Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

CONTROLLED SEQUENTIAL BIFURCATION FOR SOFTWARE RELIABILITY STUDY

Jun Xu
Feng Yang

Industrial and Management Systems Engineering Dept.
West Virginia University
Morgantown, WV 26506, U.S.A

Hong Wan

Industrial Engineering Department
Purdue University
West Lafayette, IN 47907-2023, U.S.A.

ABSTRACT

Computer simulation is an appealing approach for the reliability analysis of structure-based software systems as it can accommodate important complexities present in realistic systems. When the system is complicated, a screening experiment to quickly identify important factors (components) can significantly improve the efficiency of analysis. The challenge is to guarantee the correctness of the screening results with stochastic simulation responses. Control Sequential Bifurcation (CSB) is a new screening methods for simulation experiments. With appropriate hypothesis testing procedures, CSB can simultaneously control the Type I error and power. The past research, however, has focused on responses with normal distributions. This paper proposes a CSB procedure with binary responses for software reliability analysis. A conditional sequential test for equality of two proportions is implemented to guarantee the the desired error control.

1 INTRODUCTION

Increased research in structure-based software reliability analysis is needed with the advent of component-based software development paradigm. In the literature, both analytic models (such as discrete time Markov chain (DTMC), continuous time Markov chain (CTMC), and semi-Markov process (SMP)) and discrete-event simulation models have been developed to address the problem of quantifying software reliability. The former relies heavily on simplified assumptions of software systems to ensure analytical solutions, but is subject to the problem of intractably large state space; The latter offers an attractive alternative to analytical models since they are able to accommodate important complexities present in realistic systems. However, the current use of simulation for sensitivity analysis of software reliability 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. 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 we are interested in the effects of large number of factors potentially influencing the system's performance. In this research, a sequential ratio test-based factor-screening procedure was proposed for efficient sensitivity analysis of software reliability.

The structure of a software application may be defined as a collection of components comprising the application and the interactions among the components (Gokhale and Trivedi 2002). 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 functionalities. 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 a software application.

Once a simulation model is built, simulation experiments can be performed to estimate the software reliability under certain input factor settings. Input factors may include component reliability (or other parameters of individual components), intercomponent transition probabilities (user operational profile of the software), etc. The output response of a simulation run is represented by a binary random variable with two possible outcomes, success and failure of the software execution. To evaluate the impact of various factors upon the system reliability, experiments will be designed to carry out simulation at different factor settings. For complicated systems, however, the number of factors (i.e., the number of components) is large and the traditional design of experiments may require too many design points (i.e., too many simulation runs). In addition, usually there are only a few critical components, thus evenly spreading the computational efforts on each settings will be suboptimal. In these scenarios, a screening experiment should be conducted to eliminate the unimportant factors quickly, and therefore more effort can be saved for studying important factors. The experiment will be very useful in assessing the importance of each software component and creating a plan detailing what tasks will be performed to achieve the best 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. System parameters such as transition probabilities are estimated using the field data obtained during operational usage of the software, and they rarely can be estimated accurately. By varying the parameters subject to error, we can estimate the amount of uncertainty involved in the reliability estimates.

This paper proposes a screening framework for simulation study of the software reliability. We select Control Sequential Bifurcation (CSB) as the main algorithm (Wan et al. 2006, 2007). CSB extends the basic Sequential Bifurcation (SB) procedure (Bettonvil 1995; Bettonvil and Kleijnen 1997; Cheng 1997) to provide error control for random responses. Factors are grouped and the aggregated group effects are tested. If the group effects are classified as important, the group will be split into two smaller groups for further testing. If the group effects are classified as unimportant, on the other hand, all factors in the group will be classified as unimportant and no further testing will be needed. It is obvious that the method requires (1) main effects model is sufficient to model the data and (2) all factors' effects are with the same signs so no cancelation will happen. When only a small fraction of factors are important, CSB can eliminate unimportant factors in groups and hence ends up requiring significantly less computational efforts than traditional methods. By incorporating a multi-stage hypothesis-testing approach into sequential bifurcation, CSB is the first screening strategy to simultaneously control the Type I error for each factor and power for each bifurcation step under heterogeneous variance conditions. The methodology is easy to implement and is more efficient than traditional designs in many scenarios, which makes it attractive for many simulation applications (Wan et al. 2006, 2007).

We extend the current CSB procedure to handle binary responses. The previous hypothesis testing procedure developed for CSB with the specific error controls (which determine the error control property of the CSB procedure) are all for normal responses while the response of our current problem is binary (success (1) vs. failure(0)). A direct way to apply the current procedure is to define the response as the average of n simulated calling results, which is approximately normally distributed if n is sufficiently large. An obvious disadvantage of this approach, however, is that it takes n simulation runs to generate one observation for analysis, which could be computational intensive. In this paper, we propose to embed the conditional sequential tests proposed in Meeker (1981) in CSB to allow for the performance of factor screening for systems with binary output. The reason we choose this one over the well-known Wald's algorithm (Wald 1947) is because that Wald's tests only utilize the "untied" (i.e, one system fails and another does not) observations while Meeker's test utilizes all the observations. The superiority of Meeker's test over Wald's test has been discussed in Meeker (1981). We show that the CSB procedure with the Meeker's test is appropriate for detecting critical factors for software reliability problem.

2 RESPONSE MODEL

Suppose that there are K independent factors in the simulation experiment: $\mathbf{x} = (x_1, x_2, \ldots, x_K)$. The simulation output of interest Y is a binary random variable with distribution parameter p, i.e.,

$$Y = \begin{cases} 1, & \text{with probability } p \ ; \\ 0, & \text{with probability } 1-p \ . \end{cases} \tag{1}$$

We assume that the relationship between the success probability p and factors $\mathbf{x}$ can be represented by a logistic regression model:

$$\frac{p(\mathbf{x},\beta)}{1-p(\mathbf{x},\beta)} = \exp(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_K x_K) \tag{2}$$

where $\beta = (\beta_0, \beta_1, \ldots, \beta_K)$ are the unknown coefficients. It is assumed in model (2) that the dependence of p on $\mathbf{x}$ occurs through the linear combination $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_K x_K$, namely no interaction among the factors $\mathbf{x}$ are considered. General linear models such as (2) play an important role in both applied and theoretical work (McCullagh and Nelder 1989). Wan et al. (2005) pointed out 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. Our procedure is appropriate for both types of screening, but our presentation will focus on the latter application, that is, one form of sensitivity analysis.

The odds ratio $p(\mathbf{x},\beta)/(1-p(\mathbf{x},\beta))$ is a continuous and monotonically increasing function of the response probability p. For reasons that will be apparent in Section 4, we will treat the odds ratio as the primary response parameter of interest, and assume that the user is able to evaluate the practical importance of probability improvement in terms of odds ratio. Thus the impact of factor x_k is measured in the unit of $\exp(\beta_k)$: the effect of a unit change in x_k is to increase the odds ratio by an amount of $\exp(\beta_k)$.

Without loss of generality, it is assumed in our model that the level settings $\mathbf{x}$ are deterministic and coded either 0 or

1. The system/process is currently operating at a nominal level, say $\mathbf{x}_0 = \mathbf{0}_{K\times 1}$ (a K-dimensional zero-vector). We evaluate the effect of changing a factor by introducing a small disturbance to the nominal level, that is, by increasing the factor level from 0 (nominal level) to 1 (perturbation level). However, in practice the amount of disturbance introduced to each factor in its actual units should be determined in such a way that the effect of factor changes are comparable. Wan et al. (2006) proposed a cost model which determines the actual factor settings based on the required cost of changing a factor to produce an improvement in the output performance. We adapted their method for factor-level determination in our logistic model. Let c_k be the cost per unit change of factor k for $k = 1,2,\dots,K$, and $c^* = \max_{k=1,2,\dots,K} c_k$. Define the following thresholds of importance:

- t_0: the minimum change in the odds ratio for which we would be willing to spend c^*; and
- t_1: the change in the odds ratio that we would not want to miss if it could be achieved for only a cost of c^*.

The perturbation level for each factor can then be easily calculated (for details, see Wan et al. 2006). 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 the running of simulation. If, on the other hand, the experimenter already knows the thresholds of importance as well as the factor levels, then they do not need to use the cost model.

The objective of our procedure is to classify the factors considered into two groups: those that are unimportant, which we take to mean $\exp(\beta_k) \le t_0$, and those that are important, meaning $\exp(\beta_k) \ge t_1$. For factors with effects $\le t_0$, we control the Type I Error of declaring them important to be $\le \alpha$; and for factors with effects $\ge t_1$, we require the power for identifying them as important to be $\ge \gamma$. Those factors whose effects fall between t_0 and t_1 are considered important and we want to have reasonable, although not guaranteed, power to identify them.

3 CSB PROCEDURE REVIEW

In this section, we will review the CSB procedure proposed by Wan, Ankenman and Nelson (2006). Suppose that there are a total of K indexed factors and we use x_i to represents the setting of factor i in an experiment. In the screening experiment setting, we assume each factor has two levels, low level (coded as 0) and high level (coded as 1). An experiment at level k has the factor setting $\mathbf{x}(k) = (x_1(k), x_2(k), \dots, x_K(k))$ with $x_i(k)$ defined as

$$x_i(k) = \begin{cases} 1, & i = 1,2,\dots,k \\ 0, & i = k+1, k+2, \dots, K. \end{cases}$$

```
Initialization:  Create an empty LIFO queue for groups. Add the group $\{1,2,\dots,K\}$ 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 greater than 1, then split it into two subgroups such that all factors in the first subgroup have smaller index than those in the second subgroup. Add each subgroup to the LIFO queue.
    End Test
End While
```

Figure 1: Structure of CSB (Wan et al. 2006, 2007).

In other words, Experiment at level k is to set the factors $1,2,\dots,k$ at their high levels and the remaining at their low levels. Further define n_k as the number of observations at level k.

CSB is a series of steps. At each step, the cumulated group effect is tested. If the group effects are classified as unimportant, then all factors within the group will be classified as unimportant. No further tests are needed. If the group effects are classified as important, the group will be split to smaller groups for further testing. For cases with large number of factors and only a small percentage of them are important, CSB can eliminate many of the unimportant factors in groups, therefore saves computational efforts required for screening. On the other hand, CSB procedure requires known directions of factors' effects to avoid cancelation. Complicated interaction structures may also give misleading results (Wan et al. 2006).

An overview description of CSB is shown in Figure 1. The figure illustrates how groups are created, manipulated, tested and classified, but does not specify how data are generated or what tests are performed (which is discussed in the next section).

Suppose the group removed from the queue contains factors $\{k_1+1, k_1+2, \dots, k_2\}$ with $k_1 < k_2$. The **Test** step in Figure 1 tests the following hypothesis to determine if a group might contain important factors:

$$H_0 : e^{(\sum_{i=k_1+1}^{k_2} \beta_i)} \le t_0 \ \ vs. \ \ H_1 : e^{\left(\sum_{i=k_1+1}^{k_2} \beta_i\right)} \ge t_1.$$

Incorporating appropriate hypothesis testing procedure, CSB is the first screening method that provides simultaneous control of Type I error and power. Many research has been done to improve the original CSB procedure, which we do not discuss in this paper. Interested readers are referred to Wan et al. (2007), Sanchez et al. (2006) and Shen and Wan (2005, 2006).

The hypothesis testing procedure given in Section 4 guarantees that the probability of Type I error is $\leq \alpha$ when $\exp\left(\sum_{i=k_1+1}^{k_2} \beta_i\right) \leq t_0$, and the power is $\geq \gamma$ if $\exp\left(\sum_{i=k_1+1}^{k_2} \beta_i\right) \geq t_1$.

4 HYPOTHESIS TESTING PROCEDURE

Previous research on appropriate hypothesis tests for CSB has been focused on responses with normal distributions. For the binary responses, we adopt the conditional sequential tests proposed by Meeker (1981) which gives the desired error control (i.e., a qualified test).

Following Meeker's notations, we define the relative superiority measure for two different factor settings $\mathbf{x}(k_1)$ and $\mathbf{x}(k_2)$ as follows:

$$t = \frac{p(\mathbf{x}(k_2)/(1-p(\mathbf{x}(k_2)))}{p(\mathbf{x}(k_1)/(1-p(\mathbf{x}(k_1)))} \tag{3}$$

The hypothesis test to determine if group (k_1, k_2) might contain important factors is given as

$$H_0 : t \leq t_0 \text{ vs. } H_1 : t \geq t_1. \tag{4}$$

If $t = 1$, the two factor settings are equally good, and if $t > 1$, $\mathbf{x}(k_2)$ is superior to $\mathbf{x}(k_1)$. Two user-specified values of t, t_0 and t_1 $(1 < t_0 < t_1)$, need to be selected such that the rejection of H_0 is considered an error of practical importance whenever the true value of t is less than or equal to t_0 , and the acceptance of H_0 is considered an error of practical importance whenever the true value of t is greater than or equal to t_1. Type I error α is the probability of rejecting H_0 when $t \leq t_0$, and type II error $1-\gamma$ is the probability of accepting H_0 when $t \geq t_1$.

The required procedure is demonstrated in Figure 2 and necessary notation is given below:

- α : Required Type I Error
- γ: Required power
- $a = \ln\{(\gamma)/\alpha\}$
- $b = \ln\{(1-\gamma)/(1-\alpha)\}$
- $y_m(k)$: the mth observation at level k. $y_m(k) = 1$ (success) or 0 (failure)
- n : number of current observations used in the hypothesis test (for both level k_1 and k_2, n can be smaller than n_{k_1} or n_{k_2})

```
Test Initialization    $n = 1, x_1 = 0, x_2 = 0, r = 0$, Finish = 0
While $finish < 1$ AND $n < n_0$, do
    If $n_{k_1} < n$:    Take one observation at level $k_1, n_{k_1} = n$
    Endif
    $x_1 = x_1 + y_n(k_1)$
    If $n_{k_2} < n$:    Take one observation at level $k_2, n_{k_2} = n$
    Endif
        $x_2 = x_2 + y_n(k_2)$
        $r = x_1 + x_2$
    Unimportant:    If $x_2 < c_L(r,n)$, classify the group as unimportant, Finish =1.
    Important:    ElseIf $x_2 > c_U(r,n)$ then classify the group as important. Finish=1.
    Endif
    $n = n + 1$;
End While
If $n = n_0$
    Unimportant:    If $x_2 \leq c_L(r,n_0)$, classify the group as unimportant.
    Important:    ElseIf $x_2 \geq c_L(r,n_0) + 1$ then classify the group as important.
Endif
```

Figure 2: Fully sequential test.

- n_0 : maximum number of observations allowed for the hypothesis test (for level k_1 and k_2 individually)
- $x_2 = \sum_{i=1}^{n} y_i(k_2)$, the number of success at level k_2
- $x_1 = \sum_{i=1}^{n} y_i(k_1)$ the number of success at level k_1
- $r = x_1 + x_2$
- $c_L(r,n) = \lfloor \{b + F(t_1) - F(t_0)\}/ln(t_1/t_0) \rfloor$
- $c_L(r,n) = \lfloor \{a + F(t_1) - F(t_0)\}/ln(t_1/t_0) \rfloor + 1$
- $F(t) = F(r,n,t)$ is a function of r, n, and t:

$$F(r,n,t) = \ln\left(\sum_{j=l}^{t} \left(\begin{array}{c} n \\ j \end{array}\right) \left(\begin{array}{c} n \\ r-j \end{array}\right) t^j\right)$$

- $c_L(r,n_0) = \lfloor \{\nu + F(t_1) - F(t_0)/ln(t_1/t_0)\} \rfloor$
- $\nu = (a+b)/2$
- $c_U(r,n_0) = c_L(r,n_0) + 1$

For the discussion of the performance guarantee and implementation issues, please see Meeker (1981). It should be noticed that even $c_L(r,n_0)$ and $c_U(r,n_0)$ can be defined for small n_0, setting n_0 too small can lead to deteriorate results.

5 PERFORMANCE OF CSB WITH THE CONDITIONAL SEQUENTIAL TEST

The above hypothesis test can be proved with the following error control properties (Meeker 1981):

$$\Pr\left\{\text{Declare group effect important}\,\middle|\, e^{\left(\sum_{i=k_1+1}^{k_2}\beta_i\right)} \le t_0\right\} \le \alpha,$$

and

$$\Pr\left\{\text{Declare group effect important}\,\middle|\, e^{\left(\sum_{i=k_1+1}^{k_2}\beta_i\right)} \ge t_1\right\} \ge \gamma.$$

Given the specific hypothesis testing procedure, we can prove the following two theorems for CSB (Wan et al. 2006):

Theorem 1 *CSB guarantees that* $\Pr\{$*Declare factor i important*$\}|\beta_i \le \Delta_0\} \le \alpha$ *for each factor i individually.*

Theorem 2 *CSB guarantees that* $\Pr\{$*Declare a group effect important*$\}|\exp(\beta_i) \ge t_1\} \ge \gamma$ *for each group tested.*

6 EMPIRICAL EVALUATION

In this section, we evaluate the performance of the factor-screening procedure proposed in Section 3. Before employ system simulation models to test the effectiveness of the factor screening procedure, we chose to generate data from a logistic regression model in which we are able to fully control the size of the effects by setting the coefficients of the factors being considered.

Specifically, binary random response Y are generated from the Bernoulli distribution with parameter $p(\mathbf{x},\beta)$, and $p(\mathbf{x},\beta)$ is a function of the settings of K factors:

$$\frac{p(\mathbf{x},\beta)}{1-p(\mathbf{x},\beta)} = \exp(\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_Kx_K) \quad (5)$$

Suppose that the level settings $\mathbf{x}$ are deterministic and coded either 0 or 1. The factors are evaluated in terms of their effects on the odds ratio. The hypothesis test performed to judge that whether or not factor k is important is as follows:

$$H_0: t=\exp(\beta_k)\le t_0 \text{ vs. } H_1: t=\exp(\beta_{k_2})\ge t_1. \quad (6)$$

The parameters for the experiments are set as in Table 1, and we consider two different settings for factor effects which is discussed next.

Table 1: Parameters for experiments based on logistic regression models.

Parameter	Value
K	10
t_0	1.15
t_1	1.30
α	0.05
γ	0.05

Table 2: 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	0.10	$\exp(\beta_3)$	1.11
β_4	0.15	$\exp(\beta_4)$	1.16
β_5	0.20	$\exp(\beta_5)$	1.22
β_6	0.25	$\exp(\beta_6)$	1.28
β_7	0.30	$\exp(\beta_7)$	1.35
β_8	0.35	$\exp(\beta_8)$	1.42
β_9	0.40	$\exp(\beta_9)$	1.49
β_{10}	0.45	$\exp(\beta_{10})$	1.57

Case 1

The effect of each factor i depends on its coefficient β_i., and we set the coefficients β as in Table (2). According to (6), the factors with $\exp(\beta_i) < t_0$ are considered as unimportant, and the factors with $\exp(\beta_i) > t_1$ are considered as critical. In this case, as can be seen from Table (2), factors 1, 2 and 3 are unimportant, and the observed frequency that either one of them is declared important should be smaller than the prespecified type I error, 0.05; Factors 7, 8, 9 and 10 are critical, and the observed frequency of declaring them as important should be near 0.95. Define $P(DI)$ as the "probability of being declared important". Figure 3 plots $P(DI)$ against effect size $\exp(\beta_i)$ based on 1000 replications with each replication using a different random stream. It is shown in Figure 3 that the desired type I error and power level have been achieved in this experiments.

Case 2

We set $(\beta_0,\beta_1,\ldots,\beta_{10}) = (0.1,0.1,\ldots,0.1)_{1\times 10}$, so that $\exp(\beta_i) < t_0$ for all the 10 factors. This set is designed to study the control of Type I error for the proposed procedure. The other parameters are the same as in the previous case. Out of 1000 replications, the frequency of being declared as important is below 1% for all factors.

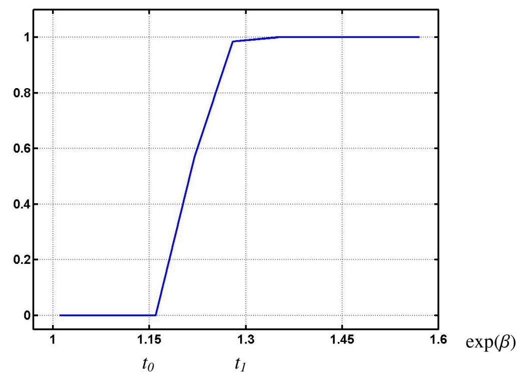

Figure 3: Case 1.

7 CASE STUDY

In this section, we demonstrate the potential of using simulation-based factor screening procedure to identify important factors that have a large impact on software reliability. We use the application reported by Cheung (1981) as an example, which has been used extensively to illustrate structure-based reliability assessment techniques (Gokhale and Trivedi 2002, Goseva-Popstojanova and Kamavaram 2003, and Lo et al. 2002).

7.1 Software Architecture

The structure of the application (Cheung 1981) is shown in Figure 4. 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) as shown in Figure 4. 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 $(i, j = 1, 2, \ldots, 10)$. The non-zero transition probabilities among the components are given in Table 3. 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, the software system consists of 10 components, and the execution of the application always starts with component 1 and ends with component 10. Furthermore, we assume that the application spends 1 time unit at each component per visit. Hence, the software execution process illustrated in Figure 4 can be modeled as discrete time Markov chain (DTMC) with transition probability matrix $P = (p_{ij})_{10\times 10}$.

7.2 Failure Behavior

We first consider the failure behavior of components, i.e., the reliability of each component. A component can fail during its execution period, which is assumed to be 1

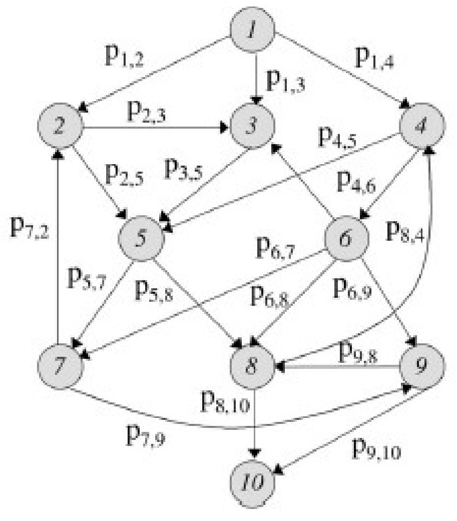

Figure 4: Structure of an example application.

Table 3: Intercomponent transition probabilities for the software example.

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

unit of time. We define the reliability of a component 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 is a DTMC, we can analytically derive the expression for system reliability as a function of transition probabilities and component reliabilities. In our experiments, we assume that the transition probabilities are fixed known values, and try to evaluate the impact of component reliabilites upon the system reliability. Let R_i denote the reliability of component i $(i = 1, 2, \ldots, 10)$, and R_s the reliability of the software system. The functional relationship between R_s and $\{R_i, i = 1, 2, \ldots, 10\}$,

$$R_s = f(R_1, R_2, \ldots, R_{10}), \tag{7}$$

can be derived based on the DTMC, and hence the effect of improving R_i on R_s can be evaluated. In the remainder of this section, we will explain how the proposed CSB procedure is applied on the software model, and compare the results obtained from simulation experiments with the analytical ones obtained from DTMC.

7.3 Application of CSB on the Software Model

Our objective 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 being considered are $\mathbf{x} = (x_1, x_2, \ldots, x_{10}) = (R_1, R_2, \ldots, R_{10})$. For the consistency of notation, we will continue to use $\mathbf{x}$ to represent the factor setting vector. The output of a simulation run of the software model is a Bernoulli random variable Y with distribution parameter $p = R_s$. The hypothesis tests (4) performed in the CSB procedure is to compare $p(\mathbf{x}(k_1))$ and $p(\mathbf{x}(k_2))$.

We define a sample path as the sequence of components visited by a application execution. In other words, a sample path represents a calling sequence of the components when the software is being executed by users. Let $S = \{S_1, S_2, S_3, \ldots\}$ denote the set of all the possible sample paths, and we have

$$p(\mathbf{x}) = \mathrm{E}_S[p(\mathbf{x}, S)] \tag{8}$$

where $p(\mathbf{x}, S)$ represents the system reliability following sample path S, and E_S denotes expectation with respect to S. Further, we denote $Y(\mathbf{x}, S)$ as the random output at factor setting $\mathbf{x}$ for given sample path S, and $Y(\mathbf{x}, S)$ follows Bernoulli distribution with parameter $p(\mathbf{x}, S)$.

In light of the fact that reliability $p(\mathbf{x}, S)$ could differ markedly depending on the particular sample path S, in the sequential simulation, we set up the experiments in such a way that simulation replications at factor settings $\mathbf{x}(k_1)$ and $\mathbf{x}(k_2)$ are performed at the same randomly selected sample paths. Specifically, we store all the sample paths, say $S_1, S_2, \ldots, S_n$, independently generated so far in the procedure. When comparing two factor settings $\mathbf{x}(k_1)$ and $\mathbf{x}(k_2)$, random outputs $Y(\mathbf{x}(k_1), S_i)$ and $Y(\mathbf{x}(k_2), S_i)$ 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 $Y(\mathbf{x}(k_1), S_{n+1})$ and $Y(\mathbf{x}(k_2), S_{n+1})$ will be collected.

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

$$p(\mathbf{x}, S) = \prod_{i=1}^{10} x_i^{c_i} \tag{9}$$

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

In our experiments, nominal factor settings are given in Table 4, and the factor increment is set as 0.013. The CSB parameters for the experiments are given in Table 5, and the CSB procedure with conditional sequential test (Meeker's test) is applied on the software system represented in Figure 4. The procedure identifies components 5, 8 and 10 as important components, which is consistent with the analytical results obtained from DTMC analysis.

Table 4: Nominal factor settings for component reliabilities.

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

Table 5: Parameters for software reliability experiments.

Parameter	Value
K	10
t_0	1.08
t_1	1.11
α	0.01
γ	0.01

8 FUTURE RESEARCH

We will further test the methods with different cases. The future research will be focused on the following two areas. First, for highly reliable softwares systems, failures can be considered as rare events. Evaluating the impact of various factors on the probability of the occurrence of rare events is challenging. We plan to incorporate importance sampling or other rare event simulation techniques with the factor screening procedure to stress these cases. Second, current CSB requires strong assumptions which are not necessarily fulfilled in practice. In this study, we do not consider the transition probabilities as factors. If we do, then the directions of these factors' effects will be hard to predict beforehand. Also, the interactions between the components in many scenarios cannot be ignored, especially for large software with many components and complex calling sequences. We will seek to improve the existing versions of CSB work (Wan et al. 2007) as well as other screening strategies to relax these assumptions.

ACKNOWLEDGMENTS

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

REFERENCES

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.

Goseva-Popstojanova K., and S. Kamavaram. 2003. Assessing uncertainty in reliability of component-based software systems. In *Proceedings of the 14th International Symposium on Software Reliability Engineering*, 307–320.

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

Meeker, W. Q. 1981. A Conditional Sequential Test for the Equality of Two Binomial Proportions. *Applied Statistics* 30 (2): 109–115.

McCullagh, P., and J. A. Nelder. 1989. Generalized linear models. 2 Ed. New York: Chapman and Hall.

Sanchez, S. M., H. Wan and T. W. Lucas. 2005. A two-phase screening procedure for simulation experiments. In *Proceeding of the 2005 Winter Simulation Conference*, ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers. 223–230

Shen, H. and H.Wan. 2005. Controlled Sequential Factorial Design for Simulation Factor Screening. In *Proceeding of the 2005 Winter Simulation Conference*, ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers. 467–474

Shen, H. and H. Wan. 2006. A hybrid method for simulation factor screening. In *Proceeding of the 2006 Winter Simulation Conference*, ed. L. F. Perrone, B. Lawson, J. Liu and F. P. Wieland, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers. 376–381

Wald, A. 1947. Sequential Analysis. New York: John Wiley.

Wan, H., B. E. Ankenman, and B. L. Nelson. 2006. Controlled Sequential Bifurcation: A New Factor-Screening Method for Discrete-Event Simulation. *Operations Research* 54 (4): 743–755.

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

AUTHOR BIOGRAPHIES

JUN XU is a PhD student in the Industrial and Management Systems Engineering Department at West Virginia University. His research work has been focused on discrete event simulation. His e-mail address is <jxu2@mix.wvu.edu>.

FENG YANG is an assistant professor in the Industrial and Management Systems Engineering Department at West Virginia University. Her research interests include simulation and metamodeling, design of experiments, and applied statistics. Her e-mail and web addresses are <feng.yang@mail.wvu.edu> and <http://www2.cemr.wvu.edu/~yang/>.

HONG WAN is an Assistant Professor in the School of Industrial Engineering at Purdue University. Her research interests include design and analysis of simulation experiments, simulation optimization; simulation of manufacturing, healthcare and financial systems; quality control and applied statistics. She has taught a variety of courses and is a member of INFORMS and ASA. Her e-mail and web addresses are <hwan@purdue.edu> and <http://web.ics.purdue.edu/~hwan>.