Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 20(2): 79-89, Jun. 2022 Regular paper

Component-Based System Reliability using MCMC Simulation

Sampa ChauPattnaik[1]*, Mitrabinda Ray[1], Mitalimadhusmita Nayak[2], and Srikanta Patnaik[1]

[1]Department of Computer Science and Engineering, Siksha 'O' Anusandhan (Deemed to be) University, Bhubaneswar 751030, India
[2]Department of Mathematics, Siksha 'O' Anusandhan (Deemed to be) University, Bhubaneswar 751030, India

Abstract

To compute the mean and variance of component-based reliability software, we focused on path-based reliability analysis. System reliability depends on the transition probabilities of components within a system and reliability of the individual components as basic input parameters. The uncertainty in these parameters is estimated from the test data of the corresponding components and arises from the software architecture, failure behaviors, software growth models etc. Typically, researchers perform Monte Carlo simulations to study uncertainty. Thus, we considered a Markov chain Monte Carlo (MCMC) simulation to calculate uncertainty, as it generates random samples through sequential methods. The MCMC approach determines the input parameters from the probability distribution, and then calculates the average approximate expectations for a reliability estimation. The comparison of different techniques for uncertainty analysis helps in selecting the most suitable technique based on data requirements and reliability measures related to the number of components.

Index Terms: Component Based Systems, MCMC, System Reliability, Discrete Time Markov Chain

I. INTRODUCTION

Software reliability is necessary for system developers to improve application performance. Delivering commercial-off-the-shelf (COTS) products is a critical challenge for all industries. Software reliability analysis at the architectural level is based on the following models: state-based [1-5], path-based [21,29], and additive [32].

Software reliability estimation occurs at the code and design levels. In architecture-based software reliability analysis, the input parameters include the software architecture of the application, transition probabilities between components, reliability of individual components, and average execution time of the components [6]. During the design phase, the major challenge is that the software architecture and component parameters may not be well known. Thus, an uncertainty analysis is required for the architecture-based software reliability model. The goal is to investigate how the inaccuracy of the operational profile [7] directly influences the transition probabilities between the components. The failure of the components is unknown and depends on the test procedures. Because the failure time data are uncertain, the system reliability computed from the parameters is also uncertain. The reliability growth of a component depends on its fault detection rate. A fault detection rate may be given by one of the software reliability growth models [8-11], and the location of faults that caused these failures can then be identified based on several accurate methods.

Motivation: Software reliability prediction based on architectural models depends on i) appropriate model considerations and ii) accurate parameter values. Practically, there is considerable uncertainty in parameters because uncertainty occurs in the estimation of the operational profile and component reliabilities. A major challenge in reliability predic-

Received 28 January 2021, **Revised** 23 April 2022, **Accepted** 10 May 2022
***Corresponding Author** Sampa ChauPattnaik (E-mail: sampa.chaupattnaik@gmail.com, Tel: +91-88-9565-6648)
Department of Computer Science and Engineering, Siksha 'O' Anusandhan (Deemed to be) University, Bhubaneswar 751030, India.

Open Access https://doi.org/10.6109/jicce.2022.20.2.79 print ISSN: 2234-8255 online ISSN: 2234-8883

tion at the architectural level is the presence of uncertainties owing to the lack of information in relation to the proposed artifact, such as operational profile, implementation details, and failure behavior.

The objectives of this paper are:

- To determine the uncertainty parameters that arise through the model such they are properly represented in the reliability analysis.
- To monitor the uncertainty results such that they help decision-makers make further decisions.
- To analyze the system reliability by considering the effect of the individual reliability of a component using some uncertain manner.

We present uncertainty in architecture-based software reliability based on a Markov chain Monte Carlo (MCMC) simulation and moment techniques. The methodology addresses the parameter uncertainty problem and shows its effect on system reliability.

The remainder of this paper is organized as follows. In Section II, we summarize the background study. This explains the different approaches to uncertainty analysis for component-based systems. Section III describes the uncertainty analysis technique based on the method of moments, Monte Carlo (MC) simulation, and Markov chain Monte Carlo (MCMC) simulation. Section IV explains the result analysis using different uncertainty methods in architecture-based software reliability analysis. Section V provides a comparative study of different methods for uncertainty analysis, and the paper concludes in Section VI.

II. BACKGROUNG STUDY

In the last decade, several architecture-based software reliability models have been proposed [8, 12, 13, 18, 35]. In the early phase, software reliability analysis can identify the critical components that improve the reliability of an application. Software reliability estimation in the design phase causes uncertainty in the component reliabilities. Several researchers [14-24] have used different techniques to analyze uncertainty in architecture-based software systems. To evaluate the overall system reliability, we used different models, such as Markov chains and fault trees. Parameters such as component reliabilities, transition probabilities between two components, and average execution time of components are used in these models, generally collected from field data, systems with similar function data, or by guessing. The exact values of the parameters may be unavailable at the design stage (as coding has not begun). Hence, the analyst can formulate a relative estimation for the values of the parameters by considering the estimation from the actual system implementation to validate the assumption prepared in the design stage. Therefore, the values of these parameters may be inaccurate. This leads to uncertainties in system reliability.

Reliability assessment occurs during the early stages of the software development life cycle using software architecture. However, the prediction of component reliabilities at the architectural level is difficult because uncertainties arise in the system and its individual components [15]. authors considered a component-based robotics test bed, the controller component of SCR over, which is a NASA JPL mission data system (MDS). They identified different uncertainty factors of components, such as the recovery and failure probabilities. Their results showed that the reliability of the component increased with the recovery probability. In addition, if the failure probability increases, the component reliability decreases.

For architecture-based reliability estimation, accuracy may depend on parameters such as environmental factors and system usage. Researchers have handled this problem by integrating uncertainties in architecture assessment models and analytically solving them with simulations. This method illustrates the mean and variation of an attribute. The case study used was on the anti-lock braking system (ABS) of a car [25]. Architecture-based software reliability analysis can identify the critical components early, which leads to a cost-effective reliability improvement of the application. However, at this phase, estimating architectural and component parameters is not entirely possible with certainty.

The problem of uncertain component reliabilities has occurred in software application reliability estimation [26]. The authors considered the levels of confidence in architectural parameters using minimal test and simulation data based on the principle of multinomial distribution confidence intervals for reliability estimation. K. Go et. al. [27] conducted a large-scale case study on architecture–based software reliability evaluation to determine potential limitations and challenges for future analysis. The study of uncertainty in architecture–based software reliability found that the MCapproach is more effective than the moments technique. The MCtechnique results indicate the following: (1) fewer parameters yield the most variance in system reliability and (2) the reliability of components has a greater effect on system reliability than the probability of transition for a given operational profile.

Software reliability models consist of several parameters that are typically determined from the test data of the component. As point estimates concentrate on random data variations, uncertainties in the measured parameters were observed. The parameter causing uncertainty underestimates the uncertainty in terms of overall system reliability [16]. Here, the authors aimed to measure the uncertainties with their correlated parameters in the software reliability modeling of a single component and multiple components. A major problem in the uncertainty analysis of software reliability modelling is the lack of available failure data. Using the views of

experts and historical data from earlier projects will help resolve this problem. An uncertainty analysis was performed by merging the concept of maximum entropy (MEP) with the Bayesian method. System-level uncertainty analysis can be conducted using the MCmethod to demonstrate the impact of uncertainty parameters on many system-level components.

In component-based software development systems, an inadequate selection of components results in low-quality software. The authors disagree with the assessment methods in which explicit uncertainty methods are represented using probability distributions. They provide details of the Bayesian model used to capture the uncertainties in the simultaneous assessment of two attributes, thereby confining the dependencies between them [28]. To handle the uncertainty in the system reliability, a MCsimulation method is used [29], where reliability is determined by the variance, and the accuracy increases by increasing the sample size. To estimate the reliability of a system, the method of moments is used for uncertainty analysis in architecture-based software reliability models [30]. The method of moments shows that the reliability evaluation occurs by the moments of component reliabilities, where the accuracy can be improved by considering the high level of Taylor expansion. The software consists of components and modules.

Software reliability assessment is the primary area of software-based systems. Software reliability estimation is based on historical data and a presumed distribution curve, which is inclined to uncertainties. A fault tree [31] is used for reliability estimation in component-based systems; and fuzzy set theory is used in fault tree analysis to quantify the basic uncertainty results. In recent years, for complex architecture systems, such as flights, a highly reliable system is required for safety purposes [32]. This leads to uncertainty in failure occurrences. The confidence interval and probability density functions were used to compute the uncertainty in the system reliability variance. The evaluations show that the uncertainty varies with the system or execution time.

III. UNCERTAINTY ANALYSIS

The behavior of the system is affected by unavoidable components, which are considered uncertain. Uncertainty is categorized into two types: i) aleatory uncertainty: stochastic process behavior (i.e., randomness from inherent variability in the system) and ii) epistemic uncertainty: vagueness owing to a lack of knowledge and information about the system [33]. The causes and aspects of uncertainty have been studied in various applications such as engineering risk modeling and reliability assessments. Although many different types of uncertainty exist, they are usually categorized as aleatory or epistemic. Aleatory uncertainty, also known as stochastic uncertainty, arises owing to the randomness of the behavior of a software system. The probability distribution is commonly used to solve the aleatory uncertainty problem with sufficient data. For example, an application using a reliability-based design is based on probability theory. Epistemic uncertainty occurs because of a lack of knowledge or information regarding a software system, particularly during the design of a complex system. It is not based on probabilistic theory.

Methods such as interval analysis, possibility theory, and evidence theory have been used to model epistemic uncertainty. This paper is based on epistemic uncertainty because we apply uncertainty analysis during the design phase and sufficient data are unavailable for estimating system reliability. Epistemic uncertainty analysis is based on probability theory as well as fuzzy sets. This paper contains probability distributions, such as beta and gamma distributions.

To conduct reliability estimation with uncertainty analysis in a system, the following steps are followed.

1. Build the software architecture obtained from a specification and design document.
2. Estimate the component reliability
3. Design the software model (using the software growth model or fault injection)
4. Use the uncertainty techniques for the uncertainty parameters (component reliability, transition probability, etc.). I.e., in this step, different uncertainty methods (method of moments, MC simulation, or MCMC simulation) are applied.
5. Obtain the system reliability.

Software reliability is estimated using an architecture-based model, and we include the dynamic behavior of the software architecture, i.e., a component dependency graph (CDG) [6], transition probabilities of two components, and nature of software failure (reliability of components or failure rate). For a given application software, the uncertainty in software reliability is caused by these parameters. Software architecture is defined as the interaction of different components. We used a path-based model to represent the architecture–based software reliability model [8, 15, 16, 26]. The path-based method uses the CDG. The components are represented as nodes, and the interactions between the components are represented as edges. Transfers between components are assumed to be a discrete-time Markov chain (DTMC) with transition probability $Pr = [pr_{ij}]$, where pr_{ij} is the transmission from component i to component j. Component failure behavior estimates the reliability of each component. (CR_i).

Several methods have been used to estimate component reliability. To develop component failure data, software reliability growth models (SRGM) [8, 15, 16, 26, 34] were applied to each component obtained during testing. Software reliability growth models cannot always be used because of the scarcity of failure data. Another manner of estimating the reliability of a component is to use a fault injection tech-

nique [13], which isa simulation basis. Therefore, the estimation of component reliability estimated may be inaccurate, which stimulates the use of uncertainty analysis. The methodology used for uncertainty analysis was obtained using a software architecture with the failure behavior of the components.

In this study, we used a path-based model [35] for reliability estimation. For the reliability assessment, we considered the software architecture with failure behavior. The reliability of a system is the probability of reaching the terminal component through the execution path along the components. The overall system reliability is calculated as follows:

$$SR = m(1, n)CR_n, \quad (1)$$

where, $m = \sum_{k=0}^{\infty} q^k = (ID - q)^{-1}$. This denotes the $(1,n)^{th}$ value of matrix m, i.e., reaching the end component n by starting from component 1. In the DTMC method, the transition probability is represented by $Pr = \begin{bmatrix} q & c \\ 0 & 1 \end{bmatrix}$, where q = (r − s) × (r − s) is a sub-stochastic matrix (minimum one-row summation <1), s × s is the identity matrix $ID = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, 0 = s × (r − s) is the matrix of 0s, and c = (r − s) × s. In addition, a Markov process with r states has s absorbing states.

Several techniques are used for uncertainty analysis. Here, we present some techniques, such as the method of moments [30], MC simulation [29], and MCMC simulation method [36].

A. Moments

This is an approximate approach that generates moments of system reliability from the moments of component reliabilities. The component reliability estimation using this method is inaccurate because of the higher-order Taylor series expansion.

For method of moments:

a) The link between system reliability (SR) and component reliabilities (CR_1, CR_2, ..., CR_n) is represented by SR= f(CR_1, CR_2, ..., CR_n) using the path-based model.
b) The system reliability is determined using Taylor series.
c) The moments are applied for the component reliabilities and transition probabilities.
d) The approximate mean and variance of the system reliability are calculated using the parameter moments using the given equations.

The method of moment is expressed as a Taylor series function. Thus, the 1st order moment is given by the equation

$$\mathbf{SR} \approx \mathbf{r_0} + \sum_{\mathbf{i=0}}^{\mathbf{n}} \mathbf{r_i}\left(\mathbf{CR_i} - \mathbf{E[CR_i]}\right) + \sum_{\mathbf{i=1}}^{\mathbf{n}} \sum_{\mathbf{j=1}}^{\mathbf{n}} \mathbf{rp_{ij}}(\mathbf{Pr_{ij}} - \mathbf{E[Pr_{ij}]}), \quad (2)$$

where, SR denotes the reliability of the components, and E[CR] is the expected reliability of components.

$$r_0 = f(E[CR_1], E[CR_2], E[CR_3], \ldots E[CR_n], \quad (3)$$

$$r_i = \left.\frac{\partial CR}{\partial CR_i}\right|_{CR_i = E[CR_i]}, \text{ for } i = 1, 2, \ldots n, \text{ and} \quad (4)$$

$$rp_{ij} = \left.\frac{\partial CR}{\partial Pr_{i,j}}\right|_{CR_i = E[CR_i], Pr_{i,j} = E[Pr_{i,j}]} \text{ for, } j = 1, 2, \ldots .n$$

The mean of the system reliability is $E[CR] \approx r_0$ (5)

Then, the variance of the system reliability is as follows.

$$V[CR] = \sigma_2 \approx \sum_{i=1}^{n} r_i^2 V[CR_i] + \sum_{k=1}^{n} \sum_{i=1}^{n} rp_{ij}^2 V[Pr_{ij}] + 2 \sum_{k=1}^{n} \sum_{i=1}^{n} \sum_{j=i+1}^{n} rp_{ki} rp_{kj} Cov(Pr_{ki}, Pr_{kj}) \quad (6)$$

The variance estimation shows the confidence in system reliability. Thus, if the variance decreases, the overall system reliability increases. Now, the efficiency of the expected reliability and variance of reliability is improved using high-order Taylor expansions.

B. Monte Carlo Technique

MCsimulation is a fairly accurate method for estimating system reliability. This method represents a well-defined distribution of the parameters of the software reliability model. The MCtechnique for a component-based system is as follows.

Algorithm: Uncertainty Analysis using MC for a Component Based System

Input: component reliability of the i^{th} component (CR (i)), transition probability between two components (Pr_{ij}), average execution time of n components; n: number of components

Output: System Reliability

1. Initialize $CR_i = X[i]$.
   X[i]= random variables, X ∈ [0,1]
2. Identify the system reliability using a path-based model (PR) with input parameters CR(i) and Pr_{ij}:

$$PR = CR_1 \times \prod_{i=2}^{n-1} CR_i^{Pr_{1,i}} \times CR_n^{\sum_{j=2}^{n-1} Pr_{1,j} \times Pr_{j,n}}$$

3. Calculate the mean for each component (CR_i) using the formula $m = (ID - q)^{-1} = \sum_{u=0}^{\infty} q^u$, where m is the mean number of visits from component I to component j, ID is the identity matrix, and q is the transition matrix explained below.
4. Calculate the variance of each component CR_i using the formula $v = m(2m_{dig} - ID) - m_{sqr}$, where v is the variance of each component, $\mathbf{m_{dig}}$ is the diagonal of matrix m, and $\mathbf{m_{sqr}}$ is the square of the matrix m.

5. Determine the system reliability:

$$SR = \left[\prod_{j=1}^{n-1} CR_j^{m_{1,j}} + \frac{1}{2}(CR_j^{m_{1,j}})(\log CR_j)^2 v_{1,j} \right] CR_n,$$

where, $m_{1,j}$ and $v_{1,j}$ are the visit mean and variance, respectively, of each component from component 1 to component j.

6. Assign probability distributions to Pr_{ij} and RC_i. (The probability distributions of the input parameters are assigned based on theoretical assumptions. Here, we use the beta and gamma distributions for the component reliability and transition probability, respectively.)

$$f(CR_i) = \frac{\Gamma(x_i + y_i)}{\Gamma(x_i) + \Gamma(y_i)} CR_i^{x_i - 1}(1 - CR_i)^{y_i - 1}$$

$$f(pr_{i1}, pr_{i2}, pr_{i3}, \ldots pr_{in} =$$

$$\frac{\Gamma(A_{i1} + A_{i2} + \ldots + A_{in}}{\Gamma(A_{i1}) + \Gamma(A_{i2}) + \ldots \Gamma \ldots_{in})} \prod_{j=1}^{n} pr_{ij}^{A_{ij}-1}$$

7. Repeat Steps 3, 5, and 5 with the new values of CR (i) and Pr_{ij} obtained from the probability distribution.
8. Repeat Steps3, 4, 5, and 6 until the required number of values is produced to obtain a lower variance and better system reliability.

MCsimulation process

** i= no. of column and j= no. of rows

MCsimulation is a method for determining the probability of possible outcomes. Predict a model is difficult owing to the use of random variables. This is also referred to as a multiple probability simulation. MCMC is subset of MC. Thus, MCMC is an MC, but not all MCs are MCMC. The main objective of the MCMC method is to build a Markov chain. The Markov chain is produced using independent samples (with a negative correlation) to reduce the variance.

C. Markov Chain Monte Carlo

MCMC is an approximation technique for system reliability estimation. Therefore, the objective of this simulation was to use Bayesian inference [37] for posterior distributions. MCMC allows the user to approximate aspects of the posterior distributions. The MCMC method produces MCsimulations depending on the Markov property, and the simulation convergence point obtained is the posterior distribution [36]. The MCMC process is stochastic. A stochastic process {s(T)} has a Markov property (Mp). The Markov property is represented as Pr{s(T + 1) = n|s(0) = 0, s(1) = …s(T) = m) = Pr{s(T + 1) = n|s(T) = m} for T = 0, 1, 2, …. This expression show that state s at time T + 1 only depends on the current state s at time T, and is not dependent on past states s(T−1), …, T(1).

Algorithm: Component Based System Uncertainty Analysis using MCMC

Input: component reliability of the i^{th} component (CR (i)), transition probability between two components ($Pr_{i,j}$), average execution time of n components

Output: variance of each component (C_i), System Reliability

1. Initialize CR_i =X[i]
   X[i]= random variables, X ∈ [0,1]
2. Identify the system reliability using the path-based model (PR) with input parameters CR(i) and pr_{ij}:

$$PR = CR_1 \times \prod_{i=2}^{n-1} CR_i^{pr_{1i}} \times CR_n^{\sum_{j=2}^{n-1} pr_{1j} \times pr_{jn}}$$

3. Calculate the mean for each component (CR_i) using the formula $m = (ID - q)^{-1} = \sum_{u=0}^{\infty} q^u$, where m is the mean number of visits from components i to j, ID is the identity matrix, and q is the transition matrix explained below.
4. Calculate the variance of each component CR_i using the formula $v = m(2m_{dig} - ID) - m_{sqr}$,where v is the variance of each component, $\mathbf{m_{dig}}$ is the diagonal of matrix m, and $\mathbf{m_{sqr}}$ is the square of matrix m.
5. Determine the system reliability

$$SR = \left[\prod_{j=1}^{n-1} CR_j^{m_{1,j}} + \frac{1}{2}(CR_j^{m_{1,j}})(\log CR_j)^2 v_{1,j} \right] CR_n,$$

where, $m_{1,j}$ and $v_{1,j}$ are the visit mean and variance, respectively of each component visit from component 1 to j component.

6. Apply the Markovian property discussed above to the transition probabilities (Pr_{ij}). We consider n-step transition probabilities for computing these transition probabilities to form a Markov chain.

$$Pr_{ij}^{(n)} = \sum_{k=0}^{M} Pr_{ik}^{(m)} \times Pr_{kj}^{(n-m)} \quad \forall \ i=0, 1, 2, \ldots M;$$

j=0, 1, 2 ...M; and any m=1, 2,..., n-1;
n=m+1, m+2, …

7. Apply the MCalgorithm to estimate the system reliability.
8. Repeat Steps 3 and 4 to determine a convergence posterior distribution.

We used the Chapman–Kolmogorov equations for the n-step transition probabilities [19].

IV. IMPLEMENTATION OF UNCERTAINTY ANALYSIS

We discuss the uncertainty analysis of the components with a waiting queue simulator application [6] as a case study. The architecture of the application consists of the following components: an ArrivalGenerator, EventList, ServiceFacility, ScheduleManager, QueuingFacility, and Measurement recorder.

The interactions between the components were evaluated using the scenarios. The architecture of the waiting queue application is represented by a graph called the component dependency graph (CDG). A CDG consists of components and the transition probabilities between the components. The interaction between the components forms possible execution paths from the starting component to the end component. Thus, it follows a path-based model. Here, we consider the graph traversal with three structures: sequences, branches, and looping.

Fig. 1 shows a CDG graph represented with nodes and edges. The nodes and edges consist of information such as component name, component reliability, transition probability from one component to another, average execution time of each component, and transition reliability from one component to another. Assuming each component's reliability is 1, the average execution times between components are shown in Table 1. The average execution time was evaluated using the following formula:

$$AET_i = \sum_{i=1}^{|s|} \Pr s_k \times T(CR_i)_{CR_i \in s_k},$$

Where AET_i is the average execution time of each component CR_i, Prs_k is the probability of executing scenario s_k, $T(CR_i)$ is the execution time of CR_i, i.e., the total number of active times in the execution of scenario s_k. Table 2 lists the transition probabilities between the components of the waiting queue system.

The values in Table 2 were obtained from the CDG in Fig. 2. The transition probabilities between two components ($pr_{i,j}$) are evaluated as follows. It is the probability of transitioning between two components (C_i, C_j), computed as the interactions between two components during the reliability analysis [6]. The sum of transaction probability from any component is always 1, i.e., $\sum_{j=1}^{n} Pr_{in} = 1$.

Fig. 2 shows the corresponding DTMC-based execution path from starting component C2 to end component C7. We consider the structure based on the architecture of the appli-

Table 1. Reliabilities and average execution times of components

Index (CI)	Reliability (CR_i) (Assuming 1 for all components)	Name (CN_i)	Average Execution Time ($AvgET_i$)
1	1	Schedule_Manager	5
2	1	Arrival_Gen	2
3	1	Service_Facility	7
4	1	EventList	14
5	1	QueuingFacility	5
6	1	Measurement	0.8
7	1	End	0

Table 2. Transition Probability between Components (C_i and C_j)

$Pr_{12} = 0.05$	$Pr_{13} = 0.215$	$Pr_{14} = 0.475$	$Pr_{15} = 0.26$
$Pr_{21} = 1$			
$Pr_{31} = 0.1787$	$Pr_{34} = 0.215$	$Pr_{36} = 0.0363$	$Pr_{37} = 0.57$
$Pr_{41} = 0.5725$	$Pr_{43} = 0.215$	$Pr_{45} = 0.2125$	
$Pr_{51} = 0.3575$	$Pr_{54} = 0.2125$	$Pr_{57} = 0.43$	
$Pr_{63} = 1$			

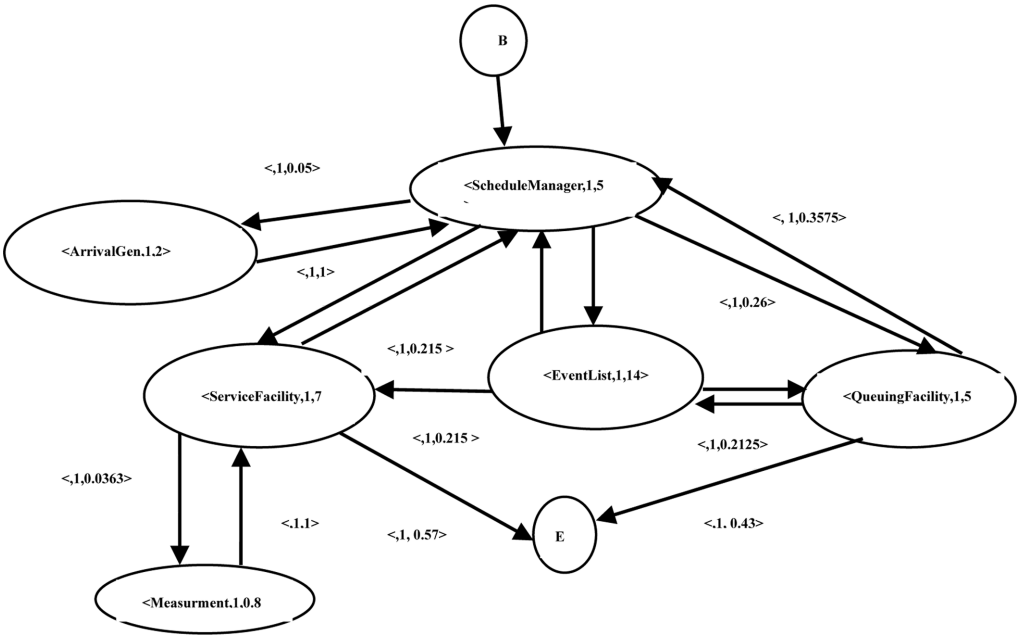

Fig. 1. CDG of the queue management application.

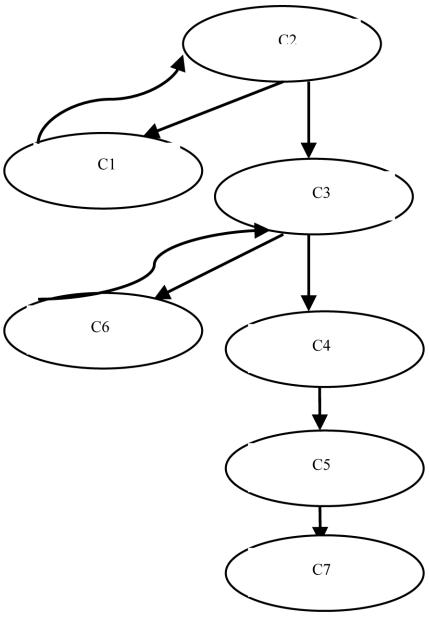

Fig. 2. Path-based structure from the CDG for the waiting queue application.

cation shown in Fig. 1. Fig. 2 consists of 3 types of path structure: a) sequence, b) branching, and c) looping. Hence, we apply the path reliability [35] for Fig. 2 and estimate the system reliability as follows. Fig. 2 obtains the expression for system reliability using the path-based approach, and its expression is as follows:

$$PR = CR_2 \times CR_1^{\frac{Pr_{21}}{1-Pr_{12}\times Pr_{23}}} \times CR_3^{\frac{Pr_{21}\times Pr_{23}}{1-Pr_{23}\times Pr_{32}\times Pr_{36}}} \times CR_6^{\frac{Pr_{21}\times Pr_{23}\times Pr_{36}}{1-Pr_{63}\times Pr_{36}\times Pr_{34}}} \times CR_4^{\frac{Pr_{21}\times Pr_{23}\times Pr_{36}\times Pr_{34}}{1-Pr_{63}\times Pr_{23}}} \times CR_5^{\frac{Pr_{21}\times Pr_{23}\times Pr_{36}\times Pr_{34}\times Pr_{45}}{1-Pr_{63}\times Pr_{23}}} \times CR_7^{\frac{Pr_{21}\times Pr_{23}\times Pr_{36}\times Pr_{34}\times Pr_{45}}{1-Pr_{63}\times Pr_{23}}}$$

The above expression is used in the method of moments to compute the mean and variance of the system reliability.

We demonstrate that the uncertainty analysis for the application has six components, and its architecture is described by the DTMC presented in Fig. 2. To compute the method of moments using Taylor series, we consider the nonzero transition probabilities Pr_{ij} given in Table 2 as the mean values. Table 3 shows the mean component reliabilities $E[Re_i]$ and variance of the component reliabilities $V[Re_i]$. We found that the uncertainty of component reliabilities results in uncertainty in software reliability. For this reason, we used the mean values of the component reliabilities and variances shown in Table 3 in the equations describing the method of moments in Section III. We observe from Table 4 that, if Taylor's expansion order is greater, then the accuracy increases, and the variance is less. Thus, the system reliability increases. This indicates that, if the variance is less, the system is more reliable.

Table 3. Mean and variance of the components

Component Index(CI)	Expected components reliability $E[Re_i]$	Variance of components $V[Re_i]$
1	0.9809	0.0565
2	0.9891	0.024
3	0.9892	0.0324
4	0.9731	0.0041
5	0.9893	0.123
6	0.9890	0.345
7	0.970	0.0565

Table 4. Results of the mean and variance of the system reliability for the waiting queue

	1st order Taylor expansion	2nd order Taylor expansion
mean calculation value	0.778	0.758
variance calculation value	0.00022362	0.0002162

A. Waiting Queue Management Case Study based on MC Simulation Method

The uncertainty analysis for the case study evaluates the system reliability using the formula $SR = f(CR_i, Pr_{ij})$. We obtained the results by varying the component reliability from 0.95 to 0.98, assuming that $E[CR_1] = E[CR_7] = 1$ and $V[CR_1]=V[CR_7]=0$. From Fig. 2, we obtain the mean of the component reliability and variance of each component's reliability. The system reliability was 92%. Table4 presents the evaluated values of E[CRi] and V[CRi] without the MC simulation method.

Similarly, applying the MCmethod for the component reliability varies in the range [0.95, 0.98]; the mean of the component reliability and variance of the component reliability are described in Table 6. The MCmethod was executed several times (approximately 100 times), and we obtained an overall system reliability of 93%. The MCmethod was

Table 5. Results for mean and variance of each component's reliability

Component #	Mean of component Reliability $E[CR_i]$	Variance of Component Reliability, $V[CR_i]$
1	1.000	0
2	0.9828	0.9496
3	0.8268	0.4239
4	0.7130	0.9769
5	0.8431	0.5565
6	0.7461	0.3645
7	1.000	0

Table 6. Results for mean and variance of each component's reliability using MC Simulation

Component #	Mean of component Reliability E1[CR_i]	Variance of Component Reliability, V1[CR_i]
1	1.000	0
2	0.9837	0.9823
3	0.8997	0.4079
4	0.9641	0.9515
5	0.9335	0.5197
6	0.8021	0.3401
7	1.000	0

described in Section III. Hence, the values of alpha(x) = 10 and beta(y) = 2.675 were set for further calculations.

The results also show that if the variance of the components is small, the system reliability increases.

B. Example of MCMC Method

The case study used for the MCmethod was also used for the MCMC method. Here, the MCMC applied for component reliability varies in the range [0.95 to 0.98], and the results are discussed below. Table7 shows the mean and variance of component reliability using the MCMC method. The n-step Markov chain is important for defining the probability of every component of a system at a random time. We start with any component as well; we reach the same probability distribution with a higher system reliability. The MCMC procedure consists of two steps.

1. Use the Markov chain process (for this n-step transition probability).
2. Apply the MCsimulation method.
3. The overall system reliability obtained using the MCMC method is 96.8%, i.e., ≈ 97%.

V. COMPARISON STUDY

The suitable selection of the method used for the analysis of uncertainty in software reliability for a given application depends on the requirements of the data, acquired reliability estimate, and performance of the solution. In this section, we explain the three methods used for the uncertainty analysis. We found that the method moment provided a system reliability of 92%. The MCsimulation showed that the overall system reliability was 93%. Similarly, the MCMC simulation provided a system reliability of 97%, with a lower variance than that of the MCmethod. Table 8 shows the variance of each component using the MC and MCMC methods. We found that the variance of each component decreased in the MCMC method compared to the MC method. Method of moments does not require a probability distribution. In method of moments, random numbers are not required. To increase the accuracy of method of moments, a high-order Taylor function is used.

MCsimulation increases the accuracy by expanding the simulation numbers randomly generated. MC simulation includes a collection of reliability methods, such as moments, probability distributions, system reliability, and lower variance, to improve confidence in the system reliability estimation. In the MCsimulation method, uncertainty arises because of the operational profile and reliability of the components. Compared with the MCsimulation method, the MCMC simulation is better. In the MCMC simulation, the Markov chain was used for the probability distribution. This enables the quantification of the variance and accuracy of the reliability system estimation is high. This implies that the results obtained from different uncertainty methods help decision makers to make better decisions.

From [6], we considered the component dependency graph for the calculation of system reliability using the MCMC technique. In [6], the component reliability, average execution time, and transition probability between two components from the scenarios of the application were evaluated. However, we considered the evaluation of the mean and variance of each component based on the random reliability of each component. (In [6], the reliability of each component was fixed at 1). Table 9 lists comparisons between the results from previous studies and the proposed method.

Table 7. Results for mean and variance of each component's reliability using MCMC simulation

Component #	Mean of component Reliability E[CR_i]	Variance of Component Reliability, V[CR_i]
1	1.000	0
2	0.2344	0.0234
3	0.0095	0.0093
4	0.0026	0.0021
5	0.0014	0.0014
6	0.2478	0.0114
7	1.000	0

Table 8. Comparison results of variance of each component using MCMC and MCsimulation

Component ID	Variance of components using MCmethod	Variance of components using Markov Chain method
1	0	0
2	0.9823	0.0234
3	0.4079	0.0093
4	0.9515	0.0021
5	0.5197	0.0014
6	0.3401	0.0114
7	0	0

Table 9. Comparisons with previous studies

Sl. No.	Investigator	Title	Input parameters	Approach	Findings
1.	[6]	A scenario-based reliability analysis approach for component-based software.	Component dependence graph, component reliability, average execution time, and transition probability	They suggest an application reliability with a scenario-based reliability analysis algorithm using the component dependency graph	This method helps discover the critical components and interfaces, as well as study the sensitivity of the application reliability to these components.
2.	Proposed Model	Component Based System Reliability using MCMC Simulation	Component dependence graph, component reliability, average execution time, and transition probability.	Evaluation of the application reliability using the mean and variance of each component. The MCMC simulation method is then applied.	This technique is used to quantify the variance of each component, and the estimated system reliability accuracy is high.

VI. CONCLUSION

Architecture-based software reliability methods use several models (such as Markov chains and fault trees) to predict the reliability of an application. We analyzed the uncertainty in parameters, transition probabilities between components, and the reliability of individual components that propagate the software reliability estimation. Using these methods, we obtain the values for the mean and variance of the system reliability, which are considered for the uncertainty owing to their operational profile and component reliabilities. We presented the waiting queue management application architecture and its reliability calculation method with input values such as component reliability and transition probability between components. The estimated value of the system reliability using MCMC simulation provides more information than the MC simulation, or even method of moments. Consequently, the system has a higher confidence in reliability estimation with a smaller variance. In the MCMC method, owing to the "memory lessness" property, knowing the history of the process does not improve future predictions. The system reliability was analyzed by considering the uncertainty of the component, as shown in Table 3. From this, we can conclude that before applying the uncertainty measure, the reliability is 0.93, and by applying the uncertainty method to the components, the system reliability changes to 0.92 for method of moments, 0.93 for MC, and 0.97 for MCMC. Thus, we conclude that the overall system reliability is insignificantly affected, even if the individual components are subjected to uncertainty.

We consider that the uncertainty analysis of software reliability is important, primarily for predicting software reliability during the analysis stage in the life cycle. Sensitivity analysis provides intuition about the input uncertainty that controls the output uncertainty towards effective resources, which can be performed on the component, and transition probabilities can be considered in the future scope of this paper.

REFERENCES

[1] R. C. Cheung, "A user-oriented software reliability model", *IEEE transactions on Software Engineering*, vol. SE-6, no. SE-2, pp. 118-125, Mar. 1980. DOI: 10.1109/TSE.1980.234477.

[2] B. Littlewood, "Software reliability model for modular program structure," *IEEE Transactions on Reliability*, vol. R-28, no. 3, pp. 241-246, Aug. 1979. DOI: 10.1109/TR.1979.5220576.

[3] J. -C. Laprie, "Dependability evaluation of software systems in operation," *IEEE Transactions on software engineering*, vol. SE-10, no. 6, pp. 701-714, Nov. 1984. DOI: 10.1109/TSE.1984.5010299.

[4] P. Kubat, "Assessing reliability of modular software," *Operations research letters*, vol. 8, no. 1, pp. 35-41, Feb. 1989. DOI: 10.1016/0167-6377(89)90031-X.

[5] S. S. Gokhale and K. S. Trivedi, "Reliability prediction and sensitivity analysis based on software architecture," in *Proceedings of the 13th International Symposium on Software Reliability Engineering*, Annapolis: MD, USA, pp. 64-75, 2002. DOI: 10.1109/ISSRE.2002.1173214.

[6] S. Yacoub, B. Cukic, and H. H. Ammar, "A scenario-based reliability analysis approach for component-based software," *IEEE transactions on reliability*, vol. 53, no. 4, pp. 465-480, Dec. 2004. DOI: 10.1109/TR.2004.838034.

[7] J. D. Musa, "Operational profiles in software-reliability engineering," *IEEE software*, vol. 10, no. 2, pp. 14-32, Mar. 1993. DOI: 10.1109/52.199724.

[8] A. L. Goel, "Software reliability models: Assumptions, limitations, and applicability," *IEEE Transactions on software engineering*, vol. SE-11, no, 12, pp. 1411-1423, Dec. 1985. DOI: 10.1109/TSE.1985.232177.

[9] K. Goseva-Popstojanova, A. P. Mathur, and K. S. Trivedi, "Comparison of architecture-based software reliability models," in *Proceedings of the 12th International Symposium on Software Reliability Engineering*, Hong Kong, China, pp. 22-31, 2001. DOI: 10.1109/ISSRE.2001.989455

[10] M. R. T. Lyu, "Software reliability theory," in *Encyclopaedia of Software Engineering*. DOI:10.1002/0471028959.sof329.

[11] J. D. Musa, A. Iannino and K. Okumoto, "Software reliability," in *Advances in computers*, vol. 30, Academic Press Inc., pp. 85-170, 1990.

[12] S. S. Gokhale, "Architecture-based software reliability analysis: Overview and limitations," *IEEE Transactions on dependable and secure computing*, vol. 4, no. 1, pp. 32-40, Jan-Mar. 2007. DOI: 10.1109/TDSC.2007.4.

[13] A. L. Goel and K. Okumoto, "A Markovian model for reliability and other performance measures of software systems," in *1979 International Workshop on Managing Requirements Knowledge (MARK)*,

New York: NY, pp. 769-774, 1979. DOI: 10.1109/MARK.1979.8817248.

[14] S. K. Chandran, A. Dimov, and S. Punnekkat, "Modeling uncertainties in the estimation of software reliability–a pragmatic approach," in *2010 Fourth International Conference on Secure Software Integration and Reliability Improvement,* Singapore, pp. 227-236, Jun. 2010. DOI: 10.1109/SSIRI.2010.22.

[15] L. Cheung, L. Golubchik, N. Medvidovic, and G. Sukhatme, "Identifying and addressing uncertainty in architecture-level software reliability modelling," in *2007 IEEE International Parallel and Distributed Processing Symposium*, Long Beach: CA, USA, pp. 1-6. Mar. 2007. DOI: 10.1109/IPDPS.2007.370524.

[16] Y. S. Dai, M. Xie, Q. Long, and S. -H. Ng, "Uncertainty analysis in software reliability modeling by bayesian analysis with maximum-entropy principle," *IEEE Transactions on Software Engineering*, vol. 33 no. 11, pp. 781-795, Oct. 2007. DOI: 10.1109/TSE.2007.70739

[17] W. W. Everett, "Software component reliability analysis," in *Proceedings 1999 IEEE Symposium on Application-Specific Systems and Software Engineering and Technology. ASSET'99 (Cat. No. PR00122)*, Richardson: TX, USA, pp. 204-211, 1999**.** DOI: 10.1109/ASSET.1999.756770.

[18] L. Yin, M. A. Smith, and K. S. Trivedi, "Uncertainty analysis in reliability modelling," in *Annual Reliability and Maintainability Symposium. 2001 Proceedings. International Symposium on Product Quality and Integrity (Cat. No. 01CH37179)*, Philadelphia: PA, USA, pp. 229-234, 2021. DOI: 10.1109/RAMS.2001.902472.

[19] M. A. Haque and N. Ahmad, "Modified Goel-Okumoto Software Reliability Model Considering Uncertainty Parameter," in *Mathematical Modeling, Computational Intelligence Techniques and Renewable Energy*, vol. 1405, pp. 369-379, 2022. DOI: 10.1007/978-981-16-5952-2_32.

[20] Z. Liu, S. Yang, and R. Kang, "Software belief reliability growth model based on uncertain differential equation," *IEEE Transactions on Reliability*, pp. 1-13, Mar. 2022. DOI: 10.1109/TR.2022.3154770.

[21] R. Dhaka, B. Pachauri, and A. Jain, "Two-Dimensional Software Reliability Model with Considering the Uncertainty in Operating Environment and Predictive Analysis," in *Data Engineering for Smart Systems*, vol. 238, pp. 57-69, 2022. DOI: 10.1007/978-981-16-2641-8_6.

[22] C. Zhang and A. Mostashari, "Influence of component uncertainty on reliability assessment of systems with continuous states," *International Journal of Industrial and Systems Engineering*, vol. 7, no. 4, pp. 542-552, Apr. 2011.

[23] S. V. Dhople, Y. C. Chen, and A. D. Domínguez-García, "A set-theoretic method for parametric uncertainty analysis in Markov reliability and reward models," *IEEE Transactions on Reliability*, vol. 62, no. 3, pp. 658-669, Sep. 2013. DOI: 10.1109/TR.2013.2270421.

[24] P. Ju, H. Li, X. Pan, C. Gan, Y. Liu, and Y. Liu, "Stochastic dynamic analysis for power systems under uncertain variability," *IEEE Transactions on Power Systems*, vol. *33*, no. 4, pp. 3789-3799, Jul. 2018. DOI: 10.1109/TPWRS.2017.2777783.

[25] I. Meedeniya, I. Moser, A. Aleti, and L. Grunske, "Architecture-based reliability evaluation under uncertainty," in *Proceedings of the joint ACM SIGSOFT conference--QoSA and ACM SIGSOFT symposium--ISARCS on Quality of software architectures--QoSA and architecting critical systems--ISARCS*, pp. 85-94, Jun. 2011. DOI: 10.1145/2000259.2000275.

[26] L. Fiondella and S. S. Gokhale, "Software reliability with architectural uncertainties," in *2008 IEEE International Symposium on Parallel and Distributed Processing*, Miami: FL, USA, pp. 1-5, 2008. DOI: 10.1109/IPDPS.2008.4536436.

[27] K. Go, M. Hamill, and X. Wang, "Adequacy, accuracy, scalability, and uncertainty of architecture-based software reliability: Lessons learned from large empirical case studies," in *2006 17th International Symposium on Software Reliability Engineering,* Raleigh: NC, USA, pp. 197-203, 2006. DOI: 10.1109/ISSRE.2006.11.

[28] I. Gashi, P. Popov, and V. Stankovic, "Uncertainty explicit assessment of off-the-shelf software: A Bayesian approach," *Information and Software Technology*, vol. 51, no. 2, pp. 497-511, Feb. 2009. DOI:10.1016/j.infsof.2008.06.003.

[29] K. Goseva-Popstojanova and S. Kamavaram, "Assessing uncertainty in reliability of component-based software systems," in*14th International Symposium on Software Reliability Engineering, ISSRE 2003*, Denver: CO, USA, pp. 307-320.DOI:10.1109/ISSRE.2003.1251052.

[30] K. Goseva-Popstojanova and S. Kamavaram, "Uncertainty analysis of software reliability based on method of moments," in *13th Int'l Symp. Software Reliability Engineering*, pp. 143-144, 2002.

[31] D. K. Mohanta and D. S. Roy, "Importance and uncertainty analysis in software reliability assessment of computer relay," in *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 225, no. 1, pp. 50-61, Jun. 2011. DOI: 10.1177/1748006XJRR341.

[32] Y. Wang, X. Gao, Y. Cai, M. Yang, S. Li, and Y. Li, "Reliability evaluation for aviation electric power system in consideration of uncertainty," *Energies*, vol. 13, no. 5, p. 1175, Mar. 2020. DOI: 10.3390/en13051175.

[33] A. K. Verma, S. Ajit, and D. R. Karanki, "Uncertainty analysis in reliability/safety assessment," in *Reliability and Safety Engineering*, pp. 457-491, Sep. 2015. DOI: 10.1007/978-1-4471-6269-8_13.

[34] K. Goševa-Popstojanova and K. S. Trivedi, "Architecture-based approach to reliability assessment of software systems," *Performance Evaluation*, vol. 45, no. 2-3, pp.179-204, Jul. 2001. DOI: 10.1016/S0166-5316(01)00034-7.

[35] C. -J. Hsu and C. -Y. Huang, "An adaptive reliability analysis using path testing for complex component-based software systems," *IEEE Transactions on Reliability*, vol. 60, no. 1, pp. 158-170, Mar. 2011. DOI: 10.1109/TR.2011.2104490.

[36] D. V. Ravenzwaaij, P. Cassey, and S. D. Brown, "A simple introduction to Markov Chain Monte–Carlo sampling," *Psychonomic bulletin & review*, vol. 25, no. 1, pp. 143-154, Mar. 2018. DOI: 10.3758/s13423-016-1015-8.

[37] B. Zhou, H. Okamura, and T. Dohi, "Markov chain Monte Carlo random testing," in *Advances in Computer Science and Information Technology*, Heidelberg, German, vol. 6059, pp. 447-456. DOI: 10.1007/978-3-642-13577-4_40.

[38] F. S. Hillier and G. J. Lieberman, "A special algorithm for the assignment problem," in *Introduction to Operations Research*, 9th ed., McGraw-Hill Education, pp. 342-346, 2008.

[39] D. Hamlet, D. Mason, and D. Woitm, "Theory of software reliability based on components," in *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*, Toronto: ON, Canada, pp. 361-370, 2001. DOI: 10.1109/TR.2011.2104490.[40] S. Krishnamurthy and A. P. Mathur, "On the estimation of reliability of a software system using reliabilities of its components," in *Proceedings of The Eighth International Symposium on Software Reliability Engineering*, Albuquerque: NM, USA, pp. 146-155, 1997. DOI: 10.1109/ISSRE.1997.630860.

[40] M. L. Shooman, "Structural models for software reliability prediction," in *Proceedings of the 2nd International Conference on Software engineering*, pp. 268-280, Oct. 1976.

[41] M. Xie and C.Wohlin, "An additive reliability model for the analysis of modular software failure data," in *Proceedings of Sixth Inter-*

national Symposium on Software Reliability Engineering. ISSRE'9, Toulouse, France, pp. 188-194, 1995. DOI: 10.1109/ISSRE.1995.497657.

[42] F. Zhang, X. Zhou, J. Chen, and Y. Dong, "A novel model for component-based software reliability analysis," in *2008 11th IEEE High Assurance Systems Engineering Symposium,* Nanjing, China, pp. 303-309. 2008. DOI: 10.1109/HASE.2008.41.

Sampa Chau Pattnaik

received her Bachelor's in Computer Science from B.J.B.(Auto.) College (Utkal University), Orissa, India in 2002. She received her Master's in Computer Science& Engineering from Utkal University Bhubaneswar, Orissa, India in 2008. Now, she continues her research (Ph.D.) at Siksha 'O' Anusandhan Deemed to be University. Her research interests include software engineering and reliability testing.

Mitrabinda Ray

received her Ph.D. degree from NIT, Rourkela. Now, she is as an associate professor at Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar, in the department of Computer Science and Engineering. Her area of interest is software testing, reliability, and estimation. She has published several papers in different journals and conferences.

Mitali Madhusmita Nayak

received her Ph.D. degree from Siksha 'O' Anusandhan Deemed to be University. She is an associate professor with the Department of Mathematics, Institute of Technical Education and Research at Siksha 'O' Anusandhan. She has over 15 years of teaching experience and 10 years of research experience. Her research interests are in the fields of Numerical Analysis, Inventory management, Fuzzy optimization, and Decision science.

Srikanta Patnaik

is a Professor in the Department of Computer Science and Engineering, Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar, India. He received his Ph. D. (Engineering) on Computational Intelligence from Jadavpur University, India in 1999. He has supervised more than 25 Ph. D. and 60 Master theses in the field of Computational Intelligence, Machine Learning, Soft Computing Applications, and Re-Engineering. Dr. Patnaik has published around 100 research papers in international journals and conference proceedings. He has authored 2 textbooks,42 edited volumes, and a few book chapters as a guest writer; published by leading international publishers such as Springer-Verlag and Kluwer Academic. Dr. Patnaik is the Editors-in-Chief of the International Journal of Information and Communication Technology and International Journal of Computational Vision and Robotics published by Inderscience Publishing House, England as well as Editors-in-Chief of the Book Series on "Modeling and Optimization in Science and Technology" published by Springer, Germany. He is also an Associate Editor for the International Journal of Telemedicine and Clinical Practices (IJTMCP) and International Journal of Granular Computing, Rough Sets and Intelligent Systems (IJGCRSIS)