2008

Sensitivity analysis of reliability for structure-based software via simulation

Jun Xu
West Virginia University

Sensitivity Analysis of Reliability for Structure-Based Software via Simulation

Jun Xu

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Industrial Engineering

Feng Yang, Ph.D., Chair
Wafik H. Iskander, Ph.D.
Majid Jaraiedi, Ph.D.

Department of Industrial and Management Systems Engineering
Morgantown, West Virginia
2008

Keywords: Software Reliability, Sensitivity analysis, Discrete-Event Simulation

ABSTRACT

Sensitivity Analysis of Reliability for Structure-Based Software via Simulation

Jun Xu

Computer simulation is an appealing approach for the reliability analysis of structure-based software systems as it can accommodate complexities present in realistic systems. When the system is complex, a screening experiment to quickly identify important factors (components) can significantly improve efficiency of the analysis. The challenge is to guarantee the correctness of the screening results with stochastic simulation responses. Control Sequential Bifurcation (CSB) is a new method for factors screening using simulation experiments, when only main effects models are considered. By grouping factors, CSB can identify the importance of factors while reducing the simulation effort. With appropriate hypothesis testing procedures embedded, CSB procedure can simultaneously control the Type I error probability and the power. The existing work has focused on normally distributed output responses. This thesis extends the existing CSB procedure by embedding Meeker's conditional sequential test to deal with binary responses and guarantee the desired error control for factor screening results. The effectiveness of the extended factor screening procedure is demonstrated with the application on a software system.

ACKNOWLEDGEMENTS

This research is supported by the NASA OSMA Software Assurance Research Program (SARP) managed through the NASA IV&V Facility, Fairmont, West Virginia.

I would like to thank my advisor, Dr. Feng Yang, for her great guidance, for giving me the opportunity to work on this project. That could not have been written without her encouragement and support.

I am also thankful to Dr. Wafik H. Iskander and Dr. Majid Jaraiedi for serving on my committee, and for their insightful ideas and assistance in preparing this thesis.

TABEL OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF NOMENCLATURE

α	Probability of type I error
γ	The power of the test
λ	Steady state failure rate
ζ_i	Nominal level of factor i
p	System success probability
K	Number of factors in the system
x_i	Factor i
$\boldsymbol{x}$	Vector of all factors
M_i	State in Markov chain
β_i	Unknown coefficient of factor i
$\boldsymbol{\beta}$	Vector of all factor unknown coefficients
Y	Random output of system: 1 (success), 0 (failure)
$P(\boldsymbol{x}, \beta)$	System success probability of factor settings x and coefficients β
c_i	The cost per unit change of factor i (i = 1, 2, …, K)
c^*	Maximum of c_i (i = 1, 2, …, K)
t	Odds ratio of ratios of two different systems
t_0	The minimum odds ratio value that people could be willing to spend c^* to obtain
t_1	The odds ratio value in that people would not want to miss if it could be achieved for only a cost of c^*
$\boldsymbol{x}(k)$	Factor level setting relative to factor k
$y_i(k)$	The i^{th} observation at factor level settings $\boldsymbol{x}(k)$
$n(k)$	The number of observations (success or failure) that has been taken under factor setting $\boldsymbol{x}(k)$
n	Sample size of the observation pairs
n_0	Upper limit for the number of observation pairs
s_1	The number of successes in the n trials for system under factor setting $\boldsymbol{x}(k_1)$
s_2	The number of successes in the n trials for system under factor setting $\boldsymbol{x}(k_2)$
a	Parameter which equals to ln (γ/α)
b	Parameter which equals to ln [(1-γ)/(1-α)]
v	Parameter which equals to $a + b$
r	Parameter which equals to $s_1 + s_2$
l	Parameter which equals to max(0, $r - n$)
u	Parameter which equals to min(n, r)
$C_L(r, n)$	The lower bound for test statistic when $n < n_0$
$C_U(r, n)$	The upper bound for test statistic when $n < n_0$

$C_L(r, n_0)$	The lower bound for test statistic when $n = n_0$
P(DI)	Probability of being declared important
P_{ij}	Transition probability between component i and j
R_s	Software reliability
R_i	Software component reliability
S	Software execution sample path
$Y(\boldsymbol{x}, S)$	Random output of software system with factor settings $\boldsymbol{x}$ and sample path S: 1 (success), 0 (failure)
$Y(\boldsymbol{x}(k), S)$	Random output of software system with factor settings $\boldsymbol{x}$ and sample path S: 1 (success), 0 (failure)
$P(\boldsymbol{x}, S)$	Success probability of software system with setting $\boldsymbol{x}$ and sample path S
d_i	The number of times that component i is visited by sample path S

CHAPTER 1: INTRODUCTION

1.1 Introduction of Software Reliability

Computers and computer systems play a significant role in the modern society. It is impossible to maintain this world running without the aid of computer systems controlled by software. In particular, complex systems such as national defense net, space shuttle launching, and oil refinery control, all heavily rely on computers and software systems.

The complexity of computer systems has grown dramatically in the previous decades. Representative examples can be easily found in projects undertaken by NASA, telecommunication industry, nuclear power generation plants, and a variety of other industries. For instance, NASA Space Shuttle flies with approximately 500,000 lines of software code on board and approximately 3.5 million lines of code in ground control and processing. The Windows XP (the 2001 version) personal computer operation system has more than 40 millions source lines of code, and Windows Server (the 2003 version) already has more than 50 millions source lines of code (Tanenbum, 2008).

Since the modern society is built on computer systems, reliable systems are highly

required. However, due to the competition between nations or business peers, the demand for complex computer systems has increased faster than the ability to design, test, and maintain them whereas the probability of software failures increases in parallel with the increased software complexity (Lyu, 1995a). This could lead to operation inconvenience, economy damage, and even human loss (Lyu, 1995b). In the NASA Voyager project, the Uranus encounter was jeopardized because of late software deliveries and reduced reliability of the Deep Space Network. On January 15, 1990, a fault in a switching system's new released software caused massive disruption of a major carrier's long-distance network, and led to enormous revenue losses to companies using this telecommunication company for business information transferring. The massive Therac-25 radiation therapy machine had enjoyed a perfect safety record until software errors in its sophisticated control systems malfunctioned and claimed several patients' lives in 1985 and 1986 (Lee, 1992). More recently, in 1996, the European Space Agency's one billion dollars prototype -- Ariane 5 rocket was destroyed less than a minute after launch, due to a bug in the on-board guidance computer program (Wikipedia, Software bug).

Clearly, software reliability can affect people's everyday living. Achieving highly reliable software has become a most challenge task in software development.

Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment (ANSI 1991). It is a key indicator of software quality which includes various customer satisfaction factors such as functionality, usability, performance, maintainability, and documentation.

Although everybody knows that software reliability is critical, and substantial effort has been devoted to reliability improvement in the software development cycle, it still remains a challenging task to achieve a high or desired reliability level. This is especially true with the more complex systems developed today. System developers tend to add complexity into software to accommodate the rapid growth of system size, the requirement of easier manipulation, and more frequent upgrading. For example, Windows 2000 operation system has more than 29 millions source lines of code. In 2001, Windows XP system was released with nearly 40 millions source lines of code. Now, the source code of new released Windows Vista system already grew to more than 50 millions lines (Wikipedia, Source lines of code).

Software failures may be due to errors, ambiguities, oversights or misinterpretation of the specification that the software is supposed to satisfy, carelessness or incompetence in writing code, inadequate testing, incorrect or unexpected usage of the software, or other unforeseen problems (Keiller 1991). The mechanism of

software failures is markedly different from that of traditional hardware. Most of hardware faults are physical faults, which is visible and relatively easier to classify, detect, and correct. However, software faults are design faults, which relate to human beings and design process. A well known bathtub curve for hardware reliability is shown in Figure 1.1 (Pan 1999) which illustrates the evolution of failure rate for hardware systems/components. The failure rate of hardware experiences a decrease in the burn-in phase, a constant level in the useful-life phase, and an increase in the wear-out phase. In Figures 1.1 and 1.2, λ is the steady state failure rate.

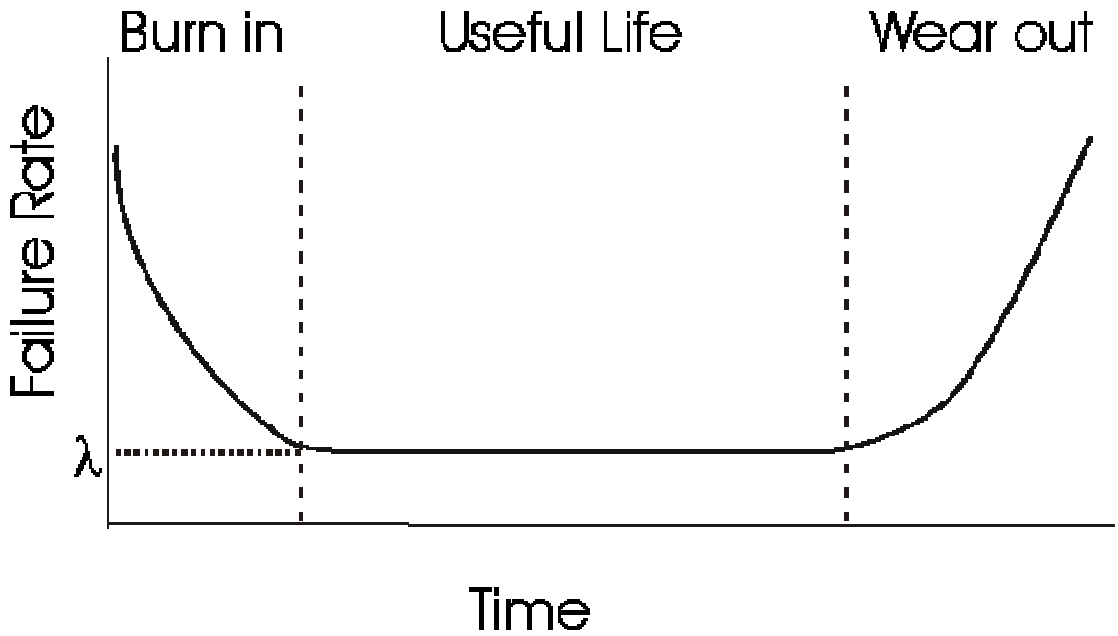

Figure 1.1: Bathtub curve for hardware reliability (Pan 1999)

However, software reliability has different characteristics. A possible curve is shown

in Figure 1.2 if software reliability is projected on the same axes (RAC 1996). Comparing these two figures, it is easy to find that there are two major differences. First, in the last phase, software failure rate does not experience an increase as hardware does, which means software will never be worn out. Once a software is uploaded, its failure rate stays unchanged unless upgrade takes place. Second, software will typically be upgraded during its in-use period, and each upgrade will cause an increase in the failure rate, which will gradually stabilize to a new level via debugging.

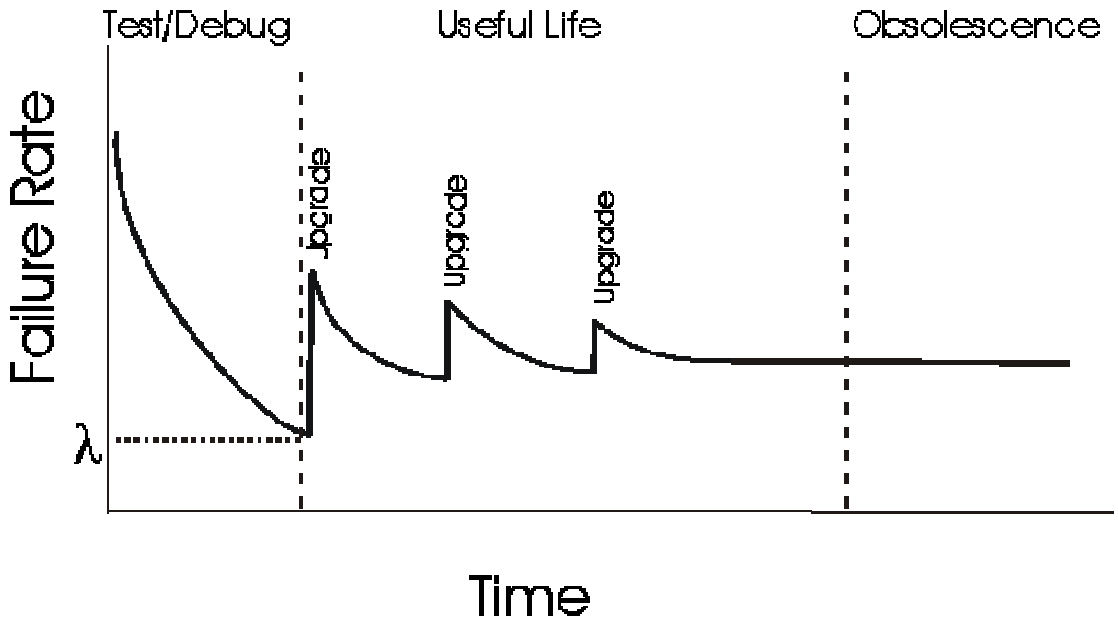

Figure 1.2: Revised bathtub curve for software reliability (RAC 1996)

1.2 Literature Review

1.2.1 Software Reliability Modeling

Software reliability research grows in compliance with the computer system development. Different software reliability assessment models have been developed to analyze and predict software reliability. Existing software reliability models can be divided into two groups, black box (functional) and white box (structural) models.

Black-box models treat software as a monolithic whole, which models the system primarily in terms of its input and output characteristics. The term "black box" is used because these models consider only the software's interaction with external environment without examining the specific execution process inside the "box". The well-known reliability growth models fall into the category of black-box models (Lyu, 1995a).

With the ever increasing software complexity, using one function to characterize the failure behavior of software systems becomes inappropriate and insufficient. Hence, white-box, as opposed to black-box, models have been developed and implemented in software reliability engineering (Cheung 1980, Krishnamurthy 1997, Gokhale 2004 and Grassi 2005). The white-box models treat the system as a composition of software components with interactions among them, and investigate how these

components could affect the overall system performance individually and interactively. When a software is being executed, different components are called following a certain sequence, and the software reliability depends on the successful execution of components and control transfer between components. In the literature, analytical models (such as discrete time Markov chain (DTMC), continuous time Markov chain (CTMC), and semi-Markov process (SMP)) have been used to model these structure-based software systems. Examples are Cheung (1980), Wang (2005), and Goseva-Popstojanova (2004). These analytical models rely heavily on simplifying the assumptions of software systems in order to be able to provide analytical solutions. For instance, the Markov property requires that the conditional distribution of any future state M_{i+1} given the past states M_0, M_1, …, M_{i-1} and the present state M_i, is independent of the past states and depends only on the present state. In the context of software execution, this means that the next component to be executed depends only on the current component being executed. However, in real situation, an execution may visit components based on several components that it has visited. Also it is difficult for current analytical models to deal with a software system with large state space. Recently, discrete event simulation has gained more attention, and it offers an attractive alternative to analytical models since it is able to accommodate important complexities that are present in realistic systems. For instance, Gokhale (2005) proposed simulation-based procedures to assess the impact

of fault detection and repair strategies on the system reliability; Gokhale et al. (2005) developed dynamic simulation procedures to model the software behavior throughout its development cycle. However, the current use of simulation for software reliability analysis calls for more sophisticated design of experiments and statistical methodologies to improve the computational efficiency of simulation and to ensure the validity of the output analysis.

1.2.2 Factor Screening

Screening experiments are designed to investigate the controllable factors in an experiment with a view toward eliminating the unimportant ones. According to the sparsity of effects principle, in many cases only a few factors are responsible for most of the response variation (Myers and Montgomery, 2002). Important factors shall be identified correctly and efficiently in screening experiments especially when dealing with complicated systems with large number of factors.

Many simulation procedures have been developed in factor screening experiments by using economical number of design points and replications (Trocine and Malone, 2000, 2001; Morris, 2005). For example, the first stage of response surface methodology is usually a factor screening. However, these procedures just emphasize physical experiments without taking advantage of the highly sequential

nature of the simulation experiments. Recent research has started to combine screening experiments and a follow-up response exploration into one design to screen out the important factors (Cheng and Wu, 2001).

Group-screening methods have been developed to deal with systems with large number of factors. The fundamental idea is to identify factors as a group to save experimental effort (Lewis and Dean, 2001). In group screening procedure, subgroups should be further tested if a factor group is identified as important. Otherwise, all factors in that group will be classified as unimportant. In group screening experiments, all factors must have their effects in the same direction in order to avoid cancellation; and a main-effects model is typically assumed (Trocine and Malone, 2001; Dean and Lewis, 2005).

The Sequential Bifurcation (SB) procedure is a combination of group screening and a sequential step-down procedure (Bettonvil and Kleijnen, 1997). A sequential design is one in which the design points are selected as the experiment results become available (Wan, 2006). SB is a series of steps. In each step, the group effect is tested for importance. As the experiment proceeds, the groups become smaller until all factors have been classified. This method was first developed for deterministic computer simulations. Later the method was extended to cover

stochastic simulations (Cheng, 1997; Kleijnen et al., 2006).

Wan (2006) modified the SB procedure for stochastic simulations and called it Controlled Stochastic Bifurcation (CSB). CSB procedure is a two-stage testing procedure to control the probability of Type I error and the power of the test in each bifurcation step. In the two-stage testing procedure, the determination of the second-stage sample size is based on a worst-case scenario. A fully sequential test was also implemented in CSB by Wan to give the same error control. In most cases, the sequential test is more efficient than the two-stage testing procedure (Wan, 2006). The CSB procedure is selected in this research for factor screening.

1.3 Problem Statement

This research intends to provide efficient simulation-based statistical procedures for the sensitivity analysis of software reliability systems.

The software reliability is studied via a structure-based software reliability model as described already in white-box models. The application is executed in such a way that components are invoked sequentially and stay active for a specific duration of time performing the requested functions. Suppose that a terminating application is considered which consists of a finite number of components, the software reliability is defined as the probability of successful execution of the software application.

In this research, the system's performance of interest is the software reliability, and the input factors considered include the reliability of each component in the system. The objective is to develop simulation-based factor screening procedures to classify the system components into two groups, important and unimportant, based on how sensitive the system reliability is to each component's reliability.

Assessing the importance of each software component will be very useful for creating a plan detailing which tasks should be performed to achieve a good system performance. For example, if it is determined that the reliability of a specific component has the most impact on the system reliability, then it is critical for the software testing-team to investigate the failure behavior of that component more thoroughly or to allocate more resources for this component for its reliability improvement. In addition, conducting sensitivity studies provides a way to assess the uncertainty in software reliability estimates.

1.4 Overview of Methodology

To achieve the objective of this research, first a simulation model will be built to represent the execution process of the software application. Once the model is built, simulation experiments will be performed to estimate the software reliability under

different settings of the investigated factors. The output response of a simulation run is represented by a binary random variable with two possible outcomes, success or failure of the software execution. However, the prevalent and "naive" method of assessing the impact of factors on software reliability by varying one factor at a time could be neither efficient nor effective, especially when people are interested in the effects of large number of factors potentially influencing the system's performance.

In this research, the Control Sequential Bifurcation (CSB) developed by Wan et. al (2006, 2007) will be adopted as the factor-screening framework, and Meeker's sequential ratio test will be embedded in the CSB procedure to control the probability of misclassifying factors.

CSB (Wan et. al, 2006) extends the basic Sequential Bifurcation (SB) procedure (Bettonvil and Kleijnen, 1997) to provide error control for random responses. Factors will be grouped and the aggregated group effects will be tested. If the group effect is classified as important, the group will be split into two smaller groups for further testing. If the group effect is classified as unimportant, all factors in the group will be classified as unimportant and no further testing will be needed. It is obvious that the effects of all factors must have the same direction so that no cancellation will happen. The CSB procedure, with its assumptions, will be

discussed in detail in Chapter 2. When only a small fraction of the factors are important, CSB can eliminate unimportant factors in groups and hence ends up requiring significantly less computational efforts than traditional methods. With the incorporation of a multi-stage hypothesis-testing approach into sequential bifurcation, CSB is the first screening strategy to simultaneously control the probability of type I error for each factor and the power for each bifurcation step under heterogeneous variance conditions. The CSB procedure will be used to screen factors for systems where only main factor effects are significant.

There are several challenges in applying CSB procedure directly to the software reliability problems. Specifically, the hypothesis testing procedures developed for CSB procedure with the error controls (which determine the error control property of the CSB procedure) is for normal responses. However, the response of software reliability system is binary (success (1) vs. failure (0)). In this research, Meeker's (1981) sequential ratio test will be adopted and embedded in the CSB framework to handle the situations where the output performance is binary.

1.5 Thesis Organization

The remainder of the thesis is organized as follows: In Chapter 2, the proposed

factor screening procedure is discussed in details; chapter 3 shows two empirical case studies used to evaluate the performance of the developed procedure; in Chapter 4, one software system, which has been studied in several articles in the literature, is used to evaluate the effectiveness of the developed factor screening procedures.

CHAPTER 2: RESEARCH APPROACH

In this chapter, the proposed simulation-based factor screening method is presented.

2.1 Response Model

Suppose that there are K independent factors in the simulation experiment: $\boldsymbol{x} = (x_1, x_2, \ldots, x_K)$. The simulation output of interest Y is a binary random variable with distribution parameter p: $Y = 1$ with probability p and $Y = 0$ with probability $1 - p$. In this research, it is assumed that the underlying input-output relationship can be approximated by models with main effects.

2.1.1 Main-Effect Model

It is assumed that the functional relationship between the probability p and the factors $\boldsymbol{x}$ can be approximated by a logistic regression model:

$$\frac{p(\boldsymbol{x},\boldsymbol{\beta})}{1-p(\boldsymbol{x},\boldsymbol{\beta})} = \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_K x_K) \tag{2.1}$$

Where $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_K)$ are the unknown coefficients; $p(\boldsymbol{x}, \boldsymbol{\beta})$ is the software reliability p with factors $\boldsymbol{x}$ and coefficients $\boldsymbol{\beta}$. Hence the software reliability p depends on the factors $\boldsymbol{x}$ through the linear combination $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_K x_K$. No interaction effect among the factors is considered. Wan (2005) discussed two situations in which main-effects models are appropriate: when there is little prior

knowledge about the system and a gross level of screening is desired; or when the goal of screening is to identify which factors have important local effects. The latter application to identify factors with important local effects (i.e., sensitivity analysis), is the main focus in this study.

The ratio $p(\boldsymbol{x}, \boldsymbol{\beta})/(1- p(\boldsymbol{x}, \boldsymbol{\beta}))$ is a continuous and monotonically increasing function of the probability p. In this study, the ratio is the primary response of interest. Denote $\boldsymbol{x}$ as factor vector ($x_1, x_2, \ldots, x_K$), and $\boldsymbol{x_k}$ as the vector ($x_1, x_2, ..., x_k+1, x_{k+1}, \ldots, x_K$). The effect on the ratio of increasing factor k by one unit is quantified as follows:

$$\frac{p(\boldsymbol{x_k}, \boldsymbol{\beta})/(1-p(\boldsymbol{x_k}, \boldsymbol{\beta}))}{p(\boldsymbol{x}, \boldsymbol{\beta})/(1-p(\boldsymbol{x}, \boldsymbol{\beta}))}$$

$$= \frac{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k (x_k + 1) + ... + \beta_K x_K)}{\exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + ... + \beta_K x_K)} \qquad (2.2)$$

$$= \exp(\beta_k)$$

In (2.2), the odds ratio of two ratios obtained at two different factor settings is calculated.

2.1.2 Determination of the Perturbation Levels

The basic idea of evaluating the effect of changing one factor on the system performance is to introduce a small disturbance to its nominal level setting and

estimate the resulting change in the output performance. In order to compare the effects of all the factors, the amount of disturbance to be introduced for each factor needs to be determined properly. Wan et al. (2006) proposed a cost model which determines the perturbation levels for the factors based on the required cost of changing a factor to produce a change in the output performance. A brief review of the cost model is given below.

Let c_i be the cost per unit change of factor x_i for $i = 1, 2, ..., K$, and $c^* = \max c_i$. Set ζ_i as the nominal level setting of factor x_i. Then the disturbance introduced to each factor is represented as $\Delta\zeta_i$ which is calculated as:

$$\Delta\zeta_i = \begin{cases} c^*/c_i & \text{continuous factor setting} \\ \lfloor c^*/c_i \rfloor & \text{discrete factor setting} \end{cases}$$

$\Delta\zeta_i$ is the maximum change in factor x_i that can be achieved without exceeding a cost c^*. For instance, suppose that there are $K = 3$ factors. The setting of the first factor can be changed continuously, but the other two are discrete. If $c_1 = 300$, $c_2 = 500$, and $c_3 = 800$, then $c^* = 800$, $\Delta\zeta_1 = 8/3$, $\Delta\zeta_2 = 1$, and $\Delta\zeta_3 = 1$.

2.1.3 Thresholds of Importance

Based on the cost model introduced above, the thresholds of importance are defined for the factors being considered.

- t_0: the minimum odds ratio value that people could be willing to spend c^* to obtain.
- t_1: the odds ratio value that people would not want to miss if it could be achieved for only a cost of c^*.

The integration of cost and thresholds of importance into the factor scaling provides a general way to determine the levels for each factor prior to running the simulation. The cost model provides a basis for fairly comparing the effects of factors in practice. However, if the experimenters already know the thresholds of importance as well as the factor levels, they do not need to use the cost model. Without loss of generality, it is assumed in response model (2.1) that the factor level settings of $\boldsymbol{x}$ are deterministic and coded as 0 (nominal level), 1 (nominal level + perturbation).

2.2 Factor Screening Procedure

The goal of screening is to identify the factors with important main effects assuming that the underlying input-output relationship can be approximated by a main-effect model. For each factor group which contains one or more factors, the importance of the group effect will be tested:

H_0: The group effect is not important

H_1: The group effect is important

The objective of screening procedure is to classify the factors being considered into

two groups: those that are unimportant, which means $\exp(\beta_k) \leq t_0$, and those that are important, meaning that $\exp(\beta_k) \geq t_1$. For factors with main effects $\leq t_0$, the probability of declaring them important (Type I Error) is controlled to be $\leq \alpha$; and for factors with effects $\geq t_1$, the power for identifying them as important to be $\geq \gamma$. Those factors whose effects fall between t_0 and t_1 are considered important and require reasonable, although not guaranteed, power to identify them.

2.2.1 CSB Review

Wan et al. (2006) proposed a factor-screening framework called Controlled Sequential Bifurcation (CSB), which is illustrated in Figure 2.1. Suppose that there are a total of K factors. CSB is a series of steps. In each step, the cumulative effect of a group of factors is tested for importance. Initially, all factors of interest are tested in a single group for the group's effect. If the group's effect is important, then the group is split into two subgroups. The effects of these two subgroups are then tested in subsequent steps and each subgroup is treated in the same way: either classified as unimportant or split into subgroups for further testing. This process continues until every individual factor has been classified as either important or unimportant.

Initialization: Create an empty LIFO queue for groups. Add the entire group $\{1,2,...,K\}$ to the LIFO queue.

While queue is not empty, do

Remove: Remove a group from the queue.

Test:

Unimportant: If the group is unimportant, then classify all factors in the group as unimportant.

Important (size = 1): If the group is important and of size 1, then classify the factor as important.

Important (size > 1): If the group is important and size is great than 1, then split it into two subgroups such that all factors in the first subgroup have smaller indices than those in the second group. Add each subgroup to the LIFO queue.

End Test

End While

Figure 2.1: Structure of CSB procedure (Wan et al. 2006, 2007)

In the CSB group-screening procedure, if one group is identified as unimportant, then all factors in this group are declared unimportant; if one group is identified as important, then further screening will be performed to identify the importance of the subgroups or individual factors within this group. Note that the group screening described in Figure 2.1 is based on the assumption that the main effects of all factors have the same sign in order to avoid cancellation. In Wan et al. (2006), without loss of generality, it was assumed that the main effects of all factors were nonnegative; increasing any factor by one unit would lead to a nonnegative increase in the output performance. This assumption is likely to hold in many realistic situations including

the software reliability systems, which are of particular interest to this research. Apparently, improving the reliability of an individual component (a factor) will have a nonnegative effect on the reliability of the entire software system (output performance of interest).

To illustrate the CSB procedure in Figure 2.1, a simple example is given as follows. Suppose that 10 factors are considered for a certain system with predetermined factor effects specified in Table 2.1. For instance, one unit improvement in factor 3 will improve the whole system performance by 2 units; one unit improvement in factor 7 will improve the whole system performance by 30 units. In this illustrating example, the importance threshold is set as 8, which means that a factor will be considered as important if improving this factor by one unit will lead to 8 units of improvement in the system performance. Based on the predetermined factors effects given in Table 2.1, only factor 7 is important. In this case, the factor screening process is presented in Figure 2.2. Initially, all 10 factors are grouped together and the group effect is tested for importance. Because the cumulative effect of this 10-factor group is 42, greater than the threshold 8, the group is declared important and split into two subgroups (factors 1-5 and factors 6-10) for further testing. For the subgroup with factors 1-5, the cumulative effect is 7, less than the threshold of importance, and hence this group along, with all the factors contained in it, is

declared unimportant. The subgroup with factors 6-10 has a cumulative effect of 35, greater than the threshold, and is thereby declared important and split into two subgroups for further screening. This process continues until all individual factors have been classified as important or unimportant. As a result, the only factor declared as important is factor 7.

Table 2.1 Factor effects in a simple example illustrating CSB factor screening procedure

Factors	Effects of improving the factor by one unit on the performance	Factors	Effects of improving the factor by one unit on the performance
Factor 1	1	Factor 6	1
Factor 2	1	Factor 7	30
Factor 3	2	Factor 8	1
Factor 4	2	Factor 9	2
Factor 5	1	Factor 10	1

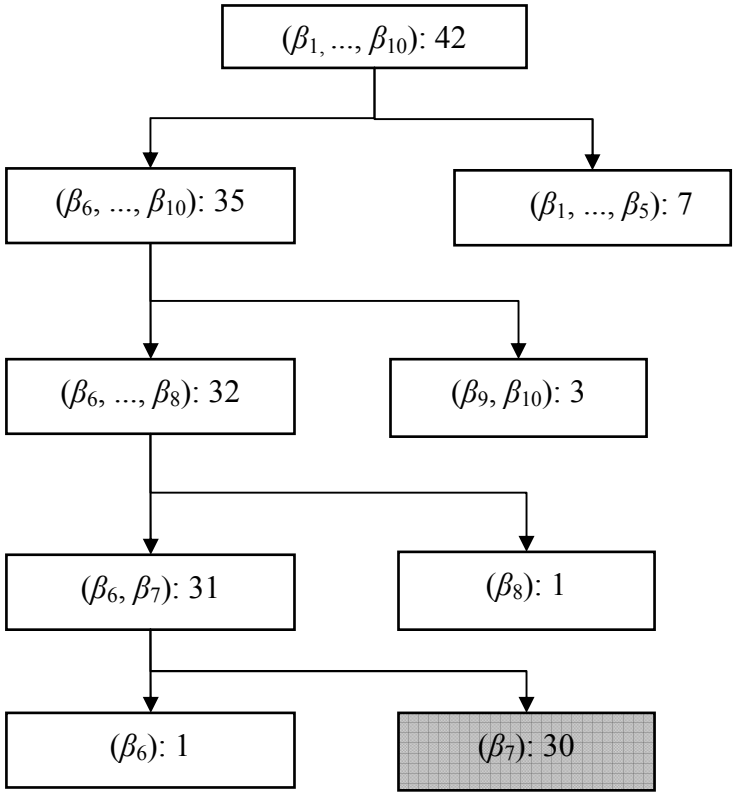

Figure 2.2 Whole factor screening process of a simple example illustrating CSB procedure

As illustrated in Figure 2.1, CSB is a top-down framework for testing the importance of factor groups. Next, the details for the hypothesis tests are discussed.

2.2.2 Hypothesis Test for Main-Effect Model

As mentioned earlier, the hypothesis test for group importance embedded in the existing CSB procedure is for normal responses, whereas in the software reliability analysis, the outcome is binary, success or failure. The current CSB screening procedure is extended to handle binary outcomes. For the testing of group

importance, the fully sequential test proposed by Meeker (1981) has been embedded in the CSB, which ensures the desired error control. In the proposed method, the Meeker's hypothesis test for binary outputs is performed based on sequential sampling via computer simulation. Specifics of the proposed method are given below.

Suppose that the group containing factors $\boldsymbol{x}_i$ ($k_1+1 \le i \le k_2$) is under consideration in a certain step of the CSB procedure. The cumulative effect of this factor group needs to be tested for importance. Define the factor setting $\boldsymbol{x}(k)$ as follows:

$$\boldsymbol{x}(k) = \begin{cases} 1, & i = 1, ..., k \\ \\ 0, & i = k+1, ..., K \end{cases}$$

Hence, $\boldsymbol{x}(k)$ is the vector with the first k factors set at their high levels, and the rest set at their nominal levels. The cumulative effect of factors $\boldsymbol{x}_i$ ($k_1+1 \le i \le k_2$) is evaluated based on the system performances, i.e., the odds ratios defined in Section 2.1.1, at two different factor settings $\boldsymbol{x}(k_1)$ and $\boldsymbol{x}(k_2)$. Assuming the main-effect model (2.1), the ratio of the two odds ratios at $\boldsymbol{x}(k_2)$ and $\boldsymbol{x}(k_1)$ are given as:

$$t = \frac{p(\boldsymbol{x}(k_2), \boldsymbol{\beta})/(1 - p(\boldsymbol{x}(k_2), \boldsymbol{\beta}))}{p(\boldsymbol{x}(k_1), \boldsymbol{\beta})/(1 - p(\boldsymbol{x}(k_1), \boldsymbol{\beta}))}$$

$$= \frac{\exp(\beta_0 + \beta_1 + \beta_2 + ... + \beta_{k_1} + ... + \beta_{k_2})}{\exp(\beta_0 + \beta_1 + \beta_2 + ... + \beta_{k_1})} \tag{2.3}$$

$$= \exp(\sum_{i=k_1+1}^{k_2} \beta_i)$$

The relative superior measure t is used to evaluate the improvement in the system performance by improving the factors $\boldsymbol{x}_i$ (k_1+1 ≤ i ≤ k_2). If t =1, it means that improving factors in this group does not lead to any improvement in the system performance. If t >1, then the system has better performance under factor setting $\boldsymbol{x}(k_2)$ than under factor setting $\boldsymbol{x}(k_1)$. Following Meeker's notation, the hypothesis test to determine whether this factor group is important is:

$$H_0 : t \le t_0 \text{ vs. } H_1 : t \ge t_1 \tag{2.4}$$

Where t_0 and t_1 are the user-specified thresholds of importance defined in Section 2.1.4 with 1 < t_0 < t_1. The factors group with $t = \exp(\sum_{i=k_1+1}^{k_2} \beta_i) \le t_0$ is considered as unimportant, and the factors group with $t = \exp(\sum_{i=k_1+1}^{k_2} \beta_i) \ge t_1$ is considered as critical. The probability of type I error, α, is the probability of rejecting H_0 when $t \le t_0$, and the power of the test $\gamma(t)$ is the probability of accepting H_1 at t, when $t \ge t_1$.

Let $y_i(k_1)$ (or $y_i(k_2)$) denote the binary random output of the system obtained from the

i^{th} (i = 1, 2, 3, …, n) simulation run under factor setting $\boldsymbol{x}(k_1)$ (or $\boldsymbol{x}(k_2)$). In simulation experiments, the observations M_n={($y_1(k_1)$, $y_1(k_2)$), ($y_2(k_1)$, $y_2(k_2)$), …, ($y_n(k_1)$, $y_n(k_2)$)} are obtained in pairs, and n is referred to as the size of the sample. The sequential method for testing the hypothesis (2.5) may be described as follows. Initially, a sample of a certain size is obtained via simulation. Based on the current sample M_n, three decisions will be made: (i) to accept H_0, (iii) to reject H_0, (iii) to continue the experiment by making an additional observation pair ($y_{n+1}(k_1)$, $y_{n+1}(k_2)$) and set $M_n = M_{n+1}$. This process is continued until a decision of type (i) or (ii) is made, or the sample size reaches a pre-specified upper limit n_0.

The testing process is illustrated in Figure 2.2, which is adapted from Meeker (1981) to suit the CSB structure. For details of the sequential hypothesis test, please refer to Meeker (1981). Necessary notations are given below.

- α: required probability of Type I Error
- γ: required power of the test at t_1
- $y_i(k)$: the i^{th} observation at factor level settings k, $y_i(k)$ = 1 (success) or 0 (failure)
- $n(k_1)$: the number of observations (success or failure) that have been taken under factor setting $\boldsymbol{x}(k_1)$
- $n(k_2)$: the number of observations (success or failure) that have been taken under factor settings $\boldsymbol{x}(k_2)$

As already mentioned, the observations are obtained in pairs to compare the system performance with factor settings $\boldsymbol{x}(k_1)$ and $\boldsymbol{x}(k_2)$. However, a certain number of observations $n(k_1)$ (or $n(k_2)$) may already have been obtained in the previous test processes. $n(k_1)$ and $n(k_2)$ are used to store all observations obtained so far at factor settings $\boldsymbol{x}(k_1)$ and $\boldsymbol{x}(k_2)$.

- n: sample size of the observation pairs $M_n=\{(y_1(k_1), y_1(k_2)), (y_1(k_1), y_1(k_2)), \ldots (y_n(k_1), y_n(k_2))\}$ used in the hypothesis test; initially, n may be $\min\{n(k_1), n(k_2)\}$ if $n(k_1) \neq n(k_2)$.
- n_0: upper limit for the number of observation pairs; the sequential test procedure will be terminated once the sample size reaches this upper limit.

The conditional sequential test is truncated at observation n_0. The truncation effect was discussed in Meeker (1981). In this research, n_0 is set at 10,000. Case studies in Chapter 3 show good control of probability of Type I error and the power of the test when the upper limit is set at 10,000.

- $s_1 = \sum_{i=1}^{n} y_i(k_1)$: the number of successes in the n trials for system under factor setting $\boldsymbol{x}(k_1)$.
- $s_2 = \sum_{i=1}^{n} y_i(k_2)$: the number of successes in the n trials for system under factor setting $\boldsymbol{x}(k_2)$.
- $a = \ln(\gamma/\alpha)$
- $b = \ln[(1-\gamma)/(1-\alpha)]$

- $r = s_1 + s_2$: the total number of successes under both factor settings with s_1 and s_2 defined above.

Parameters a, b and r are used to calculate the lower bound $C_L(r, n)$ and upper bound $C_U(r, n)$ as shown below.

- $l = \max(0, r - n)$
- $u = \min(n, r)$

Parameters l and u are used in calculation of function $F(\tau)$.

- $F(\tau) = F(r, n, \tau)$ is a function of r, n , and τ; τ can only be equal to t_0 or t_1

$$F(r,n,\tau) = \ln\left(\sum_{j=l}^{u}\binom{n}{j}\binom{n}{r-j}\tau^{j}\right)$$

- $C_L(r,n) = \lfloor \{b + F(t_1) - F(t_0)\} / \ln(t_1 / t_0) \rfloor$, the lower bound for the test statistic before the number of pair observations n meets the upper limit n_0.
- $C_U(r,n) = \lfloor \{a + F(t_1) - F(t_0)\} / \ln(t_1 / t_0) \rfloor + 1$, the upper bound for the test statistic before the number of pair observations n meets the upper limit n_0.
- $v = (a + b) / 2$
- $C_L(r,n_0) = \lfloor \{v + F(t_1) - F(t_0)\} / \ln(t_1 / t_0) \rfloor$, the lower bound for the test statistic when the number of pair observations n meets the upper limit n_0; the upper bound for test statistic is calculated as $C_L(r, n_0) + 1$ when the number of pair observations n meets the upper limit n_0.

Test Initialization Set $s_1 = 0$, $s_2 = 0$, $r = 0$, *finish* = 0, n_0 is a user-specified parameter. $n(k_1)$ and $n(k_2)$ are given from the previous steps of the CSB procedure (Figure 3.1)

If $n(k_1) = 0$ or $n(k_2) = 0$: $n = 1$

Else $n = \min(n(k_1), n(k_2))$

While *finish* < 1 AND $n < n_0$, **do**

If $n(k_1) < n$:

Take one observation $y_n(k_1)$ under factor setting $\boldsymbol{x}(k_1)$; $n(k_1) = n$

Endif

$s_1 = \sum_{i=1}^{n} y_i(k_1)$

If $n(k_2) < n$:

Take one observation $y_n(k_2)$ under factor setting $\boldsymbol{x}(k_2)$; $n(k_2) = n$

Endif

$s_2 = \sum_{i=1}^{n} y_i(k_2)$

$r = s_1 + s_2$

Unimportant: If $s_2 < C_L(r, n)$, classify the group as unimportant, *finish* = 1

Important: ElseIf $s_2 > C_U(r, n)$, then classify the group as important, *finish* = 1

Endif

$n = n + 1$;

End While

If $n = n_0$

Unimportant: If $s_2 \leq C_L(r, n_0)$, classify the group as unimportant.

Important: ElseIf $s_2 \geq C_L(r, n_0) + 1$, then classify the group as important.

Endif

Figure 2.3: Meeker's fully sequential test adapted for the CSB factor screening.

The sequential procedure in Figure 2.3 is used to test the importance of the factor group containing factors $\boldsymbol{x}_i$ ($k_1+1 \leq i \leq k_2$). Before testing the superiority between the

two systems with factor settings $\boldsymbol{x}(k_1)$ and $\boldsymbol{x}(k_2)$, one of these two systems or even both may already have been involved in the testing of other factor groups. Hence, in the initialization of the procedure, the numbers of observations already obtained, $n(k_1) \geq 0$ and $n(k_2) \geq 0$, are given. If $n(k_1) = 0$ or $n(k_2) = 0$, then the number of observation pairs initially available is 0; the procedure will first obtain n=1 observation pair by running a simulation, and the sequential test will first be performed based on a single pair of observations. Otherwise, if $n(k_1)>0$ and $n(k_2)>0$, then $n = \min(n(k_1), n(k_2))$ observation pairs are initially available, based on which the sequential test will be initiated. Other parameters s_1, s_2, r, *finish*, and n_0 are set as shown in Figure 2.3. When $n < n_0$, test decisions will be made by comparing s_2 with $C_L(r, n)$ and $C_U(r, n)$; once sample size n reaches the upper limit n_0, test decision will be made by comparing s_2 with $C_L(r, n_0)$ and $C_L(r, n_0) + 1$.

It is worth mentioning that the Wald sequential test procedure (Wald 1947) can also be used to test the difference between the means of two binominal distributions. However, Wald test only utilizes the untied observation pairs (i.e., (1, 0) and (0, 1) pairs) while disregarding the tied observation pairs. Hence, Wald's test is not as efficient as Meeker's test (Meeker 1981) which uses both tied and untied observation pairs in the hypothesis test.

CHAPTER 3: EMPIRICAL EVALUATION

In this chapter, the performance of the factor screening procedures proposed in Chapter 2 is evaluated. Before applying the CSB procedure in a case study of a software system (Chapter 4), Monte Carlo simulation is used to generate data such that the size of the main effects can be fully controlled by setting the coefficients of the factors being considered.

3.1 Case 1

Consider the system with its input-output relationship specified as

$$\frac{p(\boldsymbol{x},\boldsymbol{\beta})}{1-p(\boldsymbol{x},\boldsymbol{\beta})} = \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_{10} x_{10}) \tag{3.1}$$

which follows the main-effect model (2.1). The coefficients β_i (i = 1, 2, …, 10) are given in Table 3.1.

The factor screening procedure described in Section 2.2 (CSB procedure with embedded Meeker's test) is applied to classify the 10 factors as important or not important. For an iteration in the CSB procedure where the importance of the factor group $\boldsymbol{x}_i$ (k_1+1 ≤ i ≤ k_2) is being tested, a simulation sampling is carried out as follows to obtain the binary output observations for testing the hypothesis (2.4). Under factor setting $\boldsymbol{x}(k_1)$, the probability $p(\boldsymbol{x}(k_1), \boldsymbol{\beta})$ can be calculated from (3.1). A

random binary output, which is considered as an observation, can be generated from the Bernoulli distribution with a success probability of $p(\boldsymbol{x}(k_1), \boldsymbol{\beta})$. Similarly, the output observations can be obtained via Monte Carlo simulation under factor setting $\boldsymbol{x}(k_2)$. Based on the simulation sampling, the factor group being considered will be classified as important or not important using the sequential testing procedure given in Figure 2.3.

Table 3.1: Factor effects in case 1

Coefficients β_i	Value	$\exp(\beta_i)$	Value
β_1	0.01	$\exp(\beta_1)$	1.01
β_2	0.05	$\exp(\beta_2)$	1.05
β_3	0.1	$\exp(\beta_3)$	1.11
β_4	0.15	$\exp(\beta_4)$	1.16
β_5	0.2	$\exp(\beta_5)$	1.22
β_6	0.25	$\exp(\beta_6)$	1.28
β_7	0.3	$\exp(\beta_7)$	1.35
β_8	0.35	$\exp(\beta_8)$	1.42
β_9	0.4	$\exp(\beta_9)$	1.49
B_{10}	0.45	$\exp(\beta_{10})$	1.57

In this experiment, the parameters for the CSB procedure are given in Table 3.2. With these user-specified parameters, the factors that are considered as not important are factors 1, 2, and 3 with $\exp(\beta_i) < t_0$ for $i = 1, 2, 3$ (Table 3.1); the critical factors are factors 7, 8, 9, and 10 with $\exp(\beta_i) > t_1$ for $i = 7, 8, 9, 10$ (Table 3.1). The CSB procedure is designed in such a way that the probability of misclassifying an

unimportant factor as important is less than 0.05, and the power of classifying a critical factor as important is greater than 0.95.

The CSB procedure was applied on this case for 1000 times using different random streams, and each time the factor screening results were recorded. For each factor, the $P(\text{DI})$, the probability of being declared as important, is estimated from these 1000 replications. For instance, if factor 5 is declared important 350 times out of 1000 times, its $P(\text{DI})$ is 0.35. Figure 3.1 plots the $P(\text{DI})$ of the factors against their true effects $\exp(\beta_i)$ for $i = 1, 2, ..., 10$. As can be seen from Figure 3.1, the $P(\text{DI})$ for unimportant factors with $\exp(\beta_i) < t_0$ was well below alpha 0.05, and the $P(\text{DI})$ for critical factors with $\exp(\beta_i) > t_1$ was well above gamma 0.95.

Table 3.2: Parameters for empirical evaluation experiment within CSB procedure based on logistic regression model

Parameter	Value
t_0	1.15
t_1	1.30
α	0.05
γ	0.95

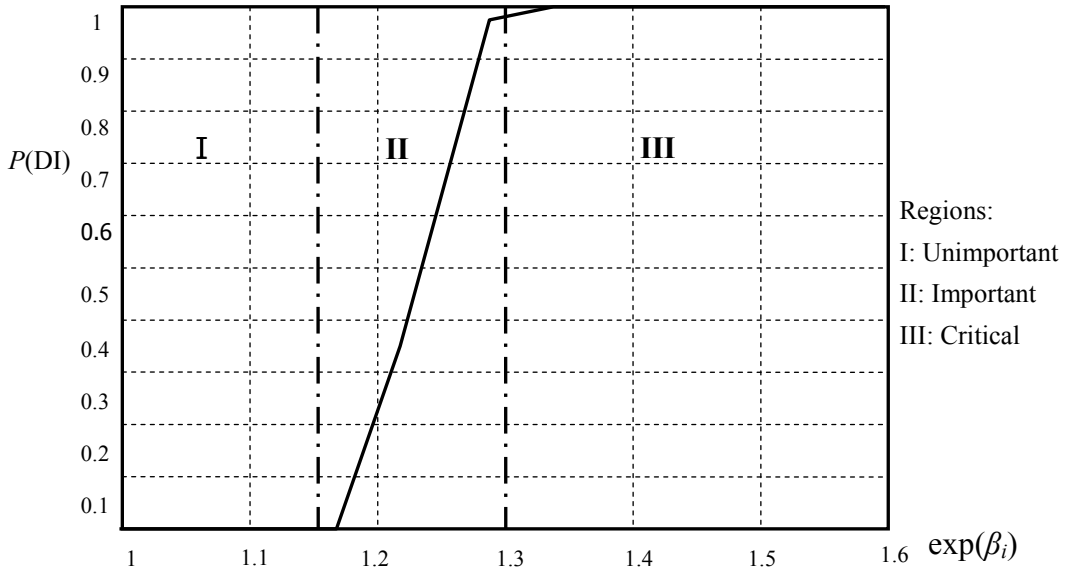

Figure 3.1: Case 1 factor screening results

3.2 Case 2

In this case, all the 10 coefficients β_i (i = 1, …, 10) in model (3.1) are set at 0.1. CSB parameters are given in Table 3.2. Hence, all the 10 factors are unimportant with $\exp(\beta_i) = 1.11 < t_0 = 1.15$. This case is designed to study the control of type I error for the factor screening procedure. Again, the CSB procedure was applied on this case for 1000 times, from which the P(DI) for each factor was estimated. From the results obtained in Table 3.3, the P(DI) for each of the 10 factors was below 5%, indicating that the type I error was well controlled as expected.

Table 3.3 P(DI) of factor i in case 2 out of 1000 replication

Factor i	exp(β_i)	P(DI)	Factor i	exp(β_i)	P(DI)
1	1.11	0	6	1.11	0
2	1.11	0	7	1.11	0
3	1.11	0	8	1.11	0
4	1.11	0	9	1.11	0.001
5	1.11	0.001	10	1.11	0

CHAPTER 4: CASE STUDY FOR A SOFTWARE SYSTEM

In this section, the effectiveness of the proposed simulation-based factor screening procedure is demonstrated through its application on a software reliability system. The software system reported by Cheung (1981) is used as an example, which has been used extensively in the literature to illustrate structure-based reliability assessment techniques (Gokhale and Trivedi 2002, Goseva-Popstojanova and Kamavaram 2003, and Lo et. al 2002).

4.1 Software Architecture

The structure of the application (Cheung 1981) is shown in Figure 5.1. The state-based approach, which uses the control flow graph to represent software architecture, is used to build the architecture-based software reliability model (Cheung 1981). The states represent active components 1, 2, …, and 10. The arcs represent the intercomponent transitions, and the transition probability P_{ij} represents the probability that component j is executed upon the completion of component i (i, j = 1, 2, …, and 10 where $i \neq j$). The non-zero transition probabilities between the components are given in Table 4.1. In practice, the parameters P_{ij} are estimated from the user operational profiles reflecting the usage of different components when the

software application is being executed. According to Figure 4.1, the software system consists of 10 components, and the execution of the application always starts with component 1 and ends with component 10. Furthermore, it is assumed that the application spends 1 time unit at each component per visit. Hence, the software execution process illustrated in Figure 4.1 can be modeled as discrete time Markov chain (DTMC) with transition probability matrix $P = (P_{ij})_{10*10}$.

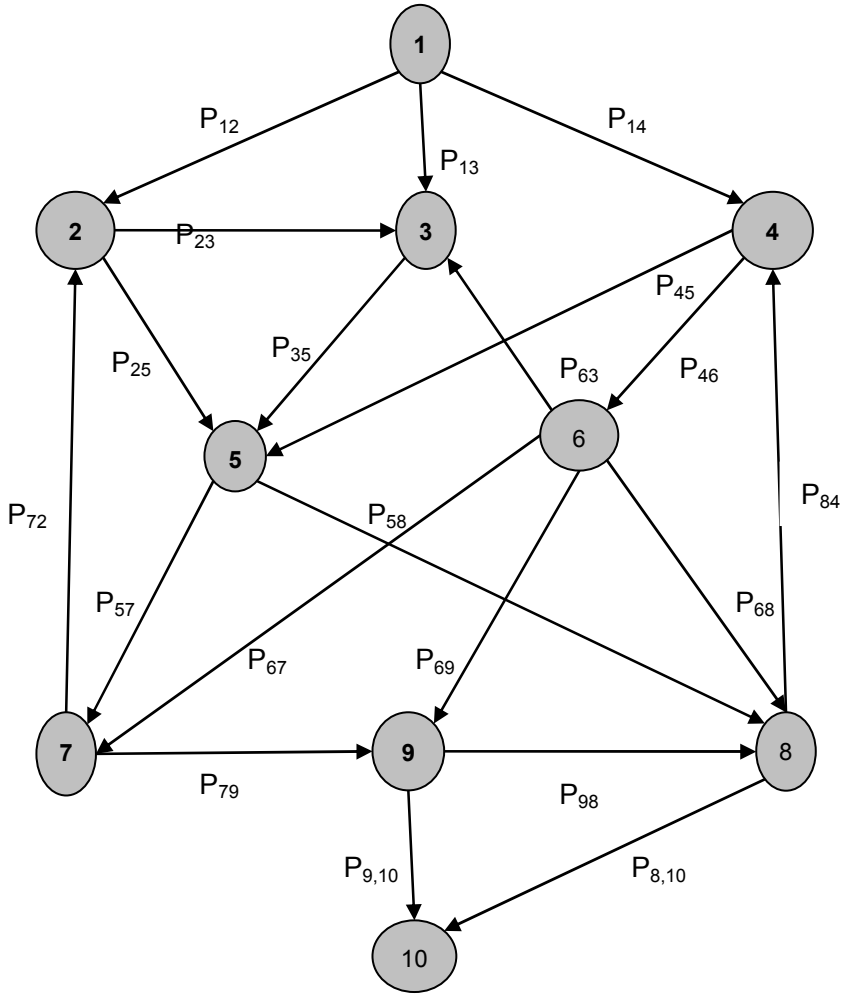

Figure 4.1: Software structure of an example application

Table 4.1: Intercomponent transition probabilities for the software example

$P_{1,2} = 0.60$	$P_{1,3} = 0.20$	$P_{1,4} = 0.20$	
$P_{2,3} = 0.70$	$P_{2,5} = 0.30$		
$P_{3,5} = 1.00$			
$P_{4,5} = 0.40$	$P_{4,6} = 0.60$		
$P_{5,7} = 0.40$	$P_{5,8} = 0.60$		
$P_{6,3} = 0.30$	$P_{6,7} = 0.30$	$P_{6,8} = 0.10$	$P_{6,9} = 0.30$
$P_{7,2} = 0.50$	$P_{7,9} = 0.50$		
$P_{8,4} = 0.25$	$P_{8,10} = 0.75$		
$P_{9,8} = 0.10$	$P_{9,10} = 0.90$		

4.2 Failure Behavior

The failure behavior of components is first considered, i.e., the reliability of each component. A component can fail during its execution period, which is assumed to be 1 unit of time. The reliability of a component is defined as the probability that the component performs its function correctly without a failure when being executed. Failures of different components occur independently from each other. The application process is considered a failure if any of the components called during the execution fails. Given that the model described in Figure 4.1 is a DTMC, the expression for system reliability as a function of transition probabilities and

component reliabilities can be analytically derived. In this case study, it is assumed that the transition probabilities are fixed known values, and try to evaluate the impact of component reliabilities upon the system reliability. Let R_i denote the reliability of component i (i = 1, 2, ..., 10), and R_s the reliability of the software system. Table 4.2 shows the component nominal level reliabilities of this 10-component software example. The functional relationship between R_s and $\{R_i, i = 1, 2, ..., 10\}$,

$$R_s = f(R_1, R_2, ..., R_{10}) \tag{4.1}$$

can be derived based on the DTMC, and hence the simulation-based factor screening results can be compared with the analytical ones obtained from DTMC.

It should be emphasized that the strength of the simulation-based method stems from its ability to analyze complicated and realistic software systems without having to use the simplifying assumptions required by analytical models (e.g, DTMC, CTMC). For the purpose of illustration and evaluation, the factor screening procedure is applied on this simple software application.

Table 4.2: Nominal factor settings for the component reliabilities

Factors x_i	x_1	x_2	x_3	x_4	x_5
Reliability R_i	0.986	0.985	0.985	0.97	0.95
Factors x_i	x_6	x_7	x_8	x_9	x_{10}
Reliability R_i	0.98	0.986	0.945	0.975	0.975

4.3 Simulating the Software Execution Process

The output of a simulation run of the software model is a Bernoulli random variable Y with success probability $p = R_s$. A sample path is defined as the sequence of components visited by an application execution. In other words, a sample path represents a calling sequence of the components when the software is being executed. Let $S = \{S_1, S_2, S_3, \ldots\}$ denote the set of all the possible sample paths (which may be infinite if there are loops) , and define

$$p(\boldsymbol{x}) = E_S[p(\boldsymbol{x}, S)] \tag{4.2}$$

Where $p(\boldsymbol{x}, S)$ represents the system reliability following sample path S, and E_S denotes the expected system reliability with respect to S. Further, $Y(\boldsymbol{x}, S)$ denotes the random output at factor setting $\boldsymbol{x}$ for a given sample path S. $Y(\boldsymbol{x}, S)$ follows a Bernoulli distribution with parameter $p(\boldsymbol{x}, S)$.

In light of the fact that the reliability of the system could differ markedly depending on the particular sample path S, in the sequential simulation, experiments are set up

in such a way that simulation replications at different factor level settings are performed at the same randomly selected sample paths. When comparing two factor level settings $\boldsymbol{x}(k_1)$ and $\boldsymbol{x}(k_2)$, random outputs $Y(\boldsymbol{x}(k_1), S_i)$ and $Y(\boldsymbol{x}(k_2), S_i)$ are taken sequentially with i increasing from 1 to n. If more than n runs are needed for the test, then a new sample path S_{n+1} will be generated and stored, and observations $Y(\boldsymbol{x}(k_1), S_{n+1})$ and $Y(\boldsymbol{x}(k_2), S_{n+1})$ are collected.

With the simulation strategy described above, the only dynamic simulation that needs to be carried out is the generation of a number of sample paths independently which will be used to produce the random outputs for the different factor settings. For a given sample path S, the output $Y(\boldsymbol{x}, S)$ can simply be generated using Monte Carlo Simulation as follows. Let d_i be the number of times that component i is visited by sample path S, then the reliability $p(\boldsymbol{x}, S)$ is given as

$$p(x,S) = \prod_{i=1}^{10} x_i^{\,d_i} \tag{4.3}$$

Thus the output $Y(\boldsymbol{x}, S)$ can be generated by drawing a random variable from the Bernoulli distribution with parameter $p(\boldsymbol{x}, S)$. The computational savings from this approach allows for the generation of the large number of sample paths.

4.4 Application of the CSB Procedure

The objective here is to classify the software components into two groups, important and unimportant, based on how sensitive the system reliability is to the reliability of each component. The factors considered are $\boldsymbol{x} = (x_1, x_2, \ldots., x_{10}) = (R_1, R_2, \ldots, R_{10})$. For the consistency of notation, $\boldsymbol{x}$ is used to represent the factor setting vector.

In this case study, nominal factor settings are given in Table 4.2, and the factor disturbance level of all factors is set as 0.013. The CSB procedure parameters for the experiment are given in Table 4.3. Based on the analytical results from the DTMC, the only component that quantifies as critical is component 5, components 1 and 10 are considered as important, and the rest of the components are not important. Simulation-based factor screening results are evaluated against the true analytical results. Again, the CSB procedure with 1000 simulation replications was applied, and Table 4.4 summarizes the factor screening results from these 1000 replications. Component 5 was identified as important with a probability of about 95% (956 out of 1000), lower than the desired power of 99%. For the unimportant components (component 2, 3, 4, 6, 7, 8, and 9), the probability of misclassifying components 4, 6, 7, and 9 were well below the desired level of 1%, and the probability of misclassifying components 2, 3, and 8 are close to 5%, above the desired level of 1%. Failing to achieve the desired probability of type I error for some components

(component 2, 3, and 8) may be attributed to the inadequacy of the assumption that there is no interaction between the factors that the main-effect model represents the input-output relationship for this system.

Table 4.3: Parameters for case study with CSB procedure application

Parameter	Value
t_0	1.07
t_1	1.085
α	0.01
γ	0.99

Table 4.4: Times of component *i* classified as important in case study out of 1000 replications with implementation of CSB procedure

Component *i*	Times	Component *i*	Times
1	272	6	0
2	38	7	0
3	48	8	54
4	0	9	0
5	956	10	289

CHAPTER 5: DISCUSSION AND CONCLUSION

In this research, the existing CSB factor screening procedure is extended to handle the cases where the system outputs are binary random variables rather than normal responses. The sequential tests developed by Meeker's (1981) is selected and embedded in the proposed factor screening procedures. Empirical evaluations of the procedures are performed on models with known results and on a software application system. Through numeric experiments, it is demonstrated that the developed factor screening procedures are able to classify factors as important or unimportant with pre-specified error control.

The limitations of the factor screening methods developed in this research are given as follows. (1) The efficiency of CSB procedure depends on the appropriateness of using the logistic regression model of the form (2.1) to approximate the underlying input-output relationship of the system being investigated. (2) CSB procedure is not universally good for all factor-screening problems: they are particularly efficient in dealing with systems with large number of factors, and a small number of factors being important.

The results in Chapter 4 show that sometimes the main effect model may not be adequate to approximate the input-output relationships for real systems. Future

studies should focus on developing factor screening procedures that can handle situations where factor interactions are present.

In practice, the software crash rarely happens for important systems, such as national defense net and space shuttle launching, since people pay much more effort in quality control in these systems. This makes the software crash a rare event and the study of factors that influence the crash probability more challenging. It is recommended to expand this effort in future research to screen important factors in systems that rarely fail.

REFERENCE

Anderson T. W., 1960, A Modification of the Sequential Probability Ratio Test to Reduce the Sample Size, *the Annals of Mathematical Statistics*, Vol. 31, 165-197

ANSI/IEEE, "Standard Glossary of Software Engineering Terminology", STD-729-1991, ANSI/IEEE, 1991

Armitage P., 1957, Restricted Sequential Procedures, *Biometrika*, Vol. 44, 9-26

Bettonvil B., Kleijnen J. P. C., 1997, Searching for Important Factors in Simulation Models with Many Factors: Sequential Bifurcation, *European Journal of Operational Research*, Vol. 96, No. 1, 180-194

Cheng, R. C. H., 1997, Searching for Important Factors: Sequential Bifurcation under Uncertainty, *In Proceeding of 1997 Winter Simulation Conference*, Piscataway, NJ, 275-280

Cheng, S., Wu, C. F. J., 2001, Factor Screening and Response Surface Exploration-Rejoinder. Statistica Sinica, Vol. 11, 553-604

Cheung R. C., A User-Oriented Software Reliability Model, 1980, *IEEE Trans. Software Engineering*, Vol. 6, No. 2, 118-125

Dean, A. M., Lewis, S. M., ed., 2005, Screening, Spring-Verlag, New York

Dickinson W., Leon D., and Podgurski A. 2001, Finding Failures by Cluster Analysis of Execution Profiles. In *Proceedings of the 23rd International Conference on Software Engineering*, ICSE 2001, 339-348

Everett W. W., 1999, Software Component Reliability Analysis, In *Proceedings 1999 IEEE Symposium on Application-Specific Systems and Software Engineering and Technology*, ASSET 99, 22-31

Gokhale S., and K. S. Trivedi. 2002. Reliability prediction and sensitivity analysis based on software architecture. In *Proceedings of the 13th International Symposium on Software Reliability Engineering*, 64-78.

Gokhale S. S., Wong W. E., Trivedi K. S., and Horgan J. R., 2004, *An Analytical Approach to Architecture-Based Software Reliability Prediction*, Performance Evaluation, Vol. 58, No. 4, 391-412

Gokhale S., and Lyu M. R. 2005, A Simulation Approach to Structure-Based Software Reliability Analysis, *IEEE Transaction on Software Engineering*. Vol. 31, No. 8, 643-656.

Goseva-Popstojanova K., Mathur A. P., and Trivedi K. S., Comparison of Architecture-Based Software Reliability Models, 2001, In *Proceedings of the International Symposium on Software Reliability Engineering*, 22-31

Goseva-Popstojanova K., and Kamavaram S. 2003, Assessing Uncertainty in Reliability of Component-Based Software Systems, In *Proceedings of the 14th International Symposium on Software Reliability Engineering*.

Goseva-Popstojanova K., Hamill M., and Perugupalli R., Large Empirical Case Study of Architecture-based Software Reliability, 2005, In *Proceedings International Symposium on Software Reliability Engineering,* 43-53

Goseva-Popstojanova K., and Kamavaram S., Software Reliability Estimation under

Uncertainty: Generalization of the Method of Moments, 2004, In *Proceedings of IEEE International Symposium on High Assurance Systems Engineering*, Vol. 8, 209-218

Goseva-Popstojanova K., Hamill M., and Wang X., Adequacy, Accuracy, Scalability, and Uncertainty of Architecture-based Software Reliability: Lessons Learned from Large Empirical Case Studies, 2006, In *Proceedings - International Symposium on Software Reliability Engineering*, 197-203

Grassi V., Architecture-Based Reliability Prediction for Service-Oriented Computing, Springer-Verlag Berlin/ Heidelberg, 2005, Vol.3549, 279-299

Hartmann M., 1991, An Improvement on Paulson's Procedure for Selecting The Population with The Largest Mean from *k* Normal Populations with A Common Unknown Variance, *Sequential Analysis*, Vol. 10, 1-16

Hoang Pham., Software Reliability, Springer, Singapore, 2000

Huang R., Lyu M. R., and Kanoun K., Simulation Techniques for Component-Based Software Reliability Modeling with Project Application, 2001, In *Proceedings of the International Symposium on Information Systems and Engineering*, 283-289

Kamavaram S., and Goseva-Popstojanova K., Sensitivity of Software Usage to Changes in the Operational Profile 2004, In *Proceedings. 28th Annual NASA Goddard Software Engineering Workshop*, 157-64

Keiller, Peter A., Miller, Douglas R., "On the Use and the Performance of Software Reliability Growth Models", *Software Reliability and Safety*, Elsevier, 1991, 95-117

Khoshgoftaar, T.M., and Munson, J.C., "A Measure of Software System Complexity

and Its Relationship to Faults," *Proceeding of the 1992 International Simulation Technology Conference*, Houston, TX, Nov. 1992, 267-272

Kleijnen, J. P. C., Bettonvil, B., Person F., 2006, Finding The Important Factors in Large Discrete-event Simulation: Sequential Bifurcation and Its Applications. Dean, A., Lewis, S., ed., Screening: Methods for Experimentation in Industry, Drug Discovery, and Genetics, Spring-Verlag, New York

Krishnamurthy S., and Mathur A. P., On the Estimation of Reliability of a Software System Using Reliability of its Components, 1997, In *Proceedings of the International Symposium on Software Reliability Engineering, ISSRE*, 146-155

Lee, L., The Day the Phones Stopped: How People Get Hurt When Computers Go Wrong, Donald I. Fine, Inc., New York, 1992.

Lo J., S. Kuo, M. R. Lyu, and C. Huang. 2002. Optimal resource allocation and reliability analysis for component-based software applications. In *Proceeding of the 26th Annual International Computer Software and Application Conference*, 7-12

Lewis, S. M., Dean, A. M., 2001, Detection of Interactions on Large Numbers of Factors, *Journal of the Royal Statistical Society: Series B*, Vol. 63, 633-672

Lyu, M. R., 1995a, Handbook of Software Reliability Engineering, New York: McGraw-Hill

Lyu, M. R., 1995b, Software Reliability: To Use or Not To Use? A Panel Discussion, http://www.stsc.hill.af.mil/crosstalk/1995/02/reliable.asp

Meeker, W. Q. Jr., 1981, A conditional Sequential Test for the Equality of Two Binomial Proportions, *Applied Statistics*, Vol. 30, No. 2, 109-115

Morris, M. D., 2006, An overview of group factor screening, Dean, A., Lewis, S., ed., Screening: Methods for Experimentation in Industry, Drug Discovery, and Genetics, Spring-Verlag, New York

Myers, R. H., Montgomery, D. C., 2002, Response Surface Methodology: Process and Product Optimization Using Designed Experiments, John Wiley and Sons, New York

Pan, J., Software Reliability: Dependable Embedded Systems, Carnegie Mellon University, 18-849b, 1999.

Paul R., editor. Software Reliability Handbook. Centre for Software Reliability, City University, London, U.K. 1990

Reliability Analysis Center, Introduction to Software Reliability: A state of the art Review. Reliability Analysis Center (RAC), 1996

Tanenbaum A., 2008, Modern Operation System, 3rd edition, Prentice Hall

Trocine, L., Malone, L. C., 2000, Finding Important Independent Variables Through Screening Designs: A comparison of methods, *In Proceeding of 2000 Winter Simulation Conference*, Piscataway, NJ, 749-753

Trocine, L., Malone, L. C., 2001, An Overview of Newer, Advanced Screening Methods for The Initial Phase in An Experiment Design, *In Proceeding of 2001 Winter Simulation Conference*, Piscataway, NJ, 169-178

Wald A., 1947, Sequential Analysis, Dover Publications, Inc., Mineloa, N.Y. 2004

Wang W., Pan D., and Chen M., 2005, Architecture-Based Software Reliability Model, *The Journal of Systems and Software*. Vol. 79, No. 1, 132-146.

Wan H., Ankenman B. E., and Nelson B. L., Controlled Sequential Bifurcation: A New Factor-Screening Method for Discrete-Event Simulation, 2006, *Operation Research*, Vol. 54, No. 4, 743-755

Wan H., Ankenman B. E., and Nelson B. L., 2007, Extended Control Sequential Bifurcation for Simulation Factor Screening in the Presence of Interactions. Working paper, School of Industrial Engineering, Purdue University

Wikipedia, Windows 7, 2008, http://en.wikipedia.org/wiki/Windows_7

Wikipedia, *Software bug*, 2008, http://en.wikipedia.org/wiki/Software_failure

Wikipedia, *Source lines of code*, 2008, http://en.wikipedia.org/wiki/Source_lines_of_code

Xu, J, Yang, F, Wan, H, Controlled Sequential Bifurcation for Software Reliability Study, *2007 Winter Simulation Conference*, Washington, DC, Dec. 2007, 281-288