_1 n_2}
\end{aligned}$$

Hence the variance of overall software reliability estimator can be rewritten such that:

$$Var(\hat{R}) = \frac{\left(P_1\sqrt{R_1(1-R_1)} + P_2\sqrt{R_2(1-R_2)}\right)^2}{N} + \frac{\left(n_2P_1\sqrt{R_1(1-R_1)} - n_1P_2\sqrt{R_2(1-R_2)}\right)^2}{Nn_1n_2} \quad (2.24)$$

By selection of n_1, n_2, \dots, n_k , we aim to minimize $Var(\hat{R})$ in above equation. Note the first term of (2.24) relies on the fixed total number of test cases, the known operational profiles, and unknown software reliability estimators. The first term of the equation is fixed, and we cannot manipulate it.

Hence, the variance will be bounded below by:

$$Var(\hat{R}) \geq \frac{\left(P_1\sqrt{R_1(1-R_1)} + P_2\sqrt{R_2(1-R_2)}\right)^2}{N} \quad (2.25)$$

with equality if:

$$\frac{n_1}{n_2} = \frac{P_1\sqrt{R_1(1-R_1)}}{P_2\sqrt{R_2(1-R_2)}} \quad (2.26)$$

By achieving the equality of (2.25), we obtain the variance incurred by the optimal allocation that is:

$$Var(\hat{R}_{optimal}) = \frac{\left(P_1\sqrt{R_1(1-R_1)} + P_2\sqrt{R_2(1-R_2)}\right)^2}{N} \quad (2.27)$$

The question becomes how to allocate the test cases among partitions to achieve the equality of (3.5). Rather than selecting n_1, n_2, \dots, n_k in advance by using balanced allocation that prevents recovery from test case distribution mistakes, we allocated test cases among partitions to force the second term of (2.24) to be zero. Thus, we dynamically determined partitions to which test cases can be allocated, as shown in the next section.

2.5 Fully Sequential Sampling Scheme

In this section, we introduce one of the sequential sampling schemes to test software to estimate software reliability. This sequential allocation promotes learning about the software such that test cases are allocated throughout the test process based on prior results. In a fully sequential sampling scheme, we allocated test cases one after another among partitions in each stage, such that an allocation decision is made after obtaining each test result. The goal in this scheme is to obtain the near-optimal choices for distributing the test cases among sub-domains that will minimize the variance of the overall software reliability estimator. This fully sequential scheme is expected to perform very well and more accurately than balanced sampling scheme in which the number of test cases is determined in advance for each partition. The following will be an outline for the fully sequential sampling scheme:

Step I :

We will distribute ℓ test cases among partitions.

Step II :

After ℓ test cases have been distributed where $\ell \geq k$, we will allocate test cases one by one.

Step IV :

The next test case $\ell + 1$ will be taken from partition i if for all partitions:

$$\frac{n_{\ell,1}}{n_{\ell,2}} < \frac{P_1 \sqrt{R_1(1-R_1)}}{P_2 \sqrt{R_2(1-R_2)}} \quad (2.28)$$

where $n_{\ell,1}$ is the cumulative test cases allocated to partition 1 after ℓ tests have been allocated and

$$\hat{R}_{\ell,1} = \sum_{j=1}^{n_{\ell,1}} \frac{X_{ij}}{n_{\ell,1}}$$

Step V :

If there is an equality, it can be resolved in an arbitrary fashion such as an independent random experiment.

Step VI :

we will repeat IV sequentially until all the test cases are allocated.

The idea of this approach is to evolve an optimal sampling method by dynamically altering allocation among partitions as we learn more about software performance. The advantage of this method is that we can gain insightful information for each sub-domain and refine test cases allocation sequentially among partitions using this information. However, the disadvantage of this method is that we have to make $\ell = N - K$ decision in advance to obtain our estimate. Also,

the execution of test cases may be time-consuming and costly run; therefore, the process should be automated. The primary benefit of automated testing is that we can execute more test cases in less time. As the software matures, more time is required to execute a single test case because, by this stage, many failures are eliminated. In the next sections, we make comparisons to prove the efficiency of a fully sequential sampling scheme over the balanced sampling scheme.

2.6 Comparison

Using theoretical results and Monte Carlo simulations, we compared the fully sequential sampling and balance sampling schemes. This comparison is made in consideration of minimizing the variance of the overall software reliability estimate. We demonstrate the efficiency of a fully sequential scheme when the total number of test cases N is large. Also, we expected that the proposed sequential scheme performs better than a balanced sampling scheme. This expectation of the fully sequential scheme is built on the refinements of allocation that are possible as we learn more about the software. Thus, we have an opportunity to recover from the mistakes in choosing the allocations of test cases among partitions by using previous test results. The distribution of test cases has a significant impact on the accuracy of the software reliability estimator. As we mentioned in equation (2.24), the first term is fixed and does not rely on the sample size of partition n_i . By choosing n_1, n_2, \dots, n_k that forces the second term to be zero to achieve equality of (3.5), the variance of a fully sequential scheme becomes:

$$Var(\hat{R}) \geq \frac{(P_1\sqrt{R_1(1-R_1)} + P_2\sqrt{R_2(1-R_2)})^2}{N} \quad (2.29)$$

In the next section, we compare theoretically sequential test allocation scheme with the optimal sampling scheme and prove its accuracy by showing an impact on the variance of the overall software reliability estimator. We also demonstrate results by Monte Carlo Simulations in which we compare a fully sequential sampling scheme with a balanced sampling scheme.

2.7 Theoretical Result

In this section, we present theoretical results to prove the efficiency of a fully sequential sampling scheme in terms of minimizing the variance of the overall software reliability estimator. It is critical to know about the advantages of choosing one specific methodology over another. The importance of this theoretical result lies in its support of a scheme that mimics the optimal sampling, especially when N is large. We aim to determine $Var(\hat{R}_{fully}) - Var(\hat{R}_{optimal})$, where $Var(\hat{R}_{fully})$ is the variance incurred by the fully sequential sampling scheme and $Var(\hat{R}_{optimal})$ is the variance associated with the optimal sampling scheme as in (2.27).

Theorem 2.7.1 *The excess variance incurred by the fully sequential sampling scheme over the variance incurred by the optimal sampling is of order of $\frac{1}{N}$.*

Proof 2.7.1 :

We want to show that

$$N \left[Var(\hat{R}_{fully}) - Var(\hat{R}_{optimal}) \right] = 0 \quad as \quad N \rightarrow \infty$$

, where

$$\text{Var}(\hat{R}_{fully}) = \frac{\left(\sum_{i=1}^k P_i \sqrt{R_1(i - R_i)}\right)^2}{N} + \sum_{i=1}^k \sum_{j=i+1}^k \frac{\left(n_j P_i \sqrt{R_i(1 - R_i)} - n_i P_j \sqrt{R_i(1 - R_i)}\right)^2}{N n_1 n_2}$$

and

$$\text{Var}(\hat{R}_{optimal}) = \frac{\left(\sum_{i=1}^k P_i \sqrt{R_1(i - R_i)}\right)^2}{N}$$

Thus,

$$\left[\text{Var}(\hat{R}_{fully}) - \text{Var}(\hat{R}_{optimal}) \right] = \sum_{i=1}^k \sum_{j=i+1}^k \frac{\left(n_j P_i \sqrt{R_i(1 - R_i)} - n_i P_j \sqrt{R_i(1 - R_i)}\right)^2}{N n_1 n_2}$$

We can establish the proof of theorem if we can show that in the fully sequential sampling scheme

$$\sum_{i=1}^k \sum_{j=i+1}^k \frac{\left(n_j P_i \sqrt{R_i(1 - R_i)} - n_i P_j \sqrt{R_j(1 - R_j)}\right)^2}{N n_1 n_2} \rightarrow 0 \quad \text{as } N \rightarrow \infty$$

Hence, the proof follows if we can show:

$$\frac{n_i}{n_j} = \hat{C}_{i,j} \quad \text{as } N \rightarrow \infty$$

, where

$$\hat{\mathbb{C}}_{i,j} = \frac{P_i \sqrt{R_i(1-R_i)}}{P_j \sqrt{R_j(1-R_j)}}$$

The rest of this proof is similar to Rekab [20]. Let k be the large enough and define $k^{(i)}$ to be the largest stage before k such that:

$$k^{(i)} = \sup \left\{ \ell < k : \frac{n_{i,\ell}}{n_{j,\ell}} = \mathbb{C}_{i,j}(\ell) \quad \text{for all } i \neq j \right\}$$

, where $n_{\ell,i}$ is the cumulative test cases allocated to partition i after ℓ tests have been allocated.

and $k^{(i)} \rightarrow \infty$ as $k \rightarrow \infty$. Then

$$\frac{n_{i,k}}{n_{j,k}} \leq \frac{n_{i,k^{(i)}} + 1}{n_{j,k^{(i)}}} \leq \hat{\mathbb{C}}_{i,j}(k^{(i)}) + \frac{1}{n_{i,k^{(i)}}}$$

On the other hand,

$$\frac{n_{i,k}}{n_{j,k}} \geq \frac{n_{i,k^{(i)}}}{n_{j,k^{(i)}} + 1} \geq \hat{\mathbb{C}}_{i,j}(k^{(i)}) \left(1 - \frac{1}{n_{j,k^{(i)}} + 1} \right)$$

The proof follows by the strong law of large numbers.

This theorem proves that the variance of fully sequential sampling schemes approach the optimal variance, as N gets large.

2.8 Monte Carlo Simulations

In this section, we make experimental comparisons among the fully sequential sampling scheme, the balanced sampling scheme, and the optimal sampling model.

In this simulation, we used R program. We discuss the case where the input domain D of all possible test cases is broken up into two disjoint partitions D_1, D_2 . Each of these sub-domains is associated with reliability R_1 and R_2 . Also, we assume that the known usage probability p_1, p_2 is associated with D_1, D_2 , respectively.

The tables show the results for the distinct software reliabilities R_1 and R_2 as well as various usage probabilities p_1, p_2 . The results show that the fully sequential sampling scheme performs more effectively than the balanced sampling scheme.

Table (1) presents results of the estimated variance under various parameters R_1, R_2 and different usage probabilities p_1, p_2 . In columns five, six, and seven, we show the estimated variance of the optimal allocation model, fully sequential allocation, and the balanced allocation model, respectively. The results show that the proposed sequential sampling scheme performs very close to the optimal sampling scheme. Also, the estimated variance of a fully sequential scheme is less than the estimated variance of the balanced scheme in most cases. The only case in which the balanced allocation performed better than fully sequential allocation scheme occurs when the usage probabilities p_1, p_2 are equal, and the reliability of partitions be $R_1 = R_2$ or $R_1 = 1 - R_2$ such as the case in the last two columns. In general, the fully sequential scheme has proved its superiority over the balanced plan through Monte Carlo Simulation.

In table (2), we focus on the comparison between the fully sequential sampling scheme and the balanced scheme. The last column contains the ratio of the proposed scheme over the balanced scheme. Where the ratio of two schemes is less than 1, the results favor the fully sequential sampling scheme. Regardless of the reliability and usage probability of each partition, the simulation results show

the superiority of a fully sequential sampling scheme over the balanced sampling scheme.

Table (6) presents the simulation results for various total sample sizes with the the ratio of the variance of optimal sampling scheme over the variance of sequential scheme. It is obvious from the simulation results that the sequential sampling scheme outperformed near optimally, as N becomes very large. Intuitively, and by Theorem (2.7.1), this result is what we would expect from that the sequential sampling schemes use gained during the testing process.

The two figures show results for various total sample sizes, N , and the ratio of optimal sampling scheme over the sequential sampling scheme. It is obvious that as N becomes large, the ratio be very close to 1. This indicates that the sequential sampling schemes become near optimally as the total number of test cases gets large.

CHAPTER 3

A TWO-STAGE SAMPLING SCHEME FOR SOFTWARE RELIABILITY ESTIMATION

3.1 Introduction

In this chapter, we consider the same problem as in the previous chapter, that of determining how test cases might be allocated among partitions. The purpose of the allocation strategy is to obtain the most accurate software reliability by minimizing the variance of overall estimated software reliability. In the previous chapter, we described the fully sequential sampling scheme in which test cases should be executed one after another; thus, allocation decisions could be made after each test execution. However, in some cases, a fully sequential sampling scheme is challenging and becomes very costly and time-consuming. Thus, we proposed another sequential sampling scheme that is more convenient in some situations.

In this chapter, we present a two-stage sampling scheme. The proposed scheme could be more accessible to implement than most fully sequential sampling schemes. This two-stage sampling scheme is executed in two batches, and therefore, requires fewer computations. Using our method, we determined the size of the second stage according to the results from the first stage. We compared the results from the two-stage sampling scheme with the balanced sampling scheme to show the strength of the two-stage method as measured by the variance of overall estimated reliability. These comparisons are made both theoretically and through Monte Carlo

simulation.

3.2 Sequential Sampling Scheme

As mentioned in chapter one, the optimal sampling scheme can only be theoretical and not practical because the software reliability for each partition R_i is unknown. In a balanced sampling scheme, we determine how to allocate test cases among partitions in advance, so we cannot refine our allocation through the testing process. Because of the shortcomings of the optimal and balanced sampling schemes, we were motivated to introduce a dynamic allocation approach.

In a two-stage sampling scheme, we have some assumptions that are:

- Total number of test cases is fixed.

$$N = n_1 + n_2 + \dots + n_k$$

- The parameters of the operational profile p_1, p_2, \dots, p_k are known.
- Each partition will be tested independently of the other partitions.

Our goal is to allocate the N test cases among k partitions to obtain the most accurate software reliability. This process shown here was determined by minimizing the variance of estimated software reliability.

From the above equation(3.3), the variance will be bounded below by:

$$\text{Var}(\hat{R}) = \sum_{i=1}^k \frac{p_i^2 R_i(1 - R_i)}{n_i} \quad (3.1)$$

Assume that we have only two partitions, then our goal is to minimize:

$$Var(\hat{R}) = \frac{p_1^2 R_1(1 - R_1)}{n_1} + \frac{p_2^2 R_2(1 - R_2)}{n_2} \quad (3.2)$$

The equation(3.2) can be re-written as:

$$Var(\hat{R}) = \frac{\left(P_1\sqrt{R_1(1 - R_1)} + P_2\sqrt{R_2(1 - R_2)}\right)^2}{N} + \frac{\left(n_2P_1\sqrt{R_1(1 - R_1)} - n_1P_2\sqrt{R_2(1 - R_2)}\right)^2}{Nn_1n_2} \quad (3.3)$$

From the above equation(3.3), the variance will be bounded below by:

$$Var(\hat{R}) \geq \frac{(P_1\sqrt{R_1(1 - R_1)} + P_2\sqrt{R_2(1 - R_2)})^2}{N} \quad (3.4)$$

with equality if:

$$\frac{n_1}{n_2} = \frac{P_1\sqrt{R_1(1 - R_1)}}{P_2\sqrt{R_2(1 - R_2)}} \quad (3.5)$$

The optimal variance is achieved when we can meet the equality of (3.5). Hence, we aim to determine the sample size for both n_1, n_2 that minimizes the variance of the software reliability estimator. In the next section, we dynamically allocate test cases among partitions.

3.3 Two Stage Sampling Scheme

In this section, we introduce a sequential sampling scheme to test software and estimate reliability. We present a two-stage sampling scheme in which test cases can be allocated among partitions in two phases. Our goal of this scheme is to obtain the near-optimal choices for distributing the test cases among sub-domains by minimizing the variance of the overall software reliability estimator. This two-stage sampling scheme is expected to be more convenient than a fully sequential sampling scheme because it requires fewer computations than the fully sequential sampling scheme. Also, the two-stage sampling scheme is expected to perform better than a balanced sampling scheme by virtue of lower the variance incurred by the overall estimated software reliability. The following description is an outline for a two-stage sampling scheme:

3.3.1 First Stage

In the first stage, we sample initial test cases from each partition. Many researchers have shown that an expected initial sample size should be about \sqrt{N} [29, 2]. The initial sample size L must satisfy with the following conditions:

- if N is large, L must be large.
- $L \leq \frac{N}{k}$, where k is the number of partitions.
- $\frac{L}{N} \rightarrow 0$ as $N \rightarrow \infty$, which means that L is relatively small compared to N .

After we allocate test cases for each partition, we can estimate the software reliability with the obtained results of the small sample size by:

$$\hat{R}_i = \frac{\sum_{j=1}^{\sqrt{N}} X_{ij}}{\sqrt{N}}$$

Also, the overall software reliability estimator based on the initial sample is given by:

$$\begin{aligned} \hat{R}_i &= \sum_{i=1}^k p_i \hat{R}_i \\ &= \sum_{i=1}^k p_i \left[\frac{\sum_{j=1}^{\sqrt{N}} X_{ij}}{\sqrt{N}} \right] \end{aligned} \tag{3.6}$$

After the outcome of the test cases is obtained for each partition, we can estimate R_i based on the initial sample. The first stage results have a significant effect on a decision on how to allocate test cases in the second stage.

3.3.2 Second Stage

In the second stage, we would use the information that we gained from first stage to sequentially allocate the remaining test cases among partitions such that the variance of overall software reliability is minimized. Because we assumed that

we have only two sub-domains, we aim to determine the number of test cases for both n_1 and n_2 . Mathematically, the total remaining of test cases would be such that:

$$N - 2\sqrt{N} = \begin{cases} n_1 - \sqrt{N} & \text{,for first partition.} \\ n_2 - \sqrt{N} & \text{,for second partition.} \end{cases} \quad (3.7)$$

We will combine the first stage results with the optimal theoretical results to determine the test cases for each partition. The number of test cases for each partition n_1 and n_2 is determined as:

$$n_1 = N \frac{P_1 \sqrt{\hat{R}_1(1 - \hat{R}_1)}}{P_1 \sqrt{\hat{R}_1(1 - \hat{R}_1)} + P_1 \sqrt{\hat{R}_1(1 - \hat{R}_1)}}$$

$$n_2 = N - n_1$$

After we have performed the second stage, we can obtain the variance incurred by two-stage sampling scheme that is given such that:

$$Var(\hat{R}) = \frac{p_1^2 R_1(1 - R_1)}{n_{1,twostage}} + \frac{p_2^2 R_2(1 - R_2)}{n_{2,twostage}} \quad (3.8)$$

In next two section, we will show theoretical result and Monte Carlo simulation to prove that the two stage sampling scheme performs much better than the balanced

sampling scheme.

3.4 Theoretical Result

In this section, we will prove the efficiency of two-stage sampling scheme in terms of minimizing the variance of overall software reliability estimator via theoretical result. We aim to determine $Var(\hat{R}_{two\ stage}) - Var(\hat{R}_{optimal})$, where $Var(\hat{R}_{two\ stage})$ is the variance incurred by the fully sequential sampling scheme and $Var(\hat{R}_{optimal})$ is the variance incurred by the optimal sampling scheme. The next theorem proves that the variance of two-stage sampling schemes approaches the optimal variance, as N gets large.

Theorem 3.4.1 *The excess variance incurred by the two stage sampling scheme over the variance incurred by the optimal sampling is of order of $\frac{1}{N}$.*

Proof 3.4.1 :

It will in fact show that

As N gets large, $N \left[Var(\hat{R}_{fully}) - Var(\hat{R}_{optimal}) \right]$ will approach zero.

The variance incurred by the optimal is given by:

$$Var(\hat{R}_{optimal}) = \frac{\left(\sum_{i=1}^k P_i \sqrt{R_1(i - R_i)} \right)^2}{N}$$

Hence, the proof will follow if we show that the test cases of partition i is nearly equal to

$$\frac{P_i \sqrt{R_i(1 - R_i)}}{\sum_{i=1}^k P_i \sqrt{R_i(1 - R_i)}}$$

We know that the estimation of n_i should not exceed the remaining of test cases for second stage and should not be smaller than the initial sample, that is:

$$n_i = \min \{N - (K - 1) L, \max \{L, \hat{n}_i\}\}$$

, where $i = 1, \dots, k - 1$ and L is the initial sample.

Then

$$\frac{n_i}{N} = \min \left\{ \frac{N}{N} - \frac{(K - 1) L}{N}, \max \left\{ \frac{L}{N}, \frac{\hat{n}_i}{N} \right\} \right\}$$

By substituting the estimation of n_i

$$\frac{n_i}{N} = \min \left\{ \frac{N}{N} - \frac{(K - 1) L}{N}, \max \left\{ \frac{L}{N}, \frac{N P_i \sqrt{\hat{R}_i(1 - R_i)}}{N \sum_{i=1}^k P_i \sqrt{R_1(i - R_i)}} \right\} \right\}$$

Hence, for a large number of test cases to be tested N , the proportion of $\frac{n_i}{N}$

approach like $\frac{P_i \sqrt{R_i(1 - R_i)}}{\sum_{i=1}^k P_i \sqrt{R_i(1 - R_i)}}$.

This theorem proves that the variance of two stage sampling schemes approaches

the optimal variance, as N gets large.

3.5 Monte Carlo Simulations

In this section, we show the results of experimental comparison between the two stage sampling scheme described above, the balanced sampling schemes and the optimal sampling scheme. The goal of this comparison is to figure out the scheme that give most accurate estimator in terms of minimizing the variance of overall estimated software reliability. We consider the case where the input domain of the program D is divided into two sub-domains D_1, D_2 . Each of these sub-domains is associated with reliability R_1 and R_2 . Also, associated with each sub-domain D_1, D_2 is the usage probability p_1, p_2 , respectively.

Table (3) displays results of the estimated variance of the optimal sampling scheme, two-stage sampling scheme, and the balanced sampling scheme in columns five, six, and seven, respectively. We show different values of software reliability as the usage probabilities across the first four columns. The results indicated that the proposed sequential sampling scheme performed near optimally. Also, the estimated variance of the two-stage sampling scheme is less than the estimated variance of the balanced scheme in most cases. A result similar to the case in which the balanced allocation performed better than the fully sequential sampling scheme, was also found with the use of the two-stage sampling scheme. The case occurs when the usage probabilities p_1, p_2 are equal, and the reliability of partitions be $R_1 = R_2$ or $R_1 = 1 - R_2$, as in the last two columns. However, the two-stage sampling scheme demonstrates superior outcomes over those from a balanced scheme in all other cases.

Table (4) presents the comparison between the fully sequential sampling scheme and the two-stage scheme. The last column contains the ratio of the fully scheme over the two scheme. Where the ratio of two schemes is less than 1, the results favor the fully sequential sampling scheme. because an allocation decision is made after obtaining each test result. However, this two-stage sampling scheme is executed in two batches, and therefore, requires fewer computations than fully sequential sampling scheme, and achieves more accuracy than the balanced sampling scheme, as we see in table (5).

In table (5), we focus on the comparison between the two-stage sampling scheme and the balanced scheme. The last column contains the ratio of the two-stage sampling scheme over the balanced scheme. If the ratio of two schemes is less than 1, the results favor the two-stage sampling scheme as more effective than the balanced scheme. In most cases, the results showed that the two-stage sampling scheme performs much better than the balanced sampling scheme regardless of the reliability and usage probability of each partition. Intuitively, this is what we expect from sequential sampling schemes over the traditional approach because these schemes dynamically use the information for decision-making as the information is obtained during the testing process.

CHAPTER 4

SUMMARY AND CONCLUSION

Software testing is an integral and valuable component of the product development cycle. Developers and those who are responsible for this task give more attention to the reliability of a system before reliability testing begins. The results of reliability estimation have a significant impact on decision-makers as they consider whether to release a product. In traditional approaches such as the balanced sampling scheme, the allocating of test cases among partitions is performed before reliability testing began. Therefore, it can be impossible to recover from mistakes in selecting the allocations of test cases among partitions as the testing process continues. Thus, the lack of use of errors to adjust the estimate during testing is a shortcoming of the balanced sampling scheme.

In this study, we adopted a sequential sampling scheme in which we can refine the allocation of test cases sequentially based on the information gained as the testing proceeds. The purpose of this scheme is to minimize the variance of the reliability estimate by the allocation of test cases dynamically among partitions.

In chapter 1, we introduced some background and definitions concerning software reliability. Also, we mentioned some assumptions underlying the use of the schemes. One of these assumptions was that the total number of test cases is fixed, and another was that the domain of all possible test cases must be divided into non-overlapping partitions. We clarified the importance of operational profile to make the software testing more efficient.

After introducing some background and assumptions, we presented a fully sequential sampling scheme in chapter 2. The goal of this scheme is to estimate the reliability of a software system using partition testing. In this scheme, test cases are allocated one after another, allowing allocation decisions after each test is executed. We have proved both theoretically and through Monte Carlo simulation that the fully sequential sampling scheme always performs at least as well as the balanced sampling scheme in which test case allocation is determined in advance.

In the third chapter, the two-stage sampling scheme is examined. The two-stage sampling scheme was easier to implement than a fully sequential sampling scheme. Unlike the fully sequential sampling scheme, the two-stage sampling scheme is executed in two batches of allocated test cases instead of one. The proposed sampling scheme requires fewer computations, costs less, while preserving the trustworthy results. Using theoretical results and Monte Carlo simulation, we showed that the two-stage sampling scheme always performs at least as well as the balanced sampling scheme.

In our research, we assumed that operational profile is fixed. The very important point is that how about if we assume of operational profile is random variable. This assumption might be addressed in future studies. In the future work, we will also discuss multistage sampling scheme. In this method, We will distribute the N test cases in L stages. Thus, we will allocate group of test cases among the partitions in L stages. We expect that this method will be more appropriate than the fully sequential sampling when N is large. Also, future research may investigate combining between fully sequential sampling scheme and the multistage sampling scheme. We expect that this method will consume less time consuming

than the fully sequential sampling scheme but achieves more accuracy than the multistage sampling scheme.

APPENDIX A

TABLES

Table 1: Comparison of Optimal , Fully, and Balanced Allocation Model

with varying Reliability and operational profile

[N = 500, 500 replication]

R_1	R_2	P_1	P_2	$Var(op)$	$Var(fully)$	$Var(Bal)$
0.9	0.3	0.3	0.7	0.000337	0.000351	0.000444
0.8	0.6	0.3	0.7	0.0004286	0.000429	0.000528
0.5	0.5	0.3	0.7	0.0005	0.000507	0.000580
0.4	0.6	0.9	0.1	0.00048	0.00050	0.00078
0.1	0.8	0.9	0.1	0.000192	0.000194	0.000298
0.7	0.01	0.5	0.5	0.000155	0.000156	0.000219
0.5	0.5	0.5	0.5	0.00050	0.00053	0.00050
0.6	0.4	0.5	0.5	0.000480	0.000485	0.000480

Table 2: Comparison of Fully Allocation, and Balanced Allocation Model

with varying Reliability and operational profile

[N = 500, 500 replication]

R_1	R_2	P_1	P_2	$Var(fully)$	$Var(Bal)$	$\frac{Var(fully)}{Var(Bal)}$
0.9	0.3	0.3	0.7	0.000351	0.000444	0.79
0.8	0.6	0.3	0.7	0.000429	0.000528	0.81
0.5	0.5	0.3	0.7	0.000507	0.000580	0.87
0.4	0.6	0.9	0.1	0.00050	0.00078	0.64
0.1	0.8	0.9	0.1	0.000194	0.000298	0.65
0.7	0.01	0.5	0.5	0.000156	0.000219	0.71
0.5	0.5	0.5	0.5	0.00053	0.00050	1.06
0.6	0.4	0.5	0.5	0.000485	0.000480	1.01

Table 3: Comparison of Optimal , Two Stage, and Balanced Allocation Model

with varying Reliability and operational profile

[N = 500, 500 replication]

R_1	R_2	P_1	P_2	$Var(op)$	$Var(TwoStage)$	$Var(Bal)$
0.9	0.3	0.3	0.7	0.000337	0.0003712	0.000444
0.8	0.6	0.3	0.7	0.0004286	0.000470	0.000528
0.5	0.5	0.3	0.7	0.0005	0.000548	0.000580
0.4	0.6	0.9	0.1	0.00048	0.00052	0.00078
0.1	0.8	0.9	0.1	0.000192	0.000215	0.000298
0.7	0.01	0.5	0.5	0.000155	0.000172	0.000219
0.5	0.5	0.5	0.5	0.00050	0.000548	0.00050
0.6	0.4	0.5	0.5	0.000480	0.000520	0.000480

Table 4: Comparison of Fully Allocation, Two-Stage Allocation Model
with varying Reliability and operational profile
[N = 500, 500 replication]

R_1	R_2	P_1	P_2	$Var(fully)$	$Var(Twostage)$	$\frac{Var(fully)}{Var(Twostage)}$
0.9	0.3	0.3	0.7	0.000351	0.0003712	0.95
0.8	0.6	0.3	0.7	0.000429	0.000470	0.91
0.5	0.5	0.3	0.7	0.000507	0.000548	0.93
0.4	0.6	0.9	0.1	0.00050	0.00052	0.96
0.1	0.8	0.9	0.1	0.000194	0.000215	0.90
0.7	0.01	0.5	0.5	0.000156	0.000172	0.91
0.5	0.5	0.5	0.5	0.00053	0.000548	0.97
0.6	0.4	0.5	0.5	0.000485	0.000520	0.93

Table 5: Comparison of Two Stage Allocation, and Balanced Allocation Model

with varying Reliability and operational profile

[N = 500, 500 replication]

R_1	R_2	P_1	P_2	$Var(TwoStage)$	$Var(Bal)$	$\frac{Var(TwoStage)}{Var(Bal)}$
0.9	0.3	0.3	0.7	0.0003712	0.000444	0.83
0.8	0.6	0.3	0.7	0.000470	0.000528	0.89
0.5	0.5	0.3	0.7	0.000548	0.000580	0.94
0.4	0.6	0.9	0.1	0.00052	0.00078	0.66
0.1	0.8	0.9	0.1	0.000215	0.000298	0.72
0.7	0.01	0.5	0.5	0.000172	0.000219	0.78
0.5	0.5	0.5	0.5	0.000548	0.00050	1.09
0.6	0.4	0.5	0.5	0.000520	0.000480	1.08

Table 6: Comparison of Optimal , Fully, Two-Stage, and Balanced Model
with varying Reliability and operational profile

[N = 500, 500 replication]

R_1	R_2	P_1	P_2	$Var(op)$	$Var(fully)$	$Var(TwoStage)$	$Var(Bal)$
0.9	0.3	0.3	0.7	0.000337	0.000351	0.0003712	0.000444
0.8	0.6	0.3	0.7	0.0004286	0.000429	0.000470	0.000528
0.5	0.5	0.3	0.7	0.0005	0.000507	0.000548	0.000580
0.4	0.6	0.9	0.1	0.00048	0.00050	0.00052	0.00078
0.1	0.8	0.9	0.1	0.000192	0.000194	0.000215	0.000298
0.7	0.01	0.5	0.5	0.000155	0.000156	0.000172	0.000219
0.5	0.5	0.5	0.5	0.00050	0.00053	0.000548	0.00050
0.6	0.4	0.5	0.5	0.000480	0.000485	0.000520	0.000480

Table 7: Comparison of Highly Reliability
with varying Reliability and operational profile
[$N = 500$, 500 replication, $p_1 = 0.3$, $p_2 = 0.7$]

R_1	R_2	$Var(fully)$	$Var(Bal)$	$\frac{Var(fully)}{Var(Bal)}$
0.99	0.9	0.000130	0.000179	0.72
0.8	0.7	0.000338	0.000469	0.82
0.9	0.8	0.0002756	0.000346	0.79
0.6	0.7	0.000472	0.000498	0.89
0.9	0.7	0.000433	0.000444	0.77
0.99	0.8	0.000224	0.000317	0.70

Table 8: Comparison of Fully Allocation, and Balanced Allocation Model with the percentage of n_1

with varying Reliability and operational profile

[N = 500, 500 replication]

R_1	R_2	p_1	p_2	$Var(fully)$	$Var(Bal)$	$\frac{Var(fully)}{Var(Bal)}$	$n_{1,optimal}\%$	$n_{1,fully}\%$
0.9	0.3	0.3	0.7	0.000351	0.000444	0.79	21%	31%
0.8	0.6	0.3	0.7	0.000429	0.000528	0.81	25%	27%
0.7	0.4	0.3	0.7	0.000507	0.000580	0.87	28%	28%
0.7	0.7	0.4	0.6	0.000420	0.000436	0.96	40%	38%
0.5	0.5	0.5	0.5	0.00053	0.00050	1.06	50%	49%

* in all cases the $n_{1,balanced}\% = 50\%$ for balances allocation.

Table 9: Comparison of Two Stage Allocation, and Balanced Allocation Model with the percentage of n_1

with varying Reliability and operational profile

[N = 500, 500 replication]

R_1	R_2	p_1	p_2	$Var(TwoStage)$	$Var(Bal)$	$\frac{Var(TwoStage)}{Var(Bal)}$	$n_{1,optimal}\%$	$n_{1,Twostage}\%$
0.9	0.3	0.3	0.7	0.0003712	0.000444	0.79	21%	35%
0.8	0.6	0.3	0.7	0.000429	0.000528	0.81	25%	33%
0.7	0.4	0.3	0.7	0.000507	0.000580	0.87	28%	26%
0.7	0.7	0.4	0.6	0.000420	0.000436	0.96	40%	36%
0.5	0.5	0.5	0.5	0.00053	0.00050	1.06	50%	45%

* in all cases the $n_{1,balanced}\% = 50\%$ for balances allocation.

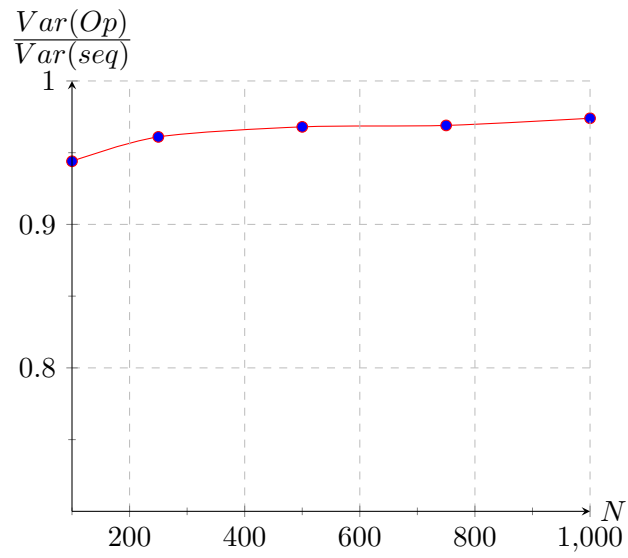
Table 10: Comparison of Optimal and Sequential Allocation Model

$$[R_1 = 0.7, R_2 = 0.1, p_1 = 0.9, p_2 = 0.1]$$

N	$\frac{Var(Op)}{Var(Seq)}$
100	0.944
250	0.961
500	0.968
750	0.969
1000	0.974

Comparison of Optimal Allocation, and Sequential Allocation Model with
varying of N

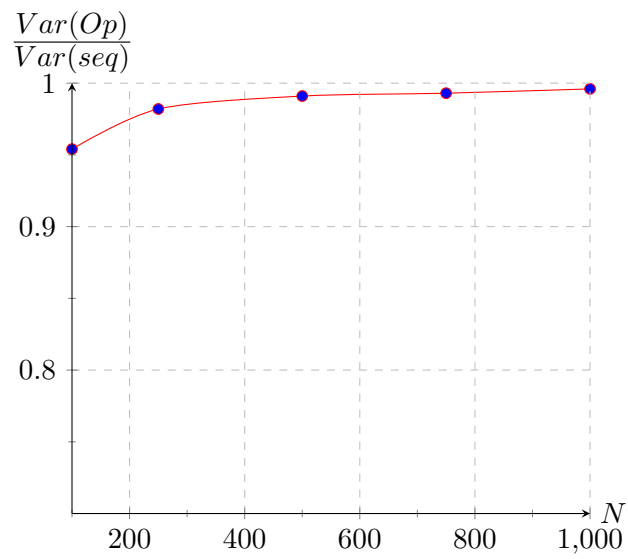
$$[R_1 = 0.7, R_2 = 0.1, p_1 = 0.9, p_2 = 0.1]$$



Comparison of Optimal Allocation, and Sequential Allocation Model

with varying of N

$$[R_1 = 0.3, R_2 = 0.6, p_1 = 0.9, p_2 = 0.1]$$



Appendix B

A Fully Sequential Sampling Scheme Program

```
simulate<-function(R1=R1,R2=R2,N=N,NRep=NRep,p1=p1,p2=p2){  
  q1=p1*sqrt(R1*(1-R1))  
  q2=p2*sqrt(R2*(1-R2))  
  n1opt= N*q1/(q1+q2)  
  VarOptimal=(q1+q2)^2/N  
  
  list_n1s=c()  
  X1Successes=c()  
  X2Successes=c()  
  for(rep in 1:NRep){  
    data=array(NA, dim=c(N,2))  
    for(i in 1:N) {  
      o1=rbinom(1, 1, R1);      data[1,1]=1;  
      data[1,2]=o1  
      o2=rbinom(1, 1, R2);      data[2,1]=2;  
      data[2,2]=o2  
  
      n1s=1  
      for(k in 3:N){  
        cid=NA  
        N2k=sum(data[,1]==2,na.rm=T)  
        N1k=sum(data[,1]==1,na.rm=T)
```

```

Xbar1k=sum(p1*data[data[,1]==1,2],na.rm=T)/N1k
Xbar2k=sum(p2*data[data[,1]==2,2],na.rm=T)/N2k
diff= N1k*p2*sqrt(Xbar2k*(1-Xbar2k))-N2k*p1*sqrt(Xbar1k*(1-Xbar1k))
  if(diff<0)cid=1
  else if(diff>0)cid=2
  else cid=sample(c(1,2),1)

p=NA
if(cid==1){
  p=R1
  n1s=n1s+1
}else if(cid==2){
  p=R2
}
o=rbinom(1,1,p); data[k,1]=cid; data[k,2]=o
}
}
X1_successCnt=sum(p1*data[data[,1]==1,2])
X2_successCnt=sum(p2*data[data[,1]==2,2])
m1s=sum(data[,1]==1)
list_m1s=c(list_n1s,n1s)
X1Successes=c(X1Successes,X1_successCnt)

```

```

X2Successes=c(X2Successes,X2_successCnt)

p1hat=sum(p1*X1Successes)/sum(list_n1s)
p2hat=sum(p2*X2Successes)/sum(N-list_n1s)
v1=p1^2*R1*(1-R1)/(n1s)
v2=p2^2*R2*(1-R2)/(N-n1s)

var_rhat=v1+v2

result <- c(0)
result[1] <- var_rhat
return(result)
}

}

```


A Two-Stage Sequential Sampling Scheme Program

```
simulate<-function (R1=R1 , R2=R2 , N=N , NRep=NRep , p1=p1 , p2=p2) {  
  q1=p1*sqrt (R1*(1-R1))  
  q2=p2*sqrt (R2*(1-R2))  
  n1opt= N*q1/(q1+q2)  
  VarOptimal=(q1+q2)^2/N  
  
  badness=0  
  df<-rbindlist (lapply (1:NRep, function (j) {  
    outcomes1<-c ()  
    outcomes2<-c ()  
    sqrtn=floor (sqrt (N))  
    for (i in 1:sqrtn) {  
  
      outcomes1<-c (outcomes1 , runif (1) < R1)  
      outcomes2<-c (outcomes2 , runif (1) < R2)  
    }  
  
    q1hat=p1*sqrt (R1*(1-R1))  
    q2hat=p2*sqrt (R2*(1-R2))  
  
    n12s=round ((N-2*sqrtn) * q1hat / (q1hat+q2hat))  
  
    n22s=N-2*sqrtn-n12s
```

```

stopifnot(!is.na(n12s))

outcomes1<-c(outcomes1,(runif(n12s)<R1))
outcomes2<-c(outcomes2,(runif(n22s)<R2))
stopifnot(length(outcomes1)+length(outcomes2)==N)

list(n12s=n12s,n22s=n22s,R1hat=mean(outcomes1,na.rm=T)
,R2hat=mean(outcomes2,na.rm=T))
)))

R12s<-mean(df$R1hat,na.rm=T)
R22s<-mean(df$R2hat,na.rm=T)
n12s<-mean(df$n12s,na.rm=T)
n22s<-mean(df$n2,na.rm=T)

VarTwoStage=p1^2*R12s*(1-R12s)/n12s+p2^2*R22s*(1-R22s)/n22s

result <- c(0)
result[1] <- VarTwoStage
return(result)
}

```

REFERENCE LIST

- [1] Adams, T. Total variance approach to software reliability estimation. *IEEE transactions on software engineering* 22, 9 (1996), 687–688.
- [2] Al-Maati, S. A., and Rekab, K. Dynamic test allocation model for software reliability. In *Third International Conference on Quality Software, 2003. Proceedings.* (2003), IEEE, pp. 26–31.
- [3] Chen, T., Sahinoglu, M., Von Mayrhauser, A., Hajjar, A., and Anderson, C. How much testing is enough? Applying stopping rules to behavioral model testing. In *Proceedings 4th IEEE International Symposium on High-Assurance Systems Engineering* (1999), IEEE, pp. 249–256.
- [4] Cochran, W. G. Sampling techniques-3.
- [5] DeMillo, R. A., Lipton, R. J., and Sayward, F. G. Hints on test data selection: Help for the practicing programmer. *Computer* 11, 4 (1978), 34–41.
- [6] Enaya, T., Rekab, L., and Tadj, L. A Batch sequential sampling scheme for estimating the reliability of a series/parallel system. *International Journal of Reliability and Applications* 11, 1 (2010), 17–22.
- [7] Frankl, P. G., and Weyuker, E. J. A formal analysis of the fault-detecting ability of testing methods. *IEEE Transactions on Software Engineering* 19, 3 (1993), 202–213.

- [8] Gutjahr, W. J. Optimal test distributions for software failure cost estimation. *IEEE Transactions on Software Engineering* 21, 3 (1995), 219–228.
- [9] Howden, W. E. Functional program testing. *IEEE Transactions on Software Engineering*, 2 (1980), 162–169.
- [10] Kaner, C., et al. The impossibility of complete testing. *Software QA* 4, 4 (1997), 28.
- [11] Littlewood, B., and Wright, D. Some conservative stopping rules for the operational testing of safety critical software. *IEEE Transactions on software Engineering* 23, 11 (1997), 673–683.
- [12] Musa, J. D. The operational profile in software reliability engineering: an overview. In *[1992] Proceedings Third International Symposium on Software Reliability Engineering* (1992), IEEE, pp. 140–154.
- [13] Musa, J. D. Operational profiles in software-reliability engineering. *IEEE software* 10, 2 (1993), 14–32.
- [14] Musa, J. D. The operational profile. In *Reliability and Maintenance of Complex Systems*. Springer, 1996, pp. 333–344.
- [15] Neyman, J. On the two different aspects of the representative method: the method of stratified sampling and the method of purposive selection. In *Breakthroughs in Statistics*. Springer, 1992, pp. 123–150.
- [16] Podgurski, A., and Yang, C. Partition testing, stratified sampling, and cluster analysis. *ACM SIGSOFT Software Engineering Notes* 18, 5 (1993), 169–181.

- [17] Poore, J. H., Mills, H. D., and Mutchler, D. Planning and certifying software system reliability. *IEEE software* 10, 1 (1993), 88–99.
- [18] Rapps, S., and Weyuker, E. J. Selecting software test data using data flow information. *IEEE transactions on software engineering*, 4 (1985), 367–375.
- [19] Rekab, K. Sequential allocation for estimation problem. *Stochastic Analysis and Applications* 10, 4 (1992), 465–470.
- [20] Rekab, K. A sampling scheme for estimating the reliability of a series system. *IEEE transactions on reliability* 42, 2 (1993), 287–290.
- [21] Rekab, K., and Cheng, Y. An accelerated sequential sampling for estimating the reliability of N-parallel systems. *International Journal of Reliability and Applications* 14, 2 (2013), 71–78.
- [22] Rekab, K., and Tahir, M. A two-stage sequential allocation scheme for estimating the product of several means. *Stochastic analysis and applications* 18, 2 (2000), 289–298.
- [23] Rekab, K., Thompson, H., and Wu, W. A fully sequential test allocation for software reliability estimation. *Journal of Applied Statistical Science* 20, 1 (2012), 63.
- [24] Rekab, K., Thompson, H., and Wu, W. An efficient test allocation for software reliability estimation. *Applied Mathematics and Computation* 220 (2013), 94–103.

- [25] Rekab, K., Thompson, H., and Wu, W. A multistage sequential test allocation for software reliability estimation. *IEEE Transactions on Reliability* 62, 2 (2013), 424–433.
- [26] Richardson, D. J., and Clarke, L. A. A partition analysis method to increase program reliability. In *Proceedings of the 5th international conference on Software engineering* (1981), IEEE Press, pp. 244–253.
- [27] Rockafellar, R. T. Lagrange multipliers and optimality. *SIAM review* 35, 2 (1993), 183–238.
- [28] Sayre, K., and Poore, J. H. Partition testing with usage models. In *Proceedings. Science and Engineering for Software Development: A Recognition of Harlin D. Mills Legacy (Cat. No. PR00010)* (1999), IEEE, pp. 24–30.
- [29] Stout, Q. Determining optimal few-stage allocation procedures.
- [30] Taylor-Sakyi, K. Reliability Testing Strategy-Reliability in Software Engineering. *arXiv preprint arXiv:1605.01097* (2016).
- [31] Walton, G. H., Poore, J. H., and Trammell, C. J. Statistical testing of software based on a usage model. *Software: Practice and Experience* 25, 1 (1995), 97–108.
- [32] Weyuker, E. J., and Jeng, B. Analyzing partition testing strategies. *IEEE Transactions on software Engineering*, 7 (1991), 703–711.
- [33] Weyuker, E. J., and Ostrand, T. J. Theories of program testing and the application of revealing subdomains. *IEEE Transactions on software engineering*, 3 (1980), 236–246.

- [34] Whittaker, J. A., and Thomason, M. G. A Markov chain model for statistical software testing. *IEEE Transactions on Software engineering* 20, 10 (1994), 812–824.

VITA

Bader Alanazi was born on March 3, 1987, in Rafha, Saudi Arabia. On the 15th of May 2010, he received a B.S. in Mathematics from Qassim University in Saudi Arabia.. He received a M.S. degree in Mathematics and Statistics from the University of Missouri-Kansas City in 2016.