Software Reliability Estimation under Uncertainty: Generalization of the Method of Moments

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

Traditionally, reliability models of component-based software systems compute the point estimate of system reliability by plugging point estimates of unknown parameters into the model. These models discard the uncertainty of the parameters, that is, do not attempt to answer the question how parameters uncertainties affect the estimates of system reliability. In this paper we focus on uncertainty analysis in software reliability based on method of moments. In particular, we present a generalization of our earlier work that allows us to consider the uncertainty in the operational profile (i.e., the way software is used) in addition to the uncertainty in components failure behavior (i.e., component reliabilities) considered earlier. The method of moments is an approximate analytical method that allows us to generate system reliability moments based on (1) the knowledge of software architecture reflected in the expression of system reliability as a function of component reliabilities and frequencies of control transfer between components, (2) estimates of the moments of components reliabilities, and (3) estimates of the moments of probabilities of control transfer between components. Further, we apply the method of moments on two case studies and discuss its advantages and disadvantages.

1 Introduction

Many analytical models for quantification of software reliability have been proposed in the past. One group of models is focused on modeling reliability growth during testing phase [4], [20]. These so called black–box models treat the software as monolithic whole, considering only its interactions with external environment. Black–box models do not consider the internal software structure and therefore are inappropriate for modeling component–based systems. For these systems, we need to use a white–box approach that takes into account the information about the architecture of the software made out of components. An extensive survey on architecture–based software reliability models, including their assumptions, usefulness, and limitations is presented in [6], [7].

Two important questions arise with respect to predications of software reliability based on models. The first question addresses the appropriateness of the model, that is, whether the model assumptions 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. However, in most cases the research work on software reliability discards the uncertainty of the parameters and does not attempt to answer the question how parameters uncertainties affect the estimates of software reliability.

The most common method for uncertainty analysis used in software reliability is conducting sensitivity studies. Thus, sensitivity of software reliability estimates to errors in the operational profile has been investigated in the context of black–box reliability growth models in [2], [17], [19]. Sensitivity studies of software reliability estimates obtained using architecture–based models have been presented in [3], [22]. In these studies the authors assumed fixed known values for the transition probabilities and derived the sensitivity of the system reliability with respect to the reliability of each component. However, any inaccuracy in the operational profile directly will affect transition probabilities among components. The sensitivity studies of software reliability with respect to the operational profile (i.e., transition probabilities) and component reliabilities are presented in [5], [8], [27].

In addition to sensitivity studies, there have been several attempts to quantify the variability of software reliability. In [15] authors used black–box approach and assumed that the failure probability has prior Beta distribution. Using Bayesian approach they derived the mean and the vari-

Proceedings of the Eighth IEEE International Symposium on High Assurance Systems Engineering (HASE'04)
1530-2059/04 $20.00 © 2004 IEEE

ance 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, as well as uncertainty in the probability of using each partition. In [24] the mean, variance, and confidence intervals for the probability of failure per demand were obtained using Bayesian approach with different types of prior information. 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 [21] the mean and the variance of software reliability were estimated for the sample generated by simulation of execution sequences from the usage model. In another related work [13] three optimization models for software reliability allocation under an uncertain operational profile were formulated and solved.

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 [11], [28].

Recently, we proposed 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 [9]. Within this methodology, we developed two methods for uncertainty analysis: method of moments and Monte Carlo simulation. In [9] we used the method of moments to quantify the uncertainty in software reliability due to uncertainty in components reliabilities. Expressions derived in [9] are valid for independent random variables and did not allow us to study the uncertainty in software reliability due to uncertainty in the operational profile.

In this paper we generalize our earlier research work on the method of moments presented in [9]; we derive expressions for the mean and the variance of system reliability that consider both 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). The presented results can be used for keeping track of software evolution throughout the life cycle. Further, they can be used for certification of component–based software systems. This is an important aspect of our work, because with the growing emphasis on reuse developers cannot afford to stay away from reliability certification.

It is important to emphasize that method of moments 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. Providing the mean and the variance of the reliability, instead of the point estimate equal to 1, is particularly important for high consequence and high assurance systems.

The rest of the paper is organized as follows. The basic concepts of the architecture–based software reliability and the motivation for using uncertainty analysis are discussed in Section 2. In Sections 3 we present the method of moments as an approach for uncertainty analysis in software reliability and in Section 4 we illustrate its application on two case studies. The concluding remarks are presented in Section 5.

2 Basic concepts of the architecture–based software reliability

In order to estimate the reliability of a component–based software system we need information on software architecture (structure of component interactions), software usage described by the operational profile (relative frequencies of component interactions determined by transition probabilities), and software failure behavior (component reliabilities or failure rates).

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], [8]. 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 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 $p_{ij} = \Pr\{\text{control is transferred from component } i \text{ to component } j\}$. We construct the Markov chain in two phases. The *structural phase* involves the establishment of the static software architecture. The *dynamic statistical phase* involves the estimation of the relative frequencies of components interactions (i.e., transition probabilities) which define the software usage (i.e., operational profile). Depending on the phase of the software life cycle, different sources of information can be used to build the DTMC that describes the dynamic software architecture. These include historical data from similar products, high level information about software architecture obtained from specification and design documents (such as for example UML use cases and sequence diagrams [10]), or component traces obtained using profilers and test coverage tools [8], [9].

Dynamic information in software architecture represented by transition probabilities clearly depends on the software usage, that is, the operational profile. In general,

operational profile is a quantitative characterization of how a system will be used [16]. In many cases, 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 a 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.

In this work we use a discrete time Markov chain (DTMC) as a model for software architectures and operational profiles. DTMC proves to be a good model for software architectures and operational profiles for several reasons. From the software engineering point of view, the model can be build both in early [10] and late phases [8], [9] of the software life cycle. Once the model has been built, any number of statistically typical test cases can be generated from the model. From the analytical point of view, discrete time Markov chain is a tractable stochastic process with well developed theory, analytical results, and computational algorithms. Furthermore, the model provides basis for building several different architecture–based software reliability models [6], [7].

The next step in building an architecture–based software reliability model 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. In the early phases of software life cycle components reliabilities can be "guestimated" based on expert knowledge or estimated based on historical data [25]. During the testing and operational usage several techniques for estimating components reliabilities can be used. *Software reliability growth models* can be applied to each software component exploiting component's failure data obtained during testing [4], [12]. 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 [14], [15], [18]. In this context, testing is not an activity for discovering faults, but an independent validation activity. 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 techniques* to estimate component's reliability [8], [26]. Fault–based techniques, however, are only as powerful as the range of fault classes that they simulate. From the above it is obvious that, regardless of the technique used, the estimates of component reliabilities may be inaccurate which further motivates the use of uncertainty analysis.

The last step in building an architecture–based software reliability model is to combine the software architecture with components failure behavior. The method of moments can be applied to any architecture–based software reliability model that provides a closed form solution for the software 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, representing the correct output and failure respectively, are added to the DTMC that describes the software architecture. 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 reaching state n from state 1. It follows that the overall system reliability is given by $R = s_{1,n} R_n$.

3 Uncertainty analysis based on method of moments

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). Thus, transition probabilities p_{ij} and component reliabilities R_i are input parameters for software reliability model described in Section 2 which are required to have numerical values so that the software reliability can be computed from the model. Regardless of the appropriateness of the mathematical model used to model software reliability, if considerable uncertainty exists in estimation of the operational pro-

file and component reliabilities (as it usually does) then a significant uncertainty exists in calculated system reliability. Therefore, the traditional approach of computing the point estimate of software reliability by plugging the point estimates of parameters into the model is not appropriate. In order to answer the question how parameters uncertainties propagate into overall system reliability, uncertainty analysis is necessary.

In [9] we used the method of moments to quantify the uncertainty in software reliability due to the uncertainty of components reliabilities. The expressions for the mean and the variance of system reliability derived in [9] are valid for independent random variables and did not allow us to study the uncertainty in software reliability due to the uncertainty in the operational profile. In this paper we generalize our earlier work, that is, we consider the uncertainty of the operational profile, in addition to the uncertainty of component reliabilities. Since transition probabilities p_{ij} out of each state i must sum to 1 (i.e., $\sum_j p_{ij} = 1$) they are not independent random variables. Therefore, in this paper we derive the expressions for the mean and the variance of system reliability considering that the transition probabilities out each state i are correlated random variables.[1]

The method of moments is an approximate analytical approach that allows us to generate the moments of system reliability from the moments of transition probabilities and component reliabilities. The method of moments 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 presented in Section 2 to derive the expression for system reliability

$$R = f(R_i, p_{ij})$$

where $0 \leq R_i \leq 1$, $p_{ij} \geq 0$, and $\sum_{j=1}^{n} p_{ij} = 1$. If we treat each component reliability R_i and transition probability p_{ij} 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). Further, let $E[p_{ij}]$ and $\mu_k[p_{ij}] = E[(p_{ij} - E[p_{ij}])^k]$ denote the mean value and the kth central moment of the transition probability (i.e.,probability of control transfer between components i and j). 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 software architecture expressed by the function $R = f(R_i, p_{ij})$, (2) data on components failures from which estimates of $E[R_i]$ and $\mu_k[R_i]$ for $i = 1, 2, \ldots, n$ can be obtained, and (3) data on software usage from which estimates of the $E[p_{ij}]$ and $\mu_k[p_{ij}]$ for $i, j = 1, 2, \ldots, n$ can be obtained.

The method of moments consists of expanding $R = f(R_i, p_{ij})$ about $(E[R_1], \ldots, E[R_n], E[p_{11}], \ldots, E[p_{nn}])$, the point at which each of the component reliabilities and transition probabilities takes its expected value, by a multivariable Taylor series. Deriving the expression for the 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 the system reliability $R = f(R_i, p_{ij})$ 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

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

where

$$a_0 = f(E[R_1], \ldots, E[R_n], E[p_{11}], \ldots, E[p_{nn}]) \tag{2}$$

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

$$a_{p_{ij}} = \left.\frac{\partial R}{\partial p_{ij}}\right|_{R_i = E[R_i],\, p_{ij} = E[p_{ij}]} \text{ for } \; i,j=1,2,\ldots n. \tag{4}$$

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

$$E[R] \sim a_0 \tag{5}$$

$$\begin{aligned}\text{Var}[R] \sim \sum_{i=1}^{n} a_{R_i}^2 \text{Var}[R_i] + \sum_{k=1}^{n}\sum_{i=1}^{n} a_{p_{ki}}^2 \text{Var}[p_{ki}] \\ +2\sum_{k=1}^{n}\sum_{i=1}^{n}\sum_{j=i+1}^{n} a_{p_{ki}} a_{p_{kj}} \text{Cov}(p_{ki}, p_{kj}).\end{aligned} \tag{6}$$

In [9] we have only considered the uncertainty due to component reliabilities, that is, the expression for Var$[R]$ had only term (the first term in equation (6)). The second and the third term in equation (6) are due to uncertainty in transition probabilities (i.e., operational profile). Assessing the value of the variance of software reliability (in addition to the mean value) 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 Var$[R]$ can be improved by including higher order terms in the Taylor series expansion. We have also derived the expression for the second

[1] Note that we assume that transition probabilities out of different states are independent random variables, that is, rows in the transition probability matrix are independent.

order Taylor series expansion

$$R \sim a_0 + \sum_{i=1}^{n} a_{R_i}(R_i – E[R_i]) + \frac{1}{2}\sum_{i=1}^{n} a_{R_i^2}(R_i – E[R_i])^2 + \sum_{i=1}^{n}\sum_{j=i+1}^{n} a_{R_iR_j}(R_i – E[R_i])(R_j – E[R_j]) + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{p_{ij}}(p_{ij} – E[p_{ij}]) + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{p_{ij}^2}(p_{ij} – E[p_{ij}])^2 + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=j+1}^{n} a_{p_{ij}p_{ik}}(p_{ij} – E[p_{ij}])(p_{ik} – E[p_{ik}]) \quad (7)$$

where a_0, a_{R_i}, and $a_{p_{ij}}$ are given by equations (2), (3), and (4) respectively, and

$$a_{R_i^2} = \left.\frac{\partial^2 R}{\partial R_i^2}\right|_{R_i=E[R_i],\, p_{ij}=E[p_{ij}]} \text{ for } i,j=1,2,\ldots n \quad (8)$$

$$a_{R_iR_j} = \left.\frac{\partial^2 R}{\partial R_i \partial R_j}\right|_{R_i=E[R_i],\, p_{ij}=E[p_{ij}]} \text{ for } i,j=1,2,\ldots n \quad (9)$$

$$a_{p_{ij}^2} = \left.\frac{\partial^2 R}{\partial p_{ij}^2}\right|_{R_i=E[R_i],\, p_{ij}=E[p_{ij}]} \text{ for } i,j=1,2,\ldots n \quad (10)$$

$$a_{p_{ij}p_{ik}} = \left.\frac{\partial^2 R}{\partial p_{ij} \partial p_{ik}}\right|_{R_i=E[R_i],\, p_{ij}=E[p_{ij}]} \text{ for } i,j=1,2,\ldots n \quad (11)$$

Using equation (7) we derive the mean of the system reliability for the second order Taylor approximation, retaining terms up to second order[2]

$$E[R] \sim a_0 + \frac{1}{2}\left[\sum_{i=1}^{n} a_{R_i^2}\mathrm{Var}[R_i] + \sum_{k=1}^{n}\sum_{i=1}^{n} a_{p_{kj}^2}\mathrm{Var}[p_{ki}]\right] + \sum_{k=1}^{n}\sum_{i=1}^{n}\sum_{j=i+1}^{n} a_{p_{ki}p_{kj}}\mathrm{Cov}(p_{ki}, p_{kj}). \quad (12)$$

The equation for the variance of the system reliability, retaining terms up to third order, is given by

$$\begin{aligned}\mathrm{Var}[R] \sim{}& \sum_{i=1}^{n} a_{R_i}^2\mathrm{Var}[R_i] + \sum_{k=1}^{n}\sum_{i=1}^{n} a_{p_{ki}}^2\mathrm{Var}[p_{ki}] \\ &+ 2\sum_{k=1}^{n}\sum_{i=1}^{n}\sum_{j=i+1}^{n} a_{p_{ki}}a_{p_{kj}}\mathrm{Cov}(p_{ki}, p_{kj}) \\ &+ \sum_{i=1}^{n} a_{R_i}a_{R_i^2}E[(R_i – E[R_i])^3] \\ &+ \sum_{k=1}^{n}\sum_{i=1}^{n} a_{p_{ki}}a_{p_{ki}^2}E[(p_{ki} – E[p_{ki}])^3] \\ &+ \sum_{k=1}^{n}\sum_{\substack{i=1\\ i\neq j}}^{n}\sum_{j=1}^{n} a_{p_{ki}}a_{p_{kj}^2}E[(p_{ki} – E[p_{ki}])(p_{kj} – E[p_{kj}])^2] \\ &+ 2\sum_{k=1}^{n}\sum_{\substack{i=1\\ i\neq j}}^{n}\sum_{j=1}^{n} a_{p_{ki}}a_{p_{ki}p_{kj}}E[(p_{ki} – E[p_{ki}])^2(p_{kj} – E[p_{kj}])] \\ &+ 2\sum_{k=1}^{n}\sum_{\substack{i=1\\ i\neq j\neq l}}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n} a_{p_{ki}}a_{p_{kj}p_{kl}}E[(p_{ki} – E[p_{ki}]) \\ &\cdot (p_{kj} – E[p_{kj}])(p_{kl} – E[p_{kl}])]. \end{aligned} \quad (13)$$

Of course, equations for the mean (12) and the variance (13) of system reliability which take into account both sources of uncertainty in software reliability (i.e., component reliabilities and the operational profile) are far more complex that the one we derived in [9].

Note that generating the mean and the variance of system reliability from the second order Taylor series expansion requires the knowledge of the higher order moments of component reliabilities and transition probabilities. Thus, the second order Taylor series approximation provides more accurate estimates for the mean and the variance of software reliability at the price of higher data requirements and more costly computations.

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 moments for component reliabilities and transition probabilities. 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 moments of the component reliabilities will exceed the error introduced by the omission of higher order terms.

The method of moments has several advantages. First, it requires only the knowledge of the moments of components reliabilities and transition probabilities that are estimated from available data, that is, no distribution function

[2] "Retaining terms up to kth order" means that in the derivation all terms whose powers of the expected value sum to k or less are retained and those whose powers sum to more than k are dropped.

must be specified. Second, method of moments can be applied to software systems that do not fail during testing, providing estimates for the mean and the variance of the software reliability (instead of a point estimate equal to 1). Of course, this is very valuable information for high assurance systems. Finally, method of moments is an analytical method and therefore generation of random numbers is not required, that is, there is no sampling error.

However, the method of moments is an approximate method and a finite error is associated with the use of only up to first (second) order terms in the Taylor series expansion. A limitation of the 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 [9].[3]

4 Numerical illustration

4.1 Case study 1

First, we illustrate the application of the method of moments on a case study from the European Space Agency (ESA) [8]. The ESA software consists of almost 10,000 lines of C code. Its architecture is described with a DTMC shown in Figure 1. States 1, 2, and 3 correspond to components 1, 2, and 3, while the state E represents the completion of execution.

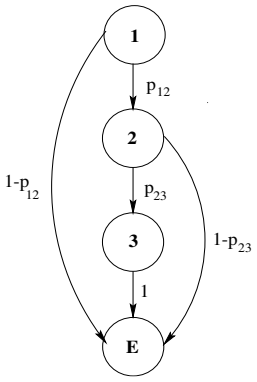

Figure 1. Software architecture for the ESA case study

In the experiment presented in [8], two faulty versions of the program were constructed. Faulty version A consisted of fault–free component 3 and faulty components 1 and 2, while faulty version B consisted of fault–free components 1 and 3 and faulty component 2. Faults reinserted in the code during the experiment were the real faults discovered during integration testing and operational use of the program. The original application which has been extensively used after the last fault removal without failures was used as an oracle. Each faulty version of the program and the oracle were executed on the same test cases generated randomly on the basis of the operational profile. Component traces obtained during testing were used for estimating transition probabilities p_{ij}. When the outputs of the faulty version and the oracle disagreed, the fault responsible for the failure was identified in order to determine which component has failed and estimate component reliabilities R_i. Faults have not been removed and the number of failures includes recurrences due to the same fault.

DTMC presented in Figure 2 is the architecture–based software reliability model of the ESA software. The expression for system reliability obtained using the model described in Section 2 is given by

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

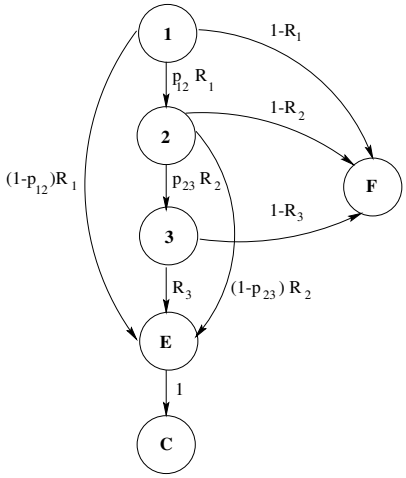

Figure 2. Architecture–based software reliability model for the ESA case study

In [8] we have compared the point estimate of the reliability obtained using equation (14) with the actual reliability of the software. It was shown that the architecture–based software reliability model gives reasonably accurate estimates compared to the actual reliability for each of the faulty versions (i.e., 2.1% error for version A and 0% error for version B), which validates the model appropriateness for this case study.

In addition to the original application, we also consider a hypothetical example of software architecture given in Figure 3 which has an additional transition from component 2 to component 1. This example is meant to illustrate how the components executed within a loop affect the uncertainty of

[3] In order to use Monte Carlo simulation probability distribution functions of component reliabilities and transition probabilities must be specified. Further details on the use of Monte Carlo simulation in software reliability are given in [9].

software reliability. For the example in Figure 3 the system reliability obtained using the model described in section 2 is given by

$$R = \frac{(1-p_{12})R_1 + p_{12}(1-p_{21}-p_{23})R_1R_2 + p_{12}p_{23}R_1R_2R_3}{1 - p_{12}p_{21}R_1R_2}. \quad (15)$$

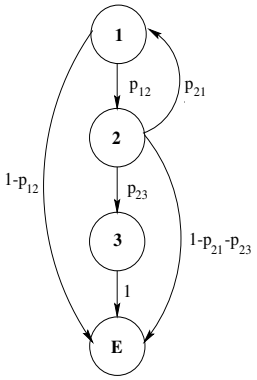

Figure 3. Software architecture for the hypothetical example

Next, we apply the method of moments on the ESA case study. The means and the variances of the transition probabilities and component reliabilities for versions A and B are given in Tables 1 and 2, respectively. In addition to the mean $E[R]$ and the variance Var$[R]$ of the system reliability, we estimate the coefficient of variation $C_R = \sqrt{\text{Var}[R]}/E[R]$ which is a relative measure of the spread of the distribution and allows us to compare different distributions.

		p_{12}	p_{23}
Version A	Mean	0.5933	0.7704
	Variance	0.02974	0.02579
Version B	Mean	0.7364	0.6866
	Variance	0.02452	0.02556

Table 1. Transition probabilities for versions A and B

		R_1	R_2	R_3
Version A	Mean	0.8428	0.8346	0.9995
	Variance	0.00571	0.00568	0.00001
Version B	Mean	0.9995	0.8346	0.9995
	Variance	0.00001	0.00568	0.00001

Table 2. Component reliabilities for versions A and B

Table 3 compares the values obtained for the mean, variance, and coefficient of variation 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 for 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 approximation does not improve the accuracy for this example. This is due to the fact that all second and higher order partial derivates are zero since the system reliability given by equation (14) is a linear function of components reliabilities and transition probabilities.

		First order Taylor series	Second order Taylor series
Version A	Mean	0.7599	0.7599
	Variance	0.0067	0.0067
	C_R	0.1073	0.1073
Version B	Mean	0.8776	0.8776
	Variance	0.0038	0.0038
	C_R	0.0698	0.0698

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

Next we consider the uncertainty analysis 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 ($E[p_{21}] = 0.25$ and Var$[p_{21}] = 0.01831$ for version C and $E[p_{21}] = 0.75$ and Var$[p_{21}] = 0.01831$ for version D) and same values for the other transition probabilities given in Table 4. Component reliabilities for versions C and D are the same as for version A given in Table 2.

	p_{12}	p_{23}
Mean	0.8	0.25
Variance	0.01164	0.01831

Table 4. Transition probabilities for versions C and D

In general, higher order Taylor series expansion increases the accuracy, as it can be seen form Table 5 which presents the results obtained for the hypothetical example. In view of Table 5 we further observe that the mean system reliability decreases for higher values of transition probability p_{21}. In addition, we see that for higher values of p_{21} the coefficient of variation (i.e., the spread of the distribution) is increasing.

Next, we study the parameters contribution to the variance of system reliability. As it can be seen from the Figure 4, in the case of version A 91.28% of the variance is

		First order Taylor series	Second order Taylor series
Version C	Mean	0.6872	0.6861
	Variance	0.0095	0.0094
	C_R	0.1417	0.1411
Version D	Mean	0.5349	0.5290
	Variance	0.0191	0.0167
	C_R	0.2582	0.2441

Table 5. The mean and variance of the system reliability for the hypothetical example

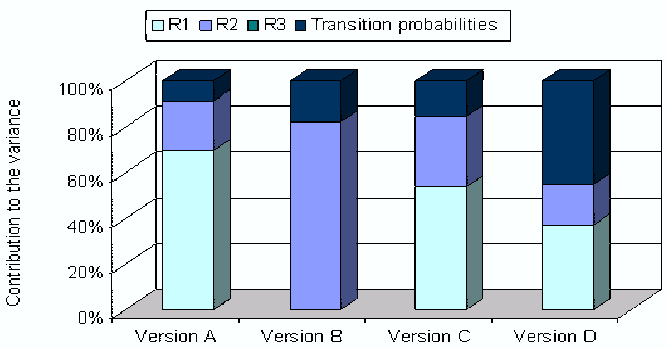

Figure 4. Parameters contribution to the variance of the system reliability for ESA

due to component reliabilities and only 8.72% to transition probabilities. In case of version B the system reliability is still more sensitive to the variation of the component reliabilities, although with smaller contribution to the variance (82.12%). Further, it is obvious that the parameter p_{21} affects significantly the parameters contribution to the variance of system reliability. Thus, in case of version C component reliabilities contribute 84.72% to the variance of system reliability, while in case of version D they contribute 55.09%. These results clearly illustrate the usefulness of the uncertainty analysis and motivate its systematic use for software reliability prediction.

4.2 Case study 2

In this section, we illustrate the method of moments on the example adopted from [3]. The application has 10 components and its architecture is described by the DTMC presented in Figure 5. The mean and variance of non–zero transition probabilities p_{ij} and the mean and variance of component reliabilities R_i are given in Table 6. The architecture–based reliability model for this application is presented in Figure 6.

Table 7 presents the values of the mean $E[R]$ and the variance $\mathrm{Var}[R]$ of the system reliability for the case study 2.

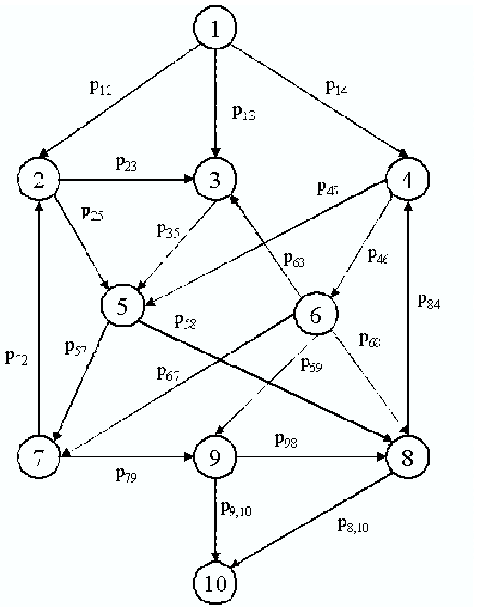

Figure 5. Software architecture for the case study 2

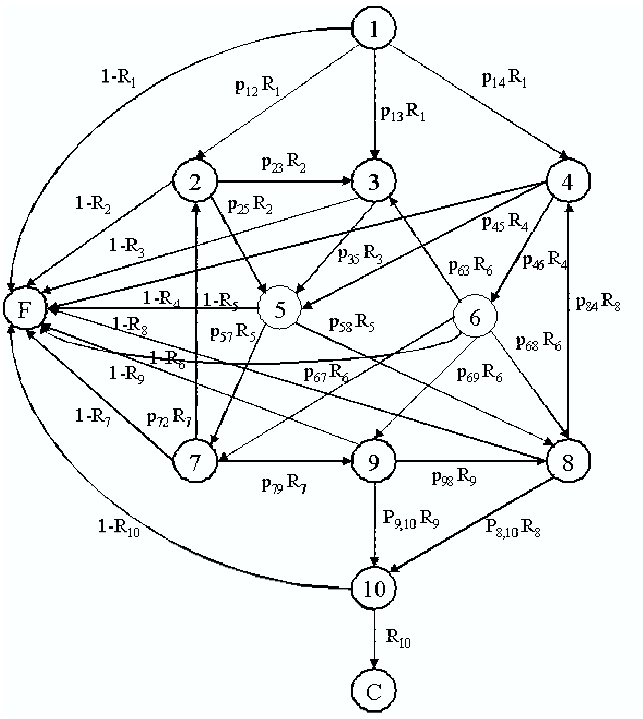

Figure 6. Architecture–based reliability model for the case study 2

$E[p_{12}] = 0.60$ $\text{Var}[p_{12}] = 0.038003$	$E[p_{13}] = 0.20$ $\text{Var}[p_{13}] = 0.025336$	$E[p_{14}] = 0.20$ $\text{Var}[p_{14}] = 0.025336$		$E[R_1] = 0.999$ $\text{Var}[R_1] = 0.000069$
$E[p_{23}] = 0.70$ $\text{Var}[p_{23}] = 0.030849$	$E[p_{25}] = 0.30$ $\text{Var}[p_{25}] = 0.030849$			$E[R_2] = 0.980$ $\text{Var}[R_2] = 0.012078$
$E[p_{35}] = 1.00$ $\text{Var}[p_{35}] = 0$				$E[R_3] = 0.990$ $\text{Var}[R_3] = 0.003850$
$E[p_{45}] = 0.40$ $\text{Var}[p_{45}] = 0.032968$	$E[p_{46}] = 0.60$ $\text{Var}[p_{46}] = 0.032968$			$E[R_4] = 0.970$ $\text{Var}[R_4] = 0.022535$
$E[p_{57}] = 0.40$ $\text{Var}[p_{57}] = 0.031784$	$E[p_{58}] = 0.60$ $\text{Var}[p_{58}] = 0.031784$			$E[R_5] = 0.950$ $\text{Var}[R_5] = 0.000451$
$E[p_{63}] = 0.30$ $\text{Var}[p_{63}] = 0.041874$	$E[p_{67}] = 0.30$ $\text{Var}[p_{67}] = 0.0041874$	$E[p_{68}] = 0.10$ $\text{Var}[p_{68}] = 0.017946$	$E[p_{69}] = 0.30$ $\text{Var}[p_{69}] = 0.041874$	$E[R_6] = 0.995$ $\text{Var}[R_6] = 0.001122$
$E[p_{72}] = 0.50$ $\text{Var}[p_{72}] = 0.038519$	$E[p_{79}] = 0.50$ $\text{Var}[p_{79}] = 0.038519$			$E[R_7] = 0.985$ $\text{Var}[R_7] = 0.007605$
$E[p_{84}] = 0.25$ $\text{Var}[p_{84}] = 0.020327$	$E[p_{8,10}] = 0.75$ $\text{Var}[p_{8,10}] = 0.020327$			$E[R_8] = 0.950$ $\text{Var}[R_8] = 0.000451$
$E[p_{98}] = 0.10$ $\text{Var}[p_{98}] = 0.004088$	$E[p_{9,10}] = 0.90$ $\text{Var}[p_{9,10}] = 0.004088$			$E[R_9] = 0.975$ $\text{Var}[R_9] = 0.017091$
				$E[R_{10}] = 0.985$ $\text{Var}[R_{10}] = 0.007605$

Table 6. Parameter values for the case study 2

	First order Taylor series	Second order Taylor series
Mean	0.8299	0.8323
Variance	0.0213	0.0173
C_R	0.1760	0.1578

Table 7. The mean and variance of the system reliability for the case study 2

We also study the contribution of the parameters to the variance of system reliability. As it can be seen from Figure 7, 53.67% of the variance is due to only two parameters R_{10} and R_2. In this case, transition probabilities contribute only 4.47% to the variance of system reliability, that is, 95.53% of the variance is due to component reliabilities.

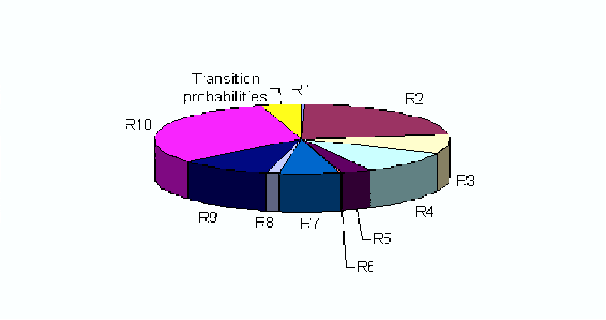

Figure 7. Parameters contribution to the variance of system reliability

5 Conclusion

In this paper we have focused on uncertainty analysis in software reliability and provided generalization of our earlier work on method of moments. In particular, we have derived expressions for the mean and the variance of the system reliability that consider the uncertainty due to operational profile, in addition to the uncertainty due to component reliabilities considered in our earlier work. It is obvious that the estimated values of the system reliability moments provide more information than the traditional point estimate. Thus, we have higher confidence in the reliability estimates for the systems that have reliability with smaller variance. This information is especially useful if we want to make predictions early in the life cycle, keep track of software evolution, and certify the reliability of component–based systems. Another important contribution of this paper is that, instead of the point estimate of the reliability equal to 1, it enables us to quantify the mean and the variance of the reliability of component–based systems that do not fail during the testing. This result is particularly important for high assurance systems.

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. Gokhale and K. 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 and K. S. Trivedi, "Architecture Based Approaches to Software Reliability Prediction", *International Journal Computers & Mathematics with Applications*, Vol.46, 2003, pp. 1023-1036.

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

[9] K. Goševa–Popstojanova and S. Kamavaram, "Assessing Uncertainty in Reliablity of Component-Based Software Systems", *14th Int'l Symp. Software Reliability Engineering*, Nov. 2003, pp. 307-320.

[10] K. Goševa–Popstojanova, A. Hassan, A. Guedem, W. Abdelmoez, D. Nassar, H. Ammar, and A. Mili, "Architectural-Level Risk Analysis using UML", *IEEE Trans. on Software Engineering*, Vol.29, No.10, 2003, pp. 946-960.

[11] 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.

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

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

[14] 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.

[15] 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.

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

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

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

[19] 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.

[20] C. V. Ramamoorthy and F. B. Bastani, "Software Reliability – Status and Perspectives" *IEEE Trans. on Software Engineering*, Vol.8, No.4, 1982, pp. 354-371.

[21] K.Sayre and J.Poore, "A Reliability Estimator for Model Based Software Testing", *13th Int'l Symp. Software Reliability Engineering*, 2002, pp. 53-63.

[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] C. Smidts, D. Sova, and G. K. Mandela, "An Architectural Model for Software Reliability Quantification", *8th Int'l Symp. Software Reliability Engineering*, 1997, pp. 324-335.

[25] C. Smidts and D. Sova, "An Architectural Model for Software Reliability Quantification", *Reliability Engineering and System Safety*, Vol.64, 1999, pp.279-290.

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

[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 Symposium*, 2001, pp. 229-234.