Assessing Uncertainty in Reliability of Component–Based Software Systems

Katerina Goševa–Popstojanova and Sunil Kamavaram
Lane Department of Computer Science and Electrical Engineering
West Virginia University, Morgantown, WV 26506–6109
{katerina, sunil}@csee.wvu.edu

Abstract

Many architecture–based software reliability models were proposed in the past. Regardless of the accuracy of these models, if a considerable uncertainty exists in the estimates of the operational profile and components reliabilities then a significant uncertainty exists in calculated software reliability. Therefore, the traditional way of estimating software reliability by plugging point estimates of unknown parameters into the model may not be appropriate since it discards any variance due to uncertainty of the parameters. In this paper we propose a methodology for uncertainty analysis of architecture–based software reliability models suitable for large complex component based applications and applicable throughout the software life cycle. First, we describe different approaches to build the architecture based software reliability model and to estimate parameters. Then, we perform uncertainty analysis using the method of moments and Monte Carlo simulation which enable us to study how the uncertainty of parameters propagates in the reliability estimate. Both methods are illustrated on two case studies and compared using several criteria.

1 Introduction

A number of analytical models have been proposed to address the problem of quantifying software reliability. One group of models is focused on modeling reliability growth during testing phase [4]. These so called black–box models treat the software as monolithic whole, considering only its interactions with external environment, without an attempt to model the internal structure. Black–box models are clearly inappropriate for large component–based systems. For these systems, we need to use a white–box approach that takes into account the information about the architecture of software made out of components. An extensive survey on architecture–based software reliability models is presented in [6].

Two important questions arise with respect to predications of software reliability based on models. The first question addresses the appropriateness of the model. Thus, the model could be inappropriate because its assumptions may not hold in practice. The second question addresses the accuracy of parameters values. Parameters can be estimated using the field data obtained during testing or operational usage of the software, historical data for products with similar functionality, or reasonable guesses based on the specification and design documentation. In practice, there is a lot of uncertainty around parameters because they rarely can be estimated accurately. The traditional way of estimating software reliability by plugging point estimates of the unknown parameters into the model [4], [6] is not appropriate since it discards any variance due to uncertainty of the parameters.

Traditionally, the most common method for uncertainty analysis in software reliability is conducting sensitivity studies. Thus, sensitivity of the software reliability estimation to errors in the operational profile has been investigated in the context of black–box reliability growth models [2], [19], [21]. Sensitivity studies of software reliability estimates obtained using architecture–based models have been presented in [3], [22]. In these studies the authors assume fixed known values for the transition probabilities (i.e., fixed operational profile) and derive the sensitivity of the system reliability with respect to the reliability of each component. Sensitivity studies of software reliability with respect to the operational profile (i.e., transition probabilities) and component reliabilities were presented in [5], [7], [27].

In addition to sensitivity studies, there have been several attempts to quantify the variability of software reliability. In [17] authors used black–box approach and assumed that the

Proceedings of the 14th International Symposium on Software Reliability Engineering (ISSRE'03)
1071-9458/03 $ 17.00 © 2003 IEEE

failure probability has prior Beta distribution. Using Bayesian approach they derived the mean and the variance of the failure probability for a software system that, in its current version, has not failed. The same problem was considered in [1] for the software with partitioned input domain. However, in this work it was recognized that there is uncertainty in the estimations of the reliability for each partition (using Beta prior distribution), as well as uncertainty in the probability of using each partition (using Dirichlet distribution). In [23] the mean and the variance of software failure probability were estimated using Bayesian approach and assuming Beta prior distributions for component failure probabilities. In another related work [15] three optimization models for software reliability allocation under an uncertain operational profile were formulated and solved.

Several papers that use discrete time Markov chains to describe software usage are also relevant to our work, although they do not consider software reliability. Thus, in [26] Markov analysis of software specifications was presented and entropy was used as a measure of uncertainty. In [25] the impact of uncertainties in the operational profile on the usage coverage was analyzed. Uncertainties were specified as intervals of transition probabilities assuming a uniform distribution in the interval.

From the above it is obvious that uncertainty analysis was not used systematically and extensively in software reliability. However, it has been applied in other engineering disciplines. Thus, several methods for uncertainty analysis of system characteristics from uncertainties in component characteristics are presented in [9], [10], [28].

In this paper we propose a methodology for uncertainty analysis of architecture–based software reliability models suitable for large complex component–based applications and applicable throughout the software life cycle. The methodology addresses the parameter uncertainty problem and enables us to study how the uncertainty of parameters propagates in the system reliability. Within this methodology we are considering several different methods for uncertainty analysis. Previously we have used entropy for uncertainty analysis [13]. Entropy is a well know concept from information theory that allows us to quantify the uncertainty of the operational profile, uncertainty of the overall system reliability, and component uncertainties. Although the obtained results are useful for verification and validation of component–based systems, this method does not provide an estimate of software reliability. In this paper we use method of moments and Monte Carlo simulation to quantify the uncertainty of software reliability estimates.

The proposed methodology provides a systematic way for uncertainty analysis that can be used for keeping track of the software evolution throughout the life cycle. Uncertainty assessment also provides valuable information for allocation of testing efforts. Also, it can be used for certification of software system given its structure and the inaccuracy in estimation of its usage. This is an important aspect of our work, because with the growing emphasis on reuse developers cannot afford to stay away from reliability certification.

The rest of the paper is organized as follows. The proposed methodology for uncertainty analysis is presented in Section 2. The application of the method of moments and Monte Carlo simulation as methods for uncertainty analysis in software reliability are described in Section 3 and Section 4, respectively. In Section 5 we present the numerical results for two case studies. In Section 6 we compare different methods for uncertainty analysis. Finally, the concluding remarks are presented in Section 7.

2 Methodology for uncertainty analysis

In order to estimate the system reliability using architecture–based model we need to know the software architecture (structure of component interactions), software usage described by the operational profile (relative frequencies of component interactions), and software failure behavior (component reliabilities or failure rates). In [7] we have shown that the architecture–based software reliability model presented in [3] provides reliability estimates close to the actual measured reliability, that is, we have validated the model appropriateness. In this paper we propose a methodology for uncertainty analysis (see Figure 1) and focus on the assessment of the uncertainty in software reliability due to uncertainty of modeling parameters. Next, we describe the proposed methodology in detail.

2.1 Software architecture

Software behavior with respect to the manner in which different components interact is defined through the software architecture. We use state–based approach to build the architecture–based software reliability model [6], [7]. This approach uses the control flow graph to represent software architecture. The states represent active components and the arcs represent the transfer of control. Based on the assumption that the transfer of control between

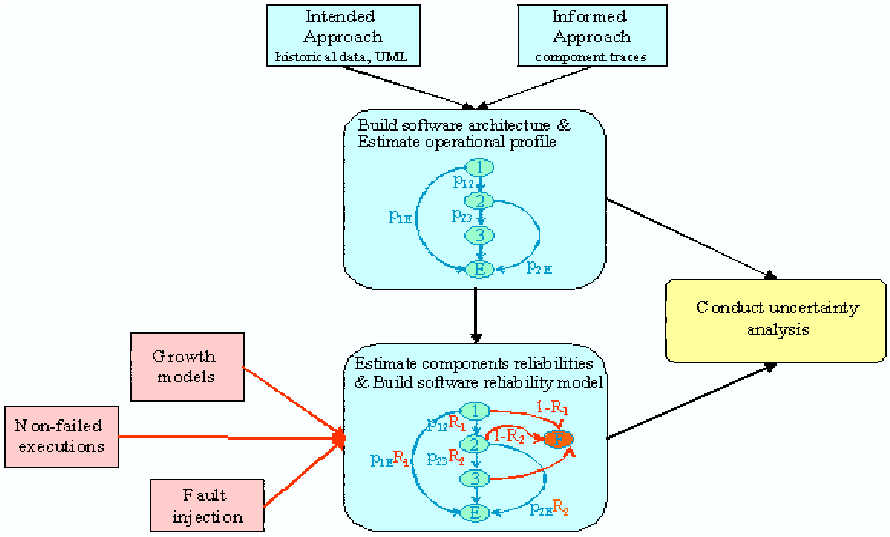

Figure 1. Methodology for uncertainty analysis of software reliability

components has a Markov property, the architecture is modeled with a discrete time Markov chain (DTMC) with a transition probability matrix $P = [p_{ij}]$, where $p_{ij} = \Pr\{\text{control is transferred from component } i \text{ to component } j\}$. Next, we briefly describe two different approaches that we use to build a DTMC that represents dynamic software architecture.

- **Intended approach** is used in early phases of software development. We base our estimates on historical data from similar products or on high level information about software architecture obtained from specification and design documents. Since, UML is rapidly becoming a standard for software development, in intended approach we are looking into the UML annotations such as use cases and sequence diagrams [29].

- **Informed approach** is used during late phases of software development when testing or field data become available. Thus, component traces obtained using profilers [30] and test coverage tools [31] can be used to obtain a set of execution paths and establish the frequency count of the transition arcs.

Dynamic information in software architecture clearly depends on the software usage, that is, the operational profile. Operational profiles have been developed successfully for the applications such as real–time telecommunication systems where the use of the software is predictable because it is related to identifiable events due to human activity [18]. In general, the estimation of a trustworthy operational profile is difficult because it requires anticipating the field usage of the software and a priori knowledge about the application and system environments. A typical example would be process control applications in which various software components are activated by complex sequences of events whose frequencies can hardly be estimated a priori. In other cases, a single operational profile is not sufficient to describe the use of the product by different users. Further problems could arise when functions are added or modified as software systems evolve. As a consequence, the way in which the software is used also evolves, and the operational profile changes. These reasons can easily lead to erroneous estimates of the operational profile which will directly affect the reliability estimate. Therefore, it is important to conduct uncertainty analysis due to uncertainty in the operational profile estimation.

2.2 Component failure behavior

The next step in our methodology is to consider components failure behavior, i.e., estimate the reliability of each component. The reliability of the component i is the probability R_i that the component performs its function correctly. Assessing the reliability of software components clearly depends on the factors such as whether or not component code is available, how well the component has been tested, and whether it is a reused or a new component.

Several techniques for estimating component's reliability have been proposed. *Software reliability growth models* can be applied to each software component exploiting component's

failure data obtained during testing [4], [14]. However, due to the scarcity of failure data it is not always possible to use software reliability growth models. Another possibility is to estimate component's reliability from explicit consideration of *non–failed executions*, possibly together with failures [16], [17], [20]. The problem that arises with these models is the large number of executions necessary to establish a reasonable statistical confidence in the reliability estimate. Finally, one can use *fault injection technique* to estimate component's reliability [7], [24]. However, fault–based techniques are only as powerful as the range of fault classes that they simulate. Regardless of the technique, the estimates of component reliabilities may be inaccurate, which further motivates the use of uncertainty analysis.

2.3 Combining software architecture with failure behavior

The presented methodology for uncertainty analysis can be applied to any architecture–based software reliability model that has a close form solution for the system reliability. In this paper we use the model first presented in [3] which uses composite method to combine software architecture with failure behavior. Two absorbing states C and F are added to the DTMC, representing the correct output and failure respectively. The transition probability matrix P is modified to $\bar{P}$ as follows. The original transition probability p_{ij} between the components i and j is modified into $R_i p_{ij}$, which represents the probability that the component i produces the correct result and the control is transferred to component j. The failure of a component i is considered by creating a directed edge to failure state F with transition probability $(1 - R_i)$. The reliability of the program is the probability of reaching the absorbing state C of the DTMC. Let Q be the matrix obtained from $\bar{P}$ by deleting rows and columns corresponding to the absorbing states C and F. The $(1, n)$ entry of the matrix Q^k represents the probability of reaching state n from 1 through k transitions. From initial state 1 to final state n, the number of transitions k may vary from 0 to infinity. It can be shown that $S = \sum_{k=0}^{\infty} Q^k = (I - Q)^{-1}$ which means that the $(1, n)$th element of matrix S denotes the probability of reacing state n from state 1. It follows that the overall system reliability is given by $R = s_{1n} R_n$.

2.4 Uncertainty analysis

For a given software architecture, there are two sources of uncertainty in software reliability: the way software is used (i.e., the operational profile) and the components failure behavior (i.e., components reliabilities). Using the model described in Section 2.3 we obtain the expression for system reliability R as a function of transition probabilities p_{ij} and component reliabilities R_i. Regardless of the accuracy of the mathematical model used to model software reliability, if considerable uncertainty exists in estimation of the operational profile and components reliabilities (as it usually does) then a significant uncertainty exists in calculated system reliability. Therefore, the traditional approach of computing the point estimate of the software reliability by plugging point estimates of the parameters into the model is not appropriate. Alternatively, we can treat unknown parameters as random variables and quantify the uncertainty of system reliability. In this case, the system reliability is also a random variable.

Different methods can be applied for synthesizing uncertainty in software reliability from uncertainties in component reliabilities and transition probabilities. Previously we have used the well–known concepts from information theory for uncertainty analysis [13]. Although this study provides useful measures that can guide the process of identifying critical components and allocating testing time and resources, it does not provide an estimate of the software reliability. In this paper we propose two methods that can be used to assess the uncertainty in software reliability due to the uncertainty of the operational profile and component reliabilities. We first expand our earlier results on the method of moments [8] and then present Monte Carlo simulation technique. Both methods are illustrated on two case studies and compared using criteria such as data requirements, reliability measures derived, and accuracy of the solutions.

It is worth emphasizing that method of moments and Monte Carlo simulation can be used for assessing the uncertainty of software reliability in cases when the software testing does not reveal any failures. Note that the traditional point estimate of system reliability for the software that in its current version has not failed will result in system reliability equal to 1. Of course, unless we do exhaustive testing without replacement, we can never be sure that software reliability is 1.

3 Method of moments

Method of moments is an approximate approach that allows us to generate the moments of system reliability from the moments of component reliabilities, that is, to quantify the uncertainty in software reliability due to the uncertainty

of components reliabilities. Expressions derived in this paper are valid for independent random variables and do not allow us to study the uncertainty of the operational profile, that is, the variation of the transition probabilities p_{ij}. For the method of moments we first obtain the relationship between system reliability R and the component reliabilities $R_1, R_2, \ldots, R_n$ given by the function $R = f(R_1, R_2, \ldots, R_n)$ using the model presented in Section 2.3. If we treat each component reliability on the right–hand side of this expression as a random variable, then the system reliability is also a random variable. Let $E[R_i]$ be the mean value of the ith component reliability and let $\mu_k[R_i] = E[(R_i - E[R_i])^k]$ denote its kth central moment (or moment about the mean). The method of moments allows us to obtain the estimates of the expected value $E[R]$ and kth central moments $\mu_k[R]$ for system reliability based on (1) the knowledge of the system structure and the operational profile $R = f(R_1, R_2, \ldots, R_n)$ and (2) data on the components failures from which estimates of $E[R_i]$ and $\mu_k[R_i]$ for $i = 1, 2, \ldots, n$ can be obtained.

Generating the system reliability moments using the method of moments is based on expanding the function $R = f(R_1, R_2, \ldots, R_n)$ in a multivariable Taylor series expansion around the statistically expected values of each of the component reliabilities $E[R_i]$. Deriving the expression for system reliability and the corresponding Taylor coefficients by hand is cumbersome and can be done only for small systems. Therefore, generation of system reliability moments using the method of moments is a natural candidate for automation. We have used *Mathematica* to derive the symbolic expression for system reliability $R = f(R_1, R_2, \ldots, R_n)$ and its partial derivates for the Taylor series expansion.

The method of moments is an approximate, rather than an exact, method because of the omission of higher order terms in the Taylor series expansion. Thus, the first order Taylor series expansion is given by

$$R \sim a_0 + \sum_{i=1}^{n} a_i (R_i - E[R_i]) \tag{1}$$

where

$$a_0 = f(E[R_1], E[R_2], \ldots, E[R_n]) \tag{2}$$

$$a_i = \left.\frac{\partial R}{\partial R_i}\right|_{R_i = E[R_i]} \text{ for } {}_{i=1,2,\ldots n.} \tag{3}$$

Then, the mean and the variance of system reliability are given by

$$E[R] \sim a_0 \tag{4}$$

$$\mathrm{Var}[R] = \mu_2[R] \sim \textstyle\sum_{i=1}^{n} a_i^2 \mathrm{Var}[R_i]. \tag{5}$$

It is obvious that generating the mean (4) and the variance (5) of the system reliability from the first order Taylor series expansion requires the knowledge of the first two central moments of component reliabilities ($E[R_i]$ and $\mathrm{Var}[R_i]$). Assessing the value of the variance of software reliability is important because it is a measure of confidence in the reliability estimate. Thus, smaller values of the variance correspond to increased confidence.

The accuracy of the $E[R]$ and $\mathrm{Var}[R]$ can be improved by including higher order terms in the Taylor series expansion. We have also derived the second order Taylor series expansion

$$R \sim a_0 + \sum_{i=1}^{n} a_i (R_i - E[R_i]) + \frac{1}{2} \sum_{i=1}^{n} a_{ii} (R_i - E[R_i])^2 + \sum_{i=1}^{n} \sum_{j=1}^{i-1} a_{ij} (R_i - E[R_i])(R_j - E[R_j]) \tag{6}$$

where a_0 and a_i are given by equations (2) and (3) respectively and

$$a_{ii} = \left.\frac{\partial^2 R}{\partial R_i^2}\right|_{R_i = E[R_i]} \text{ for } {}_{i=1,2,\ldots n} \tag{7}$$

$$a_{ij} = \left.\frac{\partial^2 R}{\partial R_i \partial R_j}\right|_{R_k = E[R_k]} \text{ for } {}_{k=1,2,\ldots n.} \tag{8}$$

It follows that the mean and the variance of the system reliability for the second order Taylor approximation are given by

$$E[R] \sim a_0 + \frac{1}{2} \sum_{i=1}^{n} a_{ii} \mathrm{Var}[R_i] \tag{9}$$

$$\mathrm{Var}[R] \sim \sum_{i=1}^{n} a_i^2 \mathrm{Var}[R_i] + \sum_{i=1}^{n} \sum_{j=1}^{i-1} a_{ij}^2 \mathrm{Var}[R_i]\mathrm{Var}[R_j] + \frac{1}{4} \sum_{i=1}^{n} a_{ii}^2 E[(R_i - E[R_i])^4] + \sum_{i=1} a_i a_{ii} E[(R_i - E[R_i])^3] - \frac{1}{4} \sum_{i=1}^{n} a_{ii}^2 (\mathrm{Var}[R_i])^2. \tag{10}$$

Note that generating the mean (9) and the variance (10) of system reliability from the second order Taylor series expansion requires the knowledge of the first four central moments of

component reliabilities. Thus, the second order Taylor approximation provides more accurate estimates for the mean and the variance of software reliability at the price of higher data requirements.

Although the accuracy may be further increased, the derivation of the third or higher order approximations would constitute a formidable task and require higher number of central moments for component reliabilities. Even if the expressions for the third (or higher) order approximation are derived, it might happen that the sampling error due to limited number of observations available for estimation of the central moments of the component reliabilities will exceed the error introduced by the omission of higher order terms.

4 Monte Carlo simulation

Monte Carlo simulation allows us to consider the uncertainty of both sources of uncertainty (i.e., the operational profile and component reliabilities). It is an approximate, but powerful method for estimating reliability of the system when the parameters of the model can be represented by well defined probability distributions. Direct sampling Monte Carlo method consists of the repeated generation of random variables from parameter distributions and their combination according to derived equation for system reliability. Essentially, this is equivalent to constructing many experiments or running many tests on identical systems.

In this paper we derive the reliability expression $R = f(p_{ij}, R_i)$ using the architecture–based software reliability model described in Section 2.3. Then, we assign probability distribution functions to component reliabilities R_i and transition probabilities p_{ij}. These distribution functions can be based on theoretical assumption or on observed data. We assume that component reliabilities are random variables with Beta distribution with pdf given by

$$f(R_i) = \frac{\Gamma(a_i + b_i)}{\Gamma(a_i)\Gamma(b_i)} R_i^{a_i - 1}(1 - R_i)^{b_i - 1} \qquad (11)$$

where $0 \leq R_i \leq 1$. We further assume that the rows in the transition probability matrix are independent and distributed accordingly to Dirichlet distribution. Thus, the joint density for the *i*th row in transition probability matrix has the form

$$f(p_{i1}, p_{i2}, \ldots, p_{in}) = \frac{\Gamma(\alpha_{i1} + \alpha_{i2} + \ldots + \alpha_{in})}{\Gamma(\alpha_{i1})\Gamma(\alpha_{i2}) \ldots \Gamma(\alpha_{in})} \prod_{j=1}^{n} p_{ij}^{\alpha_{ij} - 1} \qquad (12)$$

where $p_{ij} \geq 0$ and $\sum_{j=1}^{n} p_{ij} = 1$.

For the simulation of Dirichlet distribution we use the transformation approach [11] based on the following property [12]. The standard Dirichlet distribution is defined as the distribution of $(Y_1, Y_2, \ldots, Y_n)$ where $Y_k = Z_k / \sum_{j=1}^{n} Z_j$ and $Z_j, j = 1, 2, \ldots, n$ are independent, standard Gamma distributed random variables with shape parameter α_j. With selection of different parameters Dirichlet distribution can take a wide variety of shapes. Therefore, even in the cases where the use of the Dirichlet distribution is not implied by theory, due to its variety of shapes it will be useful as an approximation. Our method, however, is not restricted to Dirichlet distribution. For instance, in some cases it might be assumed that parameters vary by some fixed amount (e.g., 0.1 ± 0.05) and they are uniformly distributed in the interval.

5 Uncertainty analysis applied on case studies

5.1 Case study 1

First, we illustrate the uncertainty analysis on the case study from the European Space Agency (ESA) [7] which consists of almost 10,000 lines of C code. In this case study component traces obtained during testing were used to construct the software architecture and estimate transition probabilities p_{ij}, that is, we use an informed approach. Component reliabilities R_i were estimated using fault injection. Faults reinserted in the code during the experiment are the real faults discovered during integration testing and operational use of the program. Since the program has been extensively used after the last fault removal without failures, this gold version was used as an oracle in the experiment. DTMC that represents software architecture of the European Space Agency software is shown in Figure 2. Components 1, 2, and 3 correspond to the Parser, Computational, and Formatting subsystems respectively. State E represents the completion of execution.

In the experiment, two faulty versions of the program were constructed. Faulty version A consists of fault–free component 3 and faulty components 1 and 2, while faulty version B consists of fault–free components 1 and 3 and faulty component 2. Point estimates for transition probabilities p_{ij} and component reliabilities R_i for both faulty versions are given in Table 1.

Version	p_{12}	p_{23}	R_1	R_2	R_3
A	0.5933	0.7704	0.8428	0.8346	1
B	0.7364	0.6866	1	0.8346	1

Table 1. Parameters for versions A and B

DTMC presented in Figure 3 is a composite state–based software reliability model of this application. The expression

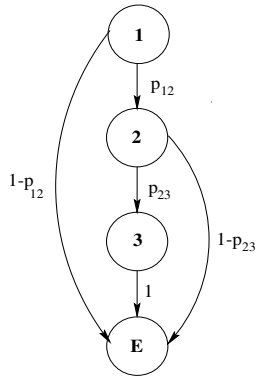

Figure 2. Software architecture for the ESA case study

for the system reliability obtained using the model described in Section 2.3 is given by

$$R = (1-p_{12})R_1 + p_{12}(1-p_{23})R_1R_2 + p_{12}p_{23}R_1R_2R_3. \quad (13)$$

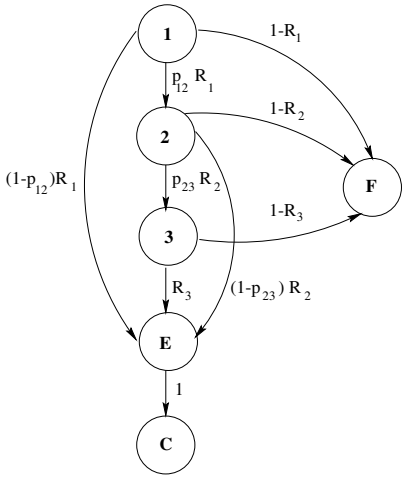

Figure 3. Architecture–based software reliability model for the ESA case study

In [7], we have compared the reliability estimated from the model with the actual reliability. As it can be seen from Table 2, the model gives reasonably accurate estimates compared to the actual reliability for each of the faulty versions, which validates the model and confirms its appropriateness for this case study.

Version	Actual reliability	Estimated reliability	Error
A	0.7393	0.7601	2.81 %
B	0.8782	0.8782	0 %

Table 2. Comparison of the results

In addition to the real case study, we also consider a hypothetical example of software architecture given in Figure 4 which has an additional transition from component 2 to component 1. For the example in Figure 4 the system reliability obtained using the model described in section 2.3 is given by

$$R = \frac{(1-p_{12})R_1 + p_{12}(1-p_{21}-p_{23})R_1R_2 + p_{12}p_{23}R_1R_2R_3}{1 - p_{12}p_{21}R_1R_2}. \quad (14)$$

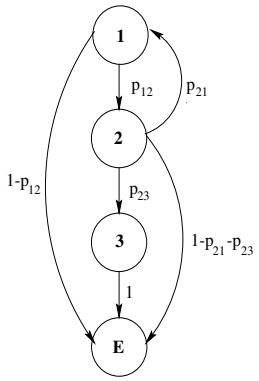

Figure 4. Software architecture for the hypothetical example

5.1.1 Uncertainty analysis of the ESA case study based on method of moments

Next, we illustrate the method of moments on the ESA case study. For the mean values of component reliabilities $E[R_i]$ we use the values of point estimates given in Table 1, while for the variance we assume $\mathrm{Var}[R_i] = 0.0064$. In addition to the mean $E[R]$ and the variance $\mathrm{Var}[R]$ of the system reliability, we estimate the coefficient of variability $C_R = \sqrt{\mathrm{Var}[R]}/E[R]$ which is a relative measure of the spread of the distribution and allows us to compare different distributions. Table 3 compares the values obtained for the mean, variance, and coefficient of variability of the system reliability for versions A and B using first and second order Taylor series expansion. As we already knew from the point estimates, version B has higher mean reliability then version A. The uncertainty analysis provides an additional information about the variance of the system reliability estimate. Thus, the reliability of version B has a smaller variance, that is, the distribution is less spread than the distribution of version A. The smaller value of the variance means that we have a higher confidence in the reliability estimate of version B. As it can be seen from Table 3, the second order ap-

proximation does not improve the accuracy for this example.[1]

		First order	Second order
Version A	Mean	0.7601	0.7601
	Variance	0.0068	0.0068
	C_R	0.1085	0.1085
Version B	Mean	0.8782	0.8782
	Variance	0.0035	0.0035
	C_R	0.0671	0.0671

Table 3. Mean and variance of the system reliability for the ESA case study

In general, higher order Taylor series expansion will increase accuracy, as it can be seen from Table 4 which presents the results obtained for the hypothetical example. In this case we choose two versions, C and D, with different values for the transition probability p_{21} associated with the arc forming a loop in the model (0.25 and 0.75 respectively) and the same values for the other parameters ($p_{12} = 0.8, p_{23} = 0.25, R_1 = 0.8428, R_2 = 0.8346, R_3 = 1$). In view of Table 4 the following observations are made. The mean system reliability decreases for higher values of transition probability p_{21}. Thus, version D is less reliable than version C. In addition, we see that for higher values of p_{21} the coefficient of variability is increasing.

		First order	Second order
Version C	Mean	0.6873	0.6894
	Variance	0.0091	0.0089
	C_R	0.1388	0.1372
Version D	Mean	0.5331	0.5454
	Variance	0.0114	0.0104
	C_R	0.2003	0.1874

Table 4. Mean and variance of the system reliability for the hypothetical example

5.1.2 Uncertainty analysis of the ESA case study based on Monte Carlo simulation

Numerical results presented in this section were obtained using two commercial tools. First, we use *Mathematica* to derive the system reliability expressions $R = f(p_{ij}, R_i)$ in symbolic form. Then, we use *Crystal Ball 2000* to run the simulations. In all cases, Monte Carlo simulation was carried for 10,000 trails.

[1] All second and higher order partial derivates are zero since the system reliability given by equation (13) is a linear function of component reliabilities.

In Figure 5 we present how the uncertainty in parameters of version A affects the system reliability. The estimation of the mean reliability converges in approximately 3000 iterations. In addition to the mean reliability, we have estimated several other characteristics of the system reliability distribution [9]: coefficient of variability C_R which is related to the spread of a distribution, skewness which relates to the lean of a distribution, and kurtosis which related to the peakedness of a distribution. Note that these measures are relative which allows us to compare different distributions.

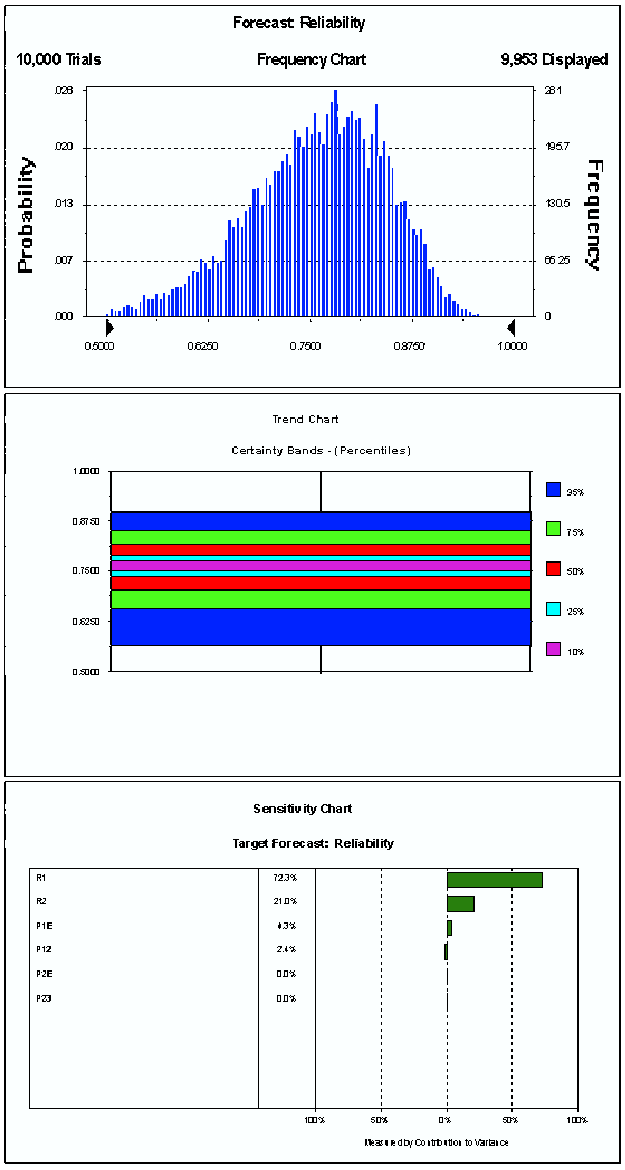

Figure 5. Uncertainty analysis for version A

The frequency chart presented in Figure 5 gives the probability (frequency) of occurrence for different values of system

reliability. The range of the reliability is $[0.3759, 0.9818]$[2] and the distribution is skewed to the left. We have also done a distribution fitting for system reliability. In this case Beta distribution with parameters $\alpha = 17.5014$ and $\beta = 5.3662$ is the closest fit to the frequency data based on the chi–square fitness test. Further, we have estimated the percentiles, i.e., certainty bands. Another interesting observation can be made from the sensitivity chart in Figure 5. We calculate sensitivity by computing rank correlation coefficients between every parameter and system reliability. High correlation coefficient means that the parameter has a significant impact on software reliability (both through its uncertainty and its model sensitivity). Positive coefficients indicate that an increase in the parameter is associated with an increase in the reliability. Negative coefficients imply the reverse situation. In this example, even though the variation of component reliabilities is small, they play critical role in the variation of system reliability. As it can be seen from the sensitivity chart 93.3% of the reliability variation is due to reliabilities R_1 and R_2.

In Figure 6 we present the results for version B obtained by varying transitions probabilities and component reliabilities. The reliability range in this case is $[0.4571, 0.9952]$[3] and the reliability distribution is also skewed to the left. As it can be seen from the values given in Table 5 the reliability distribution of version B has higher mean and smaller variance. Further, it is more skewed to the left (that is, concentrated to the right), with higher peak. Also, certainty bands for version B are narrower than for version A. The system reliability is still more sensitive to the variation of the component reliabilities, although with smaller contribution to the variance (86.2%).

Version	Mean	Coefficient of variability	Skewness	Kurtosis
A	0.7594	0.1126	-0.4781	3.1644
B	0.8798	0.0722	-0.9313	4.0617

Table 5. Characteristics of reliability distributions for versions A and B

Our next numerical example illustrates the uncertainty analysis for the hypothetical example presented in Figure 4. Table 6 compares the characteristics of the system reliability distribution for three different values of transition probability p_{21} associated with the arc forming a loop in the model. As it can be seen from Table 6, the mean system reliability decreases for

[2]For the sake of comparison the displayed range is $[0.5, 1]$

[3]For the sake of comparison the displayed range is $[0.5, 1]$

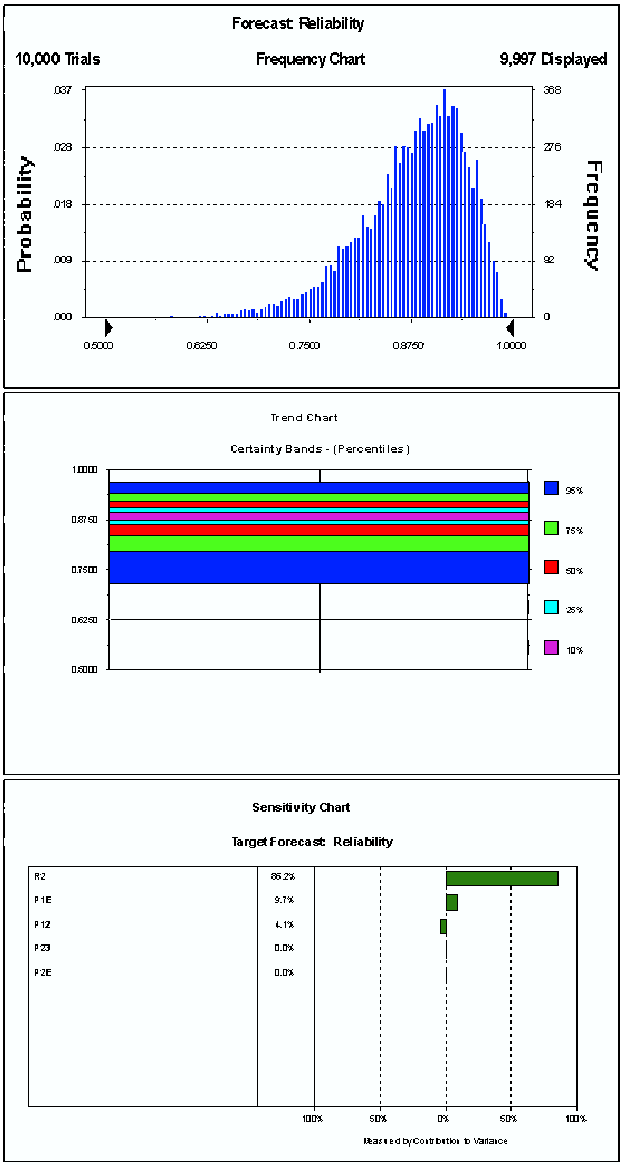

Figure 6. Uncertainty analysis for version B

higher values of transition probability p_{21}. In addition, we see that for higher values of p_{21} the coefficient of variability is increasing, distribution skewness is moving to the right, and the peak is decreasing.

p_{21}	Mean	Coefficient of variability	Skewness	Kurtosis
0	0.7318	0.1225	-0.4458	3.0745
0.5	0.6246	0.1886	-0.2524	2.7510
0.95	0.4109	0.4112	0.1334	2.4294

Table 6. Characteristics of reliability distribution for the hypothetical example

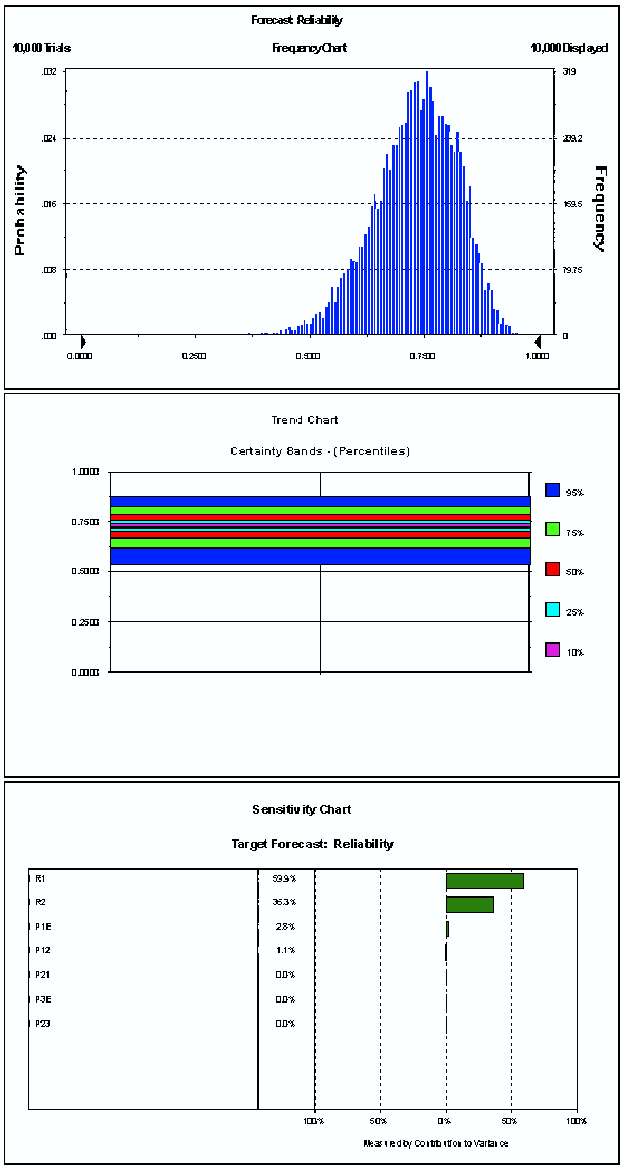

Figure 7. Uncertainty analysis ($p_{21} = 0$)

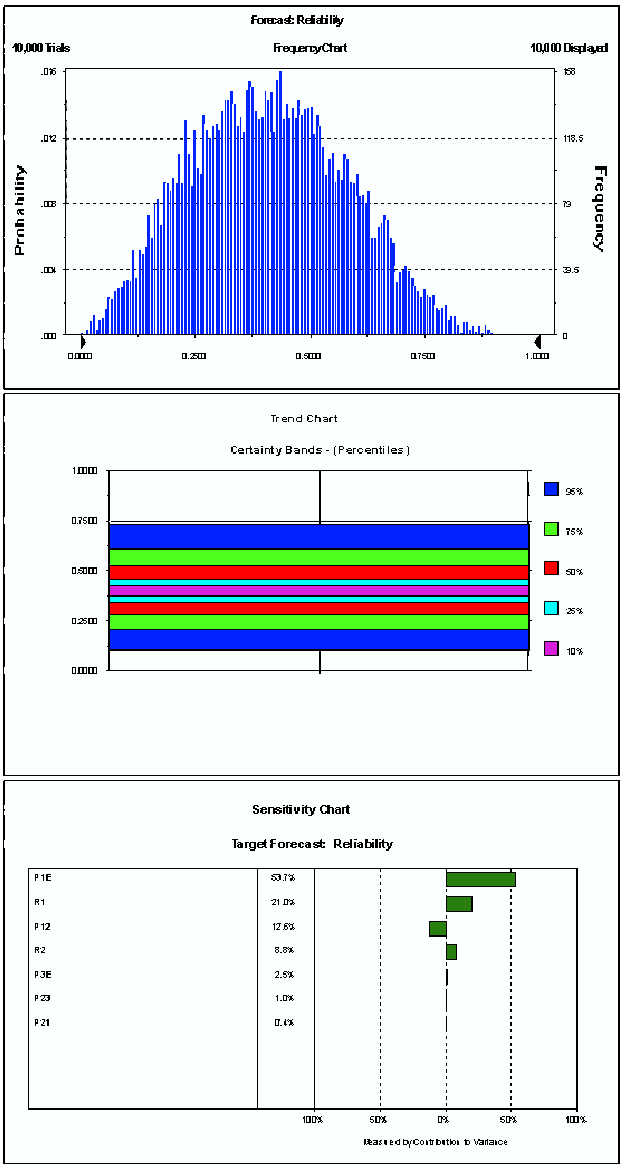

Figure 8. Uncertainty analysis ($p_{21} = 0.95$)

It is obvious from Figures 7 and 8 that the characteristics of the system reliability distribution are very sensitive to the values of modeling parameters. We already knew from the point estimates [7] that the system reliability for $p_{21} = 0.95$ is significantly lower that for $p_{21} = 0$. In addition, from uncertainty analysis we observe that the reliability distribution for $p_{21} = 0.95$ is widely spread and has wider certainty bands compared to $p_{21} = 0$. Also, the parameters contribution to the variance of system reliability changes significantly. Thus, in the case of $p_{21} = 0$ reliabilities R_1 and R_2 contribute 96.2% to the variance of system reliability, while in the case of $p_{21} = 0.95$ they contribute only 29.8%. Even more, when $p_{21} = 0.95$ the highest effect on the system reliability is coming from transition probability $p_{1E} = 1 - p_{12}$ which contributes 53.7% to the variance of system reliability. These results clearly illustrate the usefulness of uncertainty analysis and motivate its systematic use for software reliability prediction.

5.2 Case study 2

In this section, we illustrate the methods for uncertainty analysis on the example adopted from [3]. The application has 10 components and its architecture is described by the DTMC presented in Figure 9. The mean values of non–zero transition

$p_{12} = 0.60$	$p_{13} = 0.20$	$p_{14} = 0.20$		$R_1 = 0.999$
$p_{23} = 0.70$	$p_{25} = 0.30$			$R_2 = 0.980$
$p_{35} = 1.00$				$R_3 = 0.990$
$p_{45} = 0.40$	$p_{46} = 0.60$			$R_4 = 0.970$
$p_{57} = 0.40$	$p_{58} = 0.60$			$R_5 = 0.950$
$p_{63} = 0.30$	$p_{67} = 0.30$	$p_{68} = 0.10$	$p_{69} = 0.30$	$R_6 = 0.995$
$p_{72} = 0.50$	$p_{79} = 0.50$			$R_7 = 0.985$
$p_{84} = 0.25$	$p_{8,10} = 0.75$			$R_8 = 0.950$
$p_{98} = 0.10$	$p_{9,10} = 0.90$			$R_9 = 0.975$
				$R_{10} = 0.985$

Table 7. Mean values of transition probabilities and component reliabilities

probabilities p_{ij} and mean component reliabilities R_i are given in Table 7.

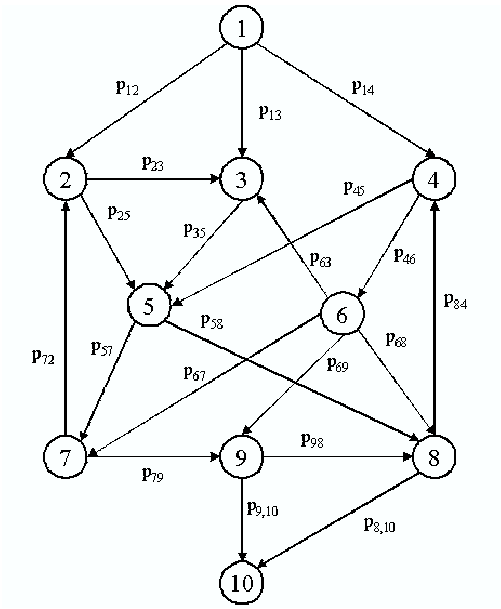

Figure 9. Software architecture for the case study 2

First, we study the uncertainty of software reliability due to uncertainty of components reliabilities. For this purpose, we use the mean values of component reliabilities $E[R_i]$ given in Table 7 and assume variances $Var[R_1] = Var[R_3] = Var[R_6] = 0.0001, Var[R_2] = Var[R_4] = Var[R_5] = Var[R_7] = Var[R_8] = Var[R_9] = Var[R_{10}] = 0.001$. As it can be seen from Table 8, the values of the mean $E[R]$ and the variance $Var[R]$ of the system reliability estimated using the method of moments and Monte Carlo simulation are in close agreement.

Next, we consider both sources of uncertainty (operational profile and components reliabilities) and study their effect on system reliability using Monte Carlo simulation. Frequency chart, certainty bands, and sensitivity chart for the case study 2 are given in Figure 10. The reliability range in this case is $[0.4623, 0.9677]$ with a mean 0.8276, variance 0.0041 and distribution skewed to the left. From the sensitivity chart it can be seen that 17 parameters contribute less than 1% each and 8 parameters contribute between 1% – 5% each to the variance in software reliability. Further, the 4 parameters (out of 29) with the highest sensitivity ranking (R_5, R_8, R_2, R_{10}) contribute 77.2% of the variance in the reliability. Among transition probabilities p_{84} has the highest sensitivity ranking with 4.9% contribution to the variance in software reliability.

	First order	Second order	Monte Carlo
Mean	0.8299	0.8319	0.8304
Variance	0.0036	0.0035	0.0035

Table 8. Mean and variance of the system reliability for case study 2

6 Comparison of the methods for uncertainty analysis

The choice of the method for uncertainty analysis in software reliability that is most appropriate for a given application depends on the criteria such as data requirements, reliability measures derived, and accuracy of the solution. In this section we first summarize in Table 9 the basic characteristics of different methods for uncertainty analysis and then discuss them in detail.

The method for uncertainty analysis based on entropy and theory of Markov chains presented in [13] allows us to characterize the uncertainty in operational profile and software reliability with a single value. Thus, the operational profile with higher entropy value will have exponentially greater number of statistically typical paths. In [13] we have also derived the

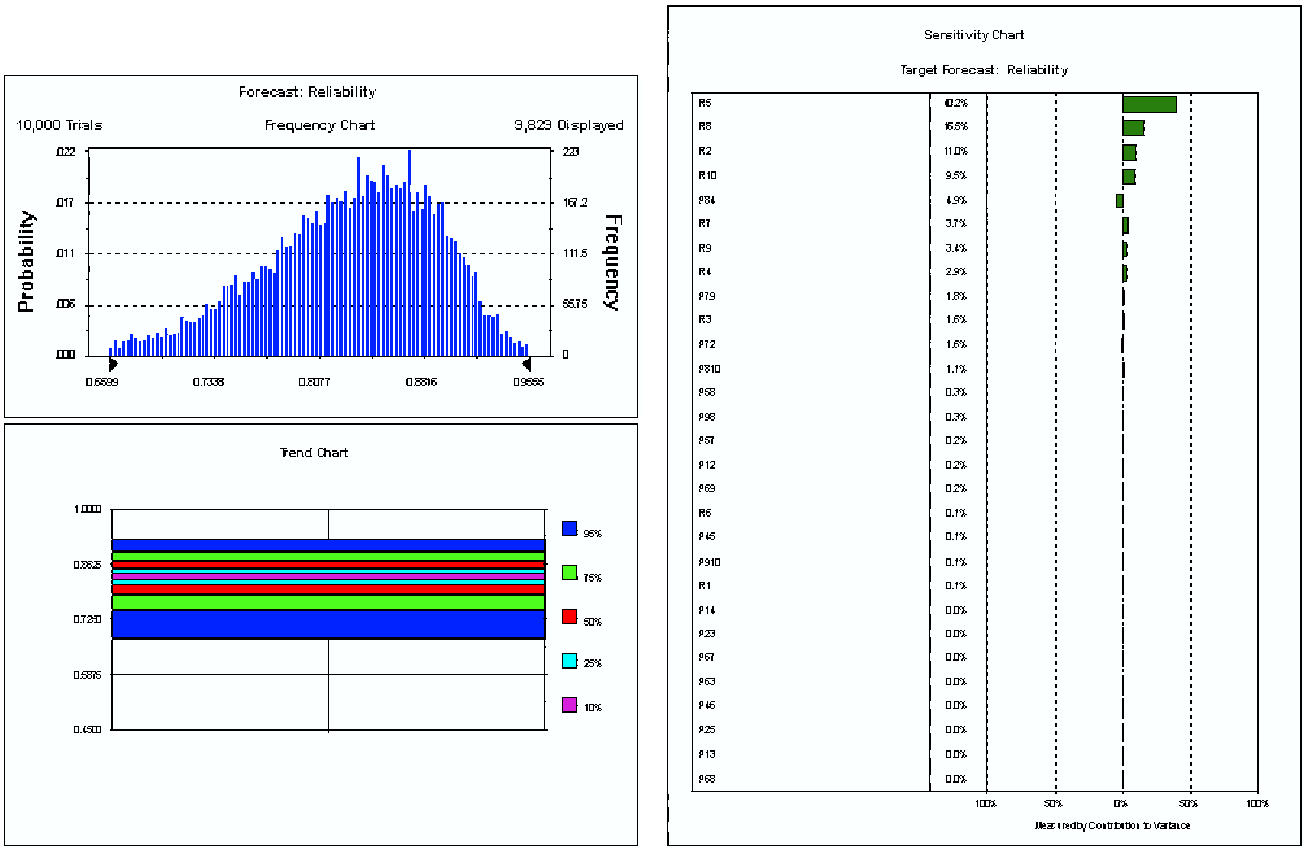

Figure 10. Uncertainty analysis for case study 2

uncertainty and expected execution rate of software components. The limitation of the entropy approach is that it does not provide an estimate of software reliability measures. Thus, we can say that entropy as a method for studying uncertainty is complementary with the method of moments and Monte Carlo simulation.

An advantage of the method of moments over Monte Carlo simulation is the lower data requirements. This method only requires the moments of components reliabilities that may be calculated easily directly from test data (i.e., no distribution must be specified). Further, method of moments is an analytical method and therefore generation of random numbers is not required. A limitation of method of moments is that accuracy may be increased only by including higher order terms in the Taylor series expansion. This is in contrast to the Monte Carlo simulation where, in principle, the accuracy may be arbitrary increased simply by increasing the number of simulations. Also, the accuracy of method of moments is not readily quantifiable. Thus, if a determination of the precise accuracy of an uncertainty analysis is required, the Monte Carlo method should be used. Also, Monte Carlo simulation provides a reacher set of reliability measures such as moments, distribution function, and percentiles of system reliability, as well as sensitivity ranking of the parameters accordingly to the contribution to the variance. Additional advantage of Monte Carlo simulation is that it allows us to quantify uncertainty due to both operational profile and components reliabilities. Note that method of moment may be applied to consider the uncertainty in the operational profile, however, these expressions will be harder to derive due to their complexity.

7 Conclusion

In this paper we have presented a methodology for uncertainty analysis of software reliability that can be applied throughout the software life cycle. Within this methodology, we have used method of moments and Monte Carlo simulation to analyze how the uncertainty of the parameters propagates into the estimates of software reliability. Since the architecture–based approach allows insight into the dynamic behavior of software executions, we have also studied the ef-

Method	Data requirements	Reliability measures	Accuracy of the solution
Entropy [13]	Point estimate	N/A	Exact analytical solution
Method of moments	Moments of components reliabilities	Moments	Approximate method; analytical solution • accuracy may be increased by higher order Taylor series
Monte Carlo simulation	• Distribution functions of transition probabilities and component reliabilities • Generation of random numbers	• Distribution • Moments • Percentiles • Parameters contribution to the variance	Approximate method; simulation • accuracy may be increased by increasing the sample size • sampling errors may be involved in case of long tail distributions

Table 9. Comparison of methods for uncertainty analysis

fect of different parameters on the uncertainty of software reliability. Obviously, the uncertainty analysis provides richer measures of software reliability than the traditional point estimate. These measures can be used for guiding allocation of testing efforts, making quantitative claims about the quality of the software subjected to different operational usages, and for reliability certification of component–based software systems. We believe that the uncertainty analysis of software reliability is not only important but also necessary, especially if we want to make predictions early in the life cycle and keep track of software evolution.

Acknowledgements

This work is funded in part by grant from the NASA OSMA Software Assurance Research Program (SARP) managed through the NASA IV & V Facility, Fairmont, West Virginia and by grant from NASA West Virginia Space Grant Consortium, Research Initiation Grant Program.

References

[1] T. Adams, "Total Variance Approach to Software Reliability Estimation", *IEEE Trans. Software Engineering*, Vol. 22, No.9, 1996, pp.687-688.

[2] M. Chen, A. P. Mathur, and V. J. Rego, "A Case Study to Investigate Sensitivity of Reliability Estimates to Errors in Operational Profile", *5th Int'l Symp. Software Reliability Engineering*, 1994, pp. 276-281.

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

[4] W. Farr, "Software Reliability Modeling Survey", in *Handbook of Software Reliability Engineering*, M. R. Lyu (Ed.), McGraw-Hill, 1996, pp. 71-117.

[5] S. S. Gokhale and K. S. Trivedi, "Reliability Prediction and Sensitivity Analysis Based on Software Architecture", *13th Int'l Symp. Software Reliability Engineering*, 2002, pp. 64-75.

[6] K. Goševa-Popstojanova and K. S. Trivedi, "Architecture–Based Approach to Reliability Assessment of Software System", *Performance Evaluation*, Vol.45, No.2-3, 2001, pp. 179-204.

[7] K. Goševa-Popstojanova, A. P. Mathur, and K. S. Trivedi, "Comparison of Architecture-Based Software Reliability Models", *12th Int'l Symp. on Software Reliability Engineering*, 2001, pp. 22-31.

[8] K. Goševa-Popstojanova and S. Kamavaram, "Uncertainty Analysis of Software Reliability Based on Method of Moments", *13th Int'l Symp. Software Reliability Engineering*, 2002, Fast abstract, pp. 143-144.

[9] G. J. Hahn and S. S. Shapiro, *Statistical Models in Engineering*, John Wiley & Sons, 1994.

[10] P. S. Jackson, R. W. Hockenbury, and M. L. Yeater, "Uncertainty Analysis of System Reliability and Availability

Assessment", *Nuclear Engineering and Design*, Vol.68, 1981, pp. 5-29.

[11] M. E. Johnson, *Multivariate Statistical Simulation*, John Wiley & Sons, 1987.

[12] N. L. Johnson, S. Kotz, *Distributions in Statistics: Continuous Multivariate Distributions*, John Wiley & Sons, 1969.

[13] S. Kamavaram and K. Goševa-Popstojanova, "Entropy as a Measure of Uncertainty in Software Reliability", *13th Int'l Symp. Software Reliability Engineering*, 2002, Student paper, pp. 209-210.

[14] K. Kanoun and T. Sabourin, "Software Dependability of a Telephone Switching System", *17th Int'l Symp. on Fault–Tolerant Computing*, 1987, pp. 236-241.

[15] Y-W Leung, "Software Reliability Allocation under an Uncertain Operational Profile", *Journal of the Operational Research Society*, Vol. 48, 1997, pp. 401-411.

[16] B.Littlewood and D.Wright, "Some Conservative Stopping Rules for Operational Testing of Safety–Critical Software" *IEEE Trans. Software Engineering*, Vol.23, No.11, 1997, pp. 673-683.

[17] K. W. Miller, L. J. Morell, R. E. Noonan, S. K. Park, D. M. Nikol, B. W. Murrill, and J. M. Voas, "Estimating the Probability of Failure when Testing Reveals no Failures", *IEEE Trans. Software Engineering*, Vol.18, No.1, 1992, pp. 33- 43.

[18] J. D. Musa, "Operational Profiles in Software Reliability Engineering", *IEEE Software*, Vol.10, 1993, pp. 14-32.

[19] J. D. Musa, "Sensitivity of Field Failure Intensity to Operational Profile Errors", *5th Int'l Symp. Software Reliability Engineering*, 1994, pp. 334-337.

[20] E. Nelson, "A Statistical Bases for Software Reliability", *TRW-SS-73- 02, TRW Software series*, 1973.

[21] A. Pasquini, A. N. Crespo, and P. Matrella,"Sensitivity of Reliability Growth Models to Operational Profile Errors vs. Testing Accuracy", *IEEE Trans. Reliability*, Vol.45, No.4, 1996, pp. 531-540.

[22] K. Siegrist, "Reliability of System with Markov Transfer of Control", *IEEE Trans. Reliability*, Vol.14 No.7, 1988, pp. 1409-1053.

[23] H. Singh, V. Cortellessa, B. Cukic, E. Guntel, and V. Bharadwaj, "A Bayesian Approach to Reliability Prediction and Assessment of Component Based Systems", *12th Int'l Symp. Software Reliability Engineering*, 2001, pp. 12-21.

[24] J. M. Voas, "Certifying Off-the-shelf Software Components", *IEEE Computer*, Vol.31, No.6, 1998, pp. 53-59.

[25] A. Wesslen, P. Runeson, and B. Regnell, "Assessing the Sensitivity to Usage Profile Changes in Test Planning", *11th Int'l Symp. Software Reliability Engineering*, 2000, pp. 317-326.

[26] J. A. Whittaker and J. H. Poore, "Markov Analysis of Software Specifications", *ACM Trans. Software Engineering and Methodology*, Vol.2, No,1, 1993, pp. 93-106.

[27] S. M. Yacoub, B. Cukic, and H. H. Ammar, "Scenario–Based Reliability Analysis of Component–Based Software", *10th Int'l Symp. Software Reliability Engineering*, 1999, pp. 22-31.

[28] L. Yin, M. A. J. Smith, and K. S. Trivedi, "Uncertainty Analysis in Reliability Modeling", *2001 Annual Reliability and Maintainability Symp.*, 2001, pp. 229-234.

[29] http://www.omg.org/uml/

[30] http://www.gnu.org/manual/gprof-2.9.1/html_mono/gprof.html

[31] http://xsuds.argreenhouse.com