Architecture-Based Reliability Evaluation under Uncertainty

Indika Meedeniya, Irene Moser, and Aldeida Aleti
Faculty of Information and Communication Technologies, Swinburne University of Technology
Hawthorn, VIC 3122, Australia
{imeedeniya,imoser,aaleti}@swin.edu.au

Lars Grunske
Faculty of Computer Science & Center for Mathematical and Computational Modelling (CM)^2
University of Kaiserslautern, D-67653, Germany
grunske@cs.uni-kl.de

ABSTRACT

The accuracy of architecture-based reliability evaluations depends on a number of parameters that need to be estimated, such as environmental factors or system usage. Researchers have tackled this problem by including uncertainties in architecture evaluation models and solving them analytically and with simulations. The usual assumption is that the input parameter distributions are normal, and that it is sufficient to report the attributes that describe the system in terms of the mean and variance of the attribute. In this work, we introduce a simulation-based approach that can accommodate a diverse set of parameter range distributions, self-regulate the number of architecture evaluations to the desired significance level and reports the desired percentiles of the values which ultimately characterise a specific quality attribute of the system. We include a case study which illustrates the flexibility of this approach using the evaluation of system reliability.

Categories and Subject Descriptors

D.2.11 [**Software Architectures**]; C.4 [**Performance of Systems**]; D.2.4 [**Software/Program Verification**]: Reliability

General Terms

Reliability, Design

Keywords

Software architecture evaluation, Reliability, Monte Carlo simulation, Uncertainty analysis

1. INTRODUCTION

Architecture-based quality evaluation models are an important asset during design of software intensive embedded systems. The benefit of these evaluation models is especially evident in the architectural design phase, since different design alternatives can be evaluated and software architects are able to make informed decisions between these alternatives. To date, a number of evaluation models have been proposed for evaluating specific quality attributes such as performance [3, 5], reliability [15] and safety [22]. However, for a considerable number of parameters the architecture evaluations are based on estimates. These estimations tend to use field data obtained during testing or operational usage, historical data from products with similar functionality, or reasonable guesses by the domain experts. In practice, however, parameters can rarely be estimated accurately [2, 13, 21]. In this paper, we investigate different aspects in relation to the parameter uncertainties in architecture based quality evaluation having specific focus on reliability, and formulate a framework that constitutes specification, evaluation and quantification of probabilistic quality attributes in the presence of uncertainty. In the context of software-intensive systems design, the sources of uncertainty can be classified into two major categories.

Aleatory uncertainty *is the inherent variation associated with the physical system or environment under consideration [30].* This category refers to the sources that are inherently stochastic in nature. Physical uncertainties such as noise in electrical conductors, humidity, temperature, material parameters, behaviour and instantaneous decisions of operators are examples in the domain of embedded systems. This type of uncertainty can not be avoided [6].

Epistemic uncertainty *is uncertainty of the outcome due to the lack of knowledge or information in any phase or activity of the modelling process [30].* This source of uncertainty reflects the lack of knowledge on the exact behaviour of the system. Uncertainties of this type are subjective and depend on factors such as maturity of the design and models, experience of the application domain, and the coverage and extent of testing.

Manifestations of the above two types of uncertainty exist in the parameters of software architecture such as software components, inter-component interactions, hardware components, communication links, behavioural distributions, operational profile and use cases. In this paper, we address the problem of architecture-based quality evaluation of probabilistic properties, when the external and probabilistic model parameters are subjected to uncertainty. We focus on probabilistic quality attributes and the probabilistic models that are used to obtain the quantitative metrics from the architecture. The accuracy of the architecture evaluation models

QoSA+ISARCS'11, June 20–24, 2011, Boulder, Colorado, USA.

and the goals are not questioned, and considered out of the scope of this study.

Related Work. In the context of software architecture evaluation under uncertainty, a considerable amount of work can be found in the area of sensitivity analysis with respect to the parameters of probabilistic quality models. Most of the approaches to date have concentrated on specific quality attributes. In the area of architecture-based reliability evaluation Cheung [9] presented a sensitivity analysis method for his original reliability model with a composite Discrete-Time Markov Chain (DTMC) abstraction. The method is purely analytical and consists of a number of 2^{nd} and 3^{rd} order partial derivatives of system reliabilities which are hard to estimate in real cases. Goševa-Popstojanova et al. [21] proposed the *method of moments* to calculate the sensitivity of a system's reliability to component reliabilities and transition probabilities analytically. Cortellessa et al. [12] discussed the significance of error propagation to other parts of the system. Their sensitivity analysis can help identify the most critical system components. Coit et al. [37, 10] have used means and variances of reliability estimates of software components to analytically derive the mean and variance of the reliability of a redundancy allocation. With the assumption of normal distributions for input, Finodella et al. [13] derived the distribution of system reliability from a multinomial distribution. Coit et al. [11] presented an analytical approach to obtain the lowerbound percentile of the reliability in series-parallel systems whereas similar analytical approach can be seen in evaluation of reliability bounds [7]. All of the above methods have taken an analytical approach to quantify the sensitivity, where the applicability is limited to analytically solvable models. However, these analytical sensitivity analysis methods are hard to generalise. Furthermore, all the discussed approaches assume the parameter distributions are normal and variations can be characterised by the mean and variance alone.

Goševa-Popstojanova et al. [20, 19] have shown that analytical methods of uncertainty analysis do not scale well, and proposed a Monte Carlo (MC) simulation based method. With the help of experimental validation they demonstrated that the MC methods scale better than the *method of moments* approach [17]. Similarly, Marseguerra et al. [26] have used mean and variance estimates of component reliabilities to obtain the mean and variance of the system reliability using MC simulation. These approaches have extended the applicability of uncertainty analysis to the analytically solvable models, assuming the applicability of specific reliability models and input distributions. Yin et al. [38] has proposed a DTMC simulation-based approach to derive system reliability from the probability distributions of component reliabilities under the assumption that component and system reliabilities are gamma-distributed. Axelsson [2] has also highlighted the significance of MC based approaches in cost evaluation with uncertainty.

However, the existing MC-simulation-based uncertainty analysis approaches are based on the assumption that there is a specific continuous input distribution and that the resulting sample distribution is normal or Weibull. However, practical experience in connection with some studies [25, 14] show that the actual distributions are hard to determine. Mean-and-variance-based quality evaluations are not sufficient for architecture-based decision making in the case of safety-and-mission-critical systems.

Contribution and Overview of the Paper. In this paper we introduce a new Monte-Carlo-based architecture evaluation method, which allows heterogeneous and diverse uncertainties as they naturally occur in software architecture evaluation models [2, 31, 32, 38]. Figure 1 illustrates the elements of the novel approach and their relationships. The leftmost element represents the specification of the software components, hardware, usage profile and quality requirements. We introduce the ability to incorporate heterogeneous information about the uncertainty of parameters at the specification phase. Model-based quality evaluations are used to determine the quality of the prospective system on the basis of the architecture. Based on the results of multiple Monte Carlo (MC) simulations, estimates for the quality attributes of the architectures are computed. A novel dynamic stopping criterion stops the MC simulations when sufficient samples have been taken.

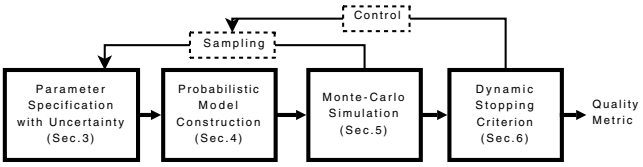


Figure 1: Architecture evaluation under uncertainty

2. EXAMPLE APPLICATION

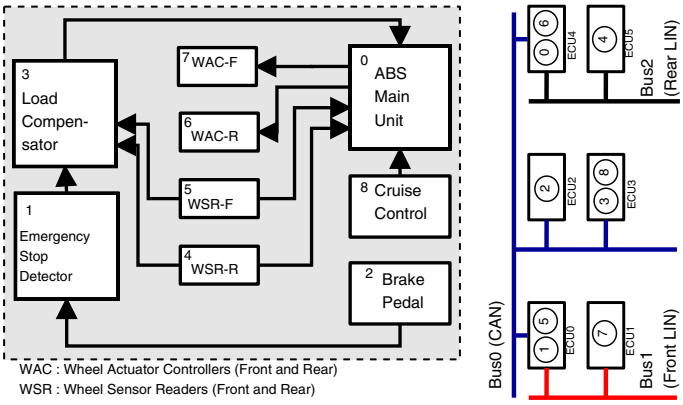
The example of a deployment architecture which assigns of software components to a hardware infrastructure of electronic control units (ECUs) is used to illustrate the concepts introduced in this paper. The software components belong to the *Anti-lock Brake System (ABS)* of a car. ABS is a system that maintains traction during braking manoeuvres to prevent skidding. It is therefore a crucial safety features found in most of contemporary cars. The system and its parameters are briefly described in this section, and further details can be found in [28].

2.1 Software Components and Interactions

The software components in this example are treated as black boxes [23], i.e. a description of the externally visible parameters such as is available, internal structures are unknown and not modifiable. The components interact to implement a set of services, defining the functional units accessed by the user of the system. The ABS activity is initiated in one software component (with a given probability) which may employ many auxiliary components it is connected to via communication links. The process and logical views of the subsystems are depicted in Figure 2a. The *ABS Main Unit* is the major decision making unit regarding the braking levels for individual wheels, while the *Load Compensator* unit assists with computing adjustment factors from the wheel load sensor inputs. Components 4 to 7 represent transceiver software components associated with each wheel, and communicate with sensors and brake actuators. *Brake Pedal* is the software component that reads from the paddle sensor and sends the data to the *Emergency Stop Detection* software module.

Software component parameters

(a)Workload (wl): computational load of a software component in executing a requested task; expressed in MI (million instructions).



(a) SW Architecture (b) Deployment

Figure 2: Software components and their deployment to HW topology in the ABS system

(b)Execution initiation probability (q_0)*:* the probability of the program execution starting at this component.

Interaction parameters

specified for a link from component $\mathcal{C}_i$ to $\mathcal{C}_j$.

(a)Data size (ds): the amount of data transmitted from software component $\mathcal{C}_i$ to $\mathcal{C}_j$ during a single communication event; expressed in KB (kilobytes).

(b)Next-step probability (p): the probability that a service calls component $\mathcal{C}_j$ after component $\mathcal{C}_i$.

2.2 Hardware Topology and Deployment

The hardware model used to deploy the software components is comprised of a distributed set of certified ECUs having different capacities of memory, processing power, access to sensors, etc. ECUs communicate through buses. Many types of buses with different data rates and reliability can be present. The ECUs and the bus system that compose the hardware architecture for the system is depicted in Figure 2b. In this example, we consider one of the feasible deployments of software components to the hardware architecture. The numbers in Figure 2b refer to the allocated software components with corresponding ids in Figure 2a.

ECU parameters

(a)Processing speed (ps): the instruction-processing capacity of the ECU; expressed in MIPS (million instructions per second). This is used to calculate the execution time, which is a function of processing speed of the ECU and the computation workload of the service.

(b)Failure rate (fr): failure rate (of the exponential distribution [8, 1]) that characterises the probability of a single ECU failure.

Bus parameters

(a)Data rate (dr): the data transmission rate of the bus; expressed in KBPS (kilobytes per second). This is used to calculate the latency in for data transmission, as it is a function of the data rate of the bus and the amount of data transmitted during the communication. *(b)Failure rate (fr):* failure rate of the exponential distribution characterising data communication failure of each bus.

2.3 Objectives

The problem in focus is to evaluate the reliability of the ABS function from the deployment architecture. With respect to the deployment, two sources of failure are considered for reliability evaluation which have been defined in [28].

Distribution	Specification Syntax	Range	Example
Normal	NORMAL,μ,σ^2	$(-\infty, \infty)$	$NORMAL, 3.75, 0.05$
Beta	BETA, α, β	[0,1]	$BETA, 10, 2$
Shifted Beta	BETA_SHD,$\underline{x}$, $\overline{x}$, α, β	$(\underline{x}, \overline{x})$	$BETA\_SHD, 3, 5, 2, 10$
Exponential	EXP,λ	$(0, \infty)$	$EXP, 7.5 \times 10^{-6}$
Uniform	UNIFORM,$\underline{x}$, $\overline{x}$	$(\underline{x}, \overline{x})$	$UNIFORM, 3.5, 4.0$
Gamma	GAMMA,λ	$(0, \infty)$	$GAMMA, 1.5$
Weibull	WEIBULL,α	$(0, \infty)$	$WEIBULL, 1.5$
Discrete	DISCRETE, $x_0, p_0, x_1, p_1, ..., x_n, p_n$	$(x_0,\ x_n)$	$DISCRETE,$ $2, 0.4, 2.1, 0.5, 2.3, 0.1$

Table 1: Probability distributions and their specification

Execution failures: Failures may occur in the ECUs during execution of a software component, which affects the reliability of the software modules running on that ECU. For this illustration, we assume a fixed deterministic scheduling of tasks with in the ECU. It is also assumed that the failure that happens in an ECU while a software component is executing or queued leads to a service execution failure.

Communication failures: Failure of a data communication bus when a software component communicates with another one over the bus, directly impacts a failure in the service that depends on this communication.

The annotations, model and evaluation of the reliability are presented in later subsections.

3. SPECIFICATION OF UNCERTAIN PARAMETERS

Even when there is uncertainty in the parameter space, not all the parameters have necessarily probabilistic values. Often there are considerable dissimilarities between parameters whose values cannot be definitively determined. Since it has been established that the variability of parameter estimates significantly affects the quality metric of the architecture [17, 4, 34], it is important to capture the variability characteristics as accurately as possible. Given these considerations, we comprehend architecture parameters as a mix of precise and imprecise sets.

3.1 Probability Distributions

As a means to capture heterogeneous uncertainties in parameter estimation, we propose to use generalised probability distributions. A parameter in an architecture specification is considered as a random variable, whose variability is characterised by its – continuous or discrete – distribution. For the parameter specifications in the architecture descriptions we propose a generic notation that can cater for any distribution. The specification is given as a parameter list, starting with a unique identifier assigned to the distribution. Some examples for Probability Density Function (PDF) specifications are given at Table 1.

3.2 Mapping Uncertainty into PDFs

The proposed approach allows to combine diverse sources that affect the nominal value of the parameter, and consider their impact on the quality evaluation in addition to the conventional point estimates. Some guidelines to obtain the PDFs at the design stage can be given as follows.

• *Derive from the source variations.* The uncertainty of pa-

rameters are often manifestations of different sources. Information from hardware manufactures, third party software vendors or system experts is useful in characterising the uncertainty in specific parameters. In some situations, the distribution of the source variables can be obtained and consequently, the desired parameter's distribution can be approximated from its sources.

Example ▷ The failure rate (λ) of an ECU is a function of its ambient temperature (T in Kelvin) [8], such as $\lambda = 4 \cdot 10^{-6} \times T + 100$. Consider an automotive electronic system where the temperature profile around ECU X varies between 300K and 400K, has a 370K mode and is skewed right. The PDF of λ of ECU X can be derived and specified as $\lambda_X = BETA\_SHD, 400 \times 4 \cdot 10^{-6}, 500 \times 4 \cdot 10^{-6}, 10, 2$ ◁

• *Histogram approximation* Prior information on the parameters may be available. For certain parameters, large numbers of raw data may be available as a result of testing. In such situations, the PDFs can be approximated from the histograms of the raw data.

Example ▷ In functional test executions of the system model, the histogram of the test results indicated that the message transfer probability from component C_i to component C_j is normally distributed. The average of the samples is 0.2 with a variance of 0.04. Therefore, the transfer probability can be given as $p_{i,j} = NORMAL, 0.2, 0.04$ ◁

• *Uniform approximation* It is common to have limited information on the range of the variation without any specification on variation within the range. Uniform distributions can be used in approximating such situations.

Example ▷ The system has a need to communicate with a new external service X, of which we only know that its worst case response time is 1.0s. The communication link takes at least 5ms for the data transfer. $rt = UNIFORM, 5 \cdot 10^{-3}, 1.0$ ◁

• *Specify distinct information as a discrete-sample distribution* : In cases where the a parameter can only vary within a discrete set, discrete-sample distributions can be used to capture it. This is a very powerful feature in our approach as in most of the practical situations, it is relatively easy to obtain discrete estimates.

Example ▷ Experts have indicated that the request rate (rr) for a service X can be either 200 or 800 per second. In 75% of the cases it is 200. This will be given as $rr = DISCRETE, 200, 0.75, 800, 0.25$ ◁

3.3 Illustration Using the Example

Not every parameter pertaining to the current example is subject to uncertainty. For instance, the processing speed(ps) of an ECU or the computational load(wl) of a software component can realistically be considered fixed and deterministic. However, parameters such as the failure rates of ECUs, failure rates of buses, execution initiation probabilities and transition probabilities are subject to uncertainty and have to be estimated. Table 2 shows an example set of parameters.

The probabilistic specification of parameters in the tables reflect the variability of these parameters in relation to the automotive ABS system. It is realistic to assume different distributions for the same parameter in different problem instances [1, 8, 18, 28]. The sources of these values may be different in each instance. Hence, within the same column, we may have mentioned different PDFs as well as distinct values.

Comp. ID	wl (MI)	q_0
0	1.2	0
1	0.6	0
2	0.4	$DISCRETE, 0.03, 0.2, 0.3, 0.4, 1.5, 0.2, 3, 0.2$
3	1	0
4	0.4	$NORMAL, 0.3, 0.075$
5	0.4	$NORMAL, 0.3, 0.075$
6	0.4	0
7	0.4	0
8	0	$DISCRETE, 0.01, 0.2, 0.1, 0.4, 0.5, 0.2, 1, 0.2$

(a) Software Components

Trans $c_i \rightarrow c_j$	$p(c_i, c_j)$	ds (KB)
$0 \rightarrow 6$	$DISCRETE, 0.05, 0.2, 0.5, 0.4, 2.5, 0.2, 5, 0.2$	2
$0 \rightarrow 7$	$DISCRETE, 0.05, 0.2, 0.5, 0.4, 2.5, 0.2, 5, 0.2$	2
$1 \rightarrow 3$	1	2
$2 \rightarrow 1$	1	2
$3 \rightarrow 0$	1	2
$4 \rightarrow 0$	$GAMMA, 0.7$	1
$4 \rightarrow 3$	$GAMMA, 0.3$	2
$5 \rightarrow 0$	$GAMMA, 0.7$	1
$5 \rightarrow 3$	$GAMMA, 0.3$	2
$8 \rightarrow 0$	1	0

(b) Component Interactions

ECU ID	ps (MIPS)	fr (h^{-1})
1	40	$BETA\_SFT, 4 \cdot 10^{-5}, 4 \cdot 10^{-3}, 10, 2$
2	22	$BETA\_SFT, 4 \cdot 10^{-5}, 4 \cdot 10^{-3}, 10, 2$
3	22	$DISCRETE, 2 \cdot 10^{-6}, 0.2, 2 \cdot 10^{-5}, 0.4, 1 \cdot 10^{-4}, 0.2, 2 \cdot 10^{-4}, 0.2$
4	22	$DISCRETE, 1 \cdot 10^{-5}, 0.2, 1 \cdot 10^{-4}, 0.4, 5 \cdot 10^{-4}, 0.2, 1 \cdot 10^{-3}, 0.2$
5	110	$NORMAL, 8 \cdot 10^{-4}, 0.04$
6	110	$NORMAL, 2 \cdot 10^{-4}, 0.01$

(c) ECUs

BUS ID	dr (KBPS)	fr (h^{-1})
0	128	$BETA\_SFT, 3 \cdot 10^{-6}, 3 \cdot 10^{-4}, 10, 2$
1	64	$BETA\_SFT, 1.2 \cdot 10^{-5}, 1.2 \cdot 10^{-3}, 10, 2$
2	64	$BETA\_SFT, 4 \cdot 10^{-6}, 4 \cdot 10^{-4}, 10, 2$

(d) Buses

Table 2: Parameter specification of software and hardware elements of the architecture

4. PROBABILISTIC MODEL CONSTRUCTION

4.1 Propagation of Uncertainty in Models

The specification of architectural elements as discussed above requires a new model for quality evaluation. Different quality attributes can be evaluated using different modelling approaches as discussed in related work section in the introduction. In the case of probabilistic quality attributes, the evaluation models are also probabilistic. The model parameters are often derived from the parameters of the architectural elements. This process results in one-to-one, one-to-many, many-to-one or many-to-many relationships of architecture parameters to the parameters of probabilistic model. As we have incorporated probabilistic specification of parameters of architectural elements, the probabilistic notion is transformed to the evaluation model parameters. Due to the fact that the inputs are probability distributions, the resulting evaluation model parameters become probability distributions or functions of probability distributions.

4.2 Illustration Using the Example

In order to obtain a quantitative estimation of the reliability of the automotive architecture in focus, a well-established DTMC-based reliability evaluation model [15, 16, 35] is used. An absorbing DTMC [35] is constructed for each subsystem

from the software components and hardware specification, such that a node represents the execution of a component and arcs denote the transfer of execution from one component to the other. A super-initial node [36] is added to represent the execution start, and arcs are added from that node annotated with relevant execution initialisation probabilities(q_0). Figure 3 shows the DTMC for the example case. The node labels point to the corresponding nodes in Figure 2a. The failure rates of the execution elements can

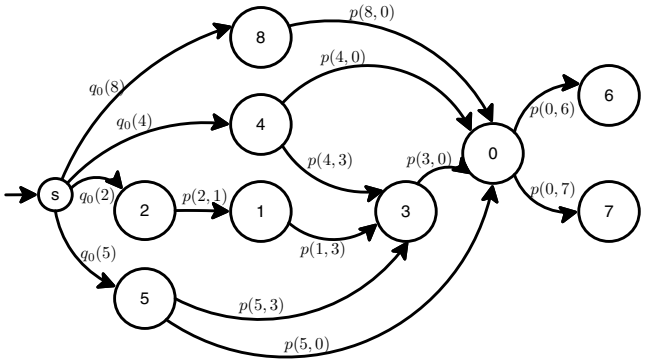


Figure 3: Annotated DTMC for service reliability evaluation

be obtained from the ECU parameters. The execution time is defined as a function of the software–component workload and the processing speed of its ECU. Similar to the models used in [1, 27], the reliability of the ABS system considers both ECU and communication link failures. In detail, the reliability of a component c_i can be computed as:

$$R_i = e^{-fr(d(c_i))\cdot\frac{wl(c_i)}{ps(d(c_i))}} \tag{1}$$

where $d(c_i)$ denotes the ECU allocation relationship of component c_i. A similar computation can be employed for the reliability of communication elements [27], which, in our model, are characterised by the failure rates of hardware buses, and the time taken for communication, defined as a function of the buses data rates dr and data sizes ds required for software communication. Therefore, the reliability of the communication between component c_i and c_j is defined as:

$$R_{ij} = e^{-fr(d(c_i),d(c_j))\cdot\frac{ds(c_i,c_j)}{dr(d(c_i),d(c_j))}} \tag{2}$$

The expected number of visits of a DTMC node $v_i : C \to \mathbb{R}_{\geq 0}$, quantifies the expectation of use of a component (or subsystem) during a single system execution. This can be computed by solving the following set of simultaneous equations [24, 16]:

$$v(c_i) = q_0(c_i) + \sum_{j\in\mathcal{I}}(v(c_j)\cdot p(c_j,c_i)) \tag{3}$$

The following expansion of the formula (3) can be used to transform the equation in matrix form:

$$\begin{aligned}
v(c_0) &= q_0(c_0) + v(c_0)\cdot p(c_0,c_0) + v(c_1)\cdot p(c_1,c_0) + ... + v(c_n)\cdot p(c_n,c_0)\\
v(c_1) &= q_0(c_1) + v(c_0)\cdot p(c_0,c_1) + v(c_1)\cdot p(c_1,c_1) + ... + v(c_n)\cdot p(c_n,c_1)\\
v(c_2) &= q_0(c_2) + v(c_0)\cdot p(c_0,c_2) + v(c_1)\cdot p(c_1,c_2) + ... + v(c_n)\cdot p(c_n,c_2)\\
&\vdots\\
v(c_n) &= q_0(c_n) + v(c_0)\cdot p(c_0,c_1) + v(c_1)\cdot p(c_1,c_n) + ... + v(c_n)\cdot p(c_n,c_n)
\end{aligned}$$

In matrix form, the transfer probabilities $p(c_i, c_j)$ can be written as $P_{n\times n}$, and the execution initiation probabilities $q_0(c_i)$ as $Q_{n\times 1}$. The matrix of expected number of visits $V_{n\times 1}$ can be expressed as:

$$V = Q + P^T \cdot V \tag{4}$$

With the usual matrix operations, the above can be transformed into the solution format:

$$I \times V - P^T \times V = Q \tag{5}$$

$$(I - P^T) \times V = Q \tag{6}$$

$$V = (I - P^T)^{-1} \times Q \tag{7}$$

For absorbing DTMCs, a term that applies to the model used in this illustration, it has been proved that the inverse matrix $(I - P^T)^{-1}$ exists [35].

The expected number of visits of a communication link, $v(l_{ij}) : C \times C \to \mathbb{R}_{\geq 0}$, quantifies the expected number of occurrences of the transition (c_i, c_j) for each link $l_{ij} = (c_i, c_j)$. To obtain this value, we have extend the work of Kubat et al. [24] for computing the expected frequency of system component access to communication links. In the extension, we understand communication links as first-class elements of the model, and view each probabilistic transition $c_i \xrightarrow{p(c_i,c_j)} c_j$ in the model as a tuple of transitions $c_i \xrightarrow{p(c_i,l_{ij})} l_{ij} \xrightarrow{1} c_j$, the first adopting the original probability and the second having probability $= 1$. Then we can apply the above, and compute the expected number of visits of a communication link as:

$$\begin{aligned}
v(l_{ij}) &= 0 + \sum_{x\in\{X\}}(v(c_x)\cdot p(c_x,l_{ij}))\\
&= v(c_x)\cdot p(c_x,c_j)
\end{aligned} \tag{8}$$

where X is the subset of the indexes of components using link l_{ij} in the deployment under scrutiny.

Based on the relationships obtained in equations (1) and (2), the reliability of the deployment is calculated as follows:

$$R \approx \prod_{i\in\mathcal{I}} R_i^{v(c_i)} \cdot \prod_{i,j\in\mathcal{I}} R_{ij}^{v(l_{ij})} \tag{9}$$

Some of the entries for the parameters in Table 2 $p_{i,j}$, fr_i, $fr_{i,j}$, q_{0_i} are probability distributions. These parameters form part of the final reliability calculation in equation (9). The matrix formula (7) contains entries of heterogeneous PDFs for p_{ij} and q_{0_i}. Consequently, the model evaluation results of vector V of formula (7) becomes probabilistic, and cannot be solved with numerical matrix operations. Furthermore, R_i and R_{ij} in formulations (1) and (2) are influenced by the probabilistic specifications of fr_i and $fr_{i,j}$ in Table 2. The propagation of parameter uncertainties to the system reliability R can be further observed in the use of probabilistic V in combination with uncertain R_i and R_{ij} in (9).

5. QUALITY METRIC ESTIMATION

The probabilistic model with partially uncertain parameter space has to be evaluated in order to obtain the quantitative metric of the quality of the system architecture at hand. It has been emphasised before that these models are often hard to represent as linear mathematical functions. When many parameters are uncertain with diverse distributions, the quantitative metric as a distribution cannot be derived

analytically. Consequently, the Monte-Carlo(MC)-based approach presented here draws samples from the probability distributions of input parameters.

Figure 4 illustrates the architecture evaluation process using MC simulation. The input of the MC simulation of the probabilistic model (PM) is a set of parameters specified as probability distributions (UPs) as well as deterministic/certain parameters (DPs).

5.1 Monte Carlo Simulation

The MC simulation takes samples from input probability distributions of the architectural elements within the probabilistic evaluation model. Any one parameter of an architectural element may contribute to more than one parameter in the evaluation model. Every time a sample is taken from input distribution, all model parameters dependent on this parameter have to be updated. The resulting strategy for a single run of the MC simulation is explained in the following.
1. **Sample:** A sample is taken from the Cumulative Distribution Function (CDF) of each parameter. *Inverse transformation method* [33] can be used for this process.
2. **Update:** Using the samples drawn from the input distributions, the numerical values for the evaluation model parameters are updated. Since more than one parameter of the probabilistic model may be dependent on a parameter in the architecture, a Publisher-Subscriber paradigm can be used. Whenever a sample is taken for specific architectural parameter, all the subscribing model parameters are updated and recomputed.
3. **Resolve dependencies:** The model specific parameter dependencies are solved in this phase. The numerical values for the outgoing transition probabilities are normalised to comply with the model assumptions.
4. **Compute:** Analytically solve/simulate the model and obtain the quantitative metric of the system quality.

The single run of MC simulation results in one numerical value of the quality attribute (a). Due to the probabilistic inputs (UPs), the values obtained from different runs ($\{a_1, a_2, a_3, .., a_N\} = A$) are most often not identical. From the software architect's point of view, a single statistical index of a quality metric ($\hat{a}$) is desirable despite the uncertainty. The level of tolerance in the statistical estimation depends on the application domain and the quality attribute. Depending on the character of the system to be designed, the expected value, variance, quartiles, confidence intervals and worst case values have been used to describe the quality of a system.

One important challenge regarding this estimation is that the actual distribution of the quality metric (A) is unknown. The existing uncertainty analysis techniques in software architecture evaluation have a prior assumption on the distribution of the (A). With some exceptions [11, 38], most studies assume normal distribution [10, 21]. Due to the heterogeneity of input parameter uncertainty, and the non-linearity and complexity of model evaluation techniques, the resulting quality distribution after MC simulation is unpredictable. For our approach, we introduce a generalised estimation of ($\hat{a}$), using the flexible percentiles while supporting the expected/worst case measures.

5.2 Distribution estimation

From the statistical data ($A = \{a_1, a_2, a_3, ..., a_N\}$) in the MC runs, statistical methods can be used to identify parameters of a candidate distribution. Possible approaches are the method of maximum likelihood and the method of moments, as well as Bayesian estimation [29]. These methods can be applied when prior information about the distribution of the resulting quality metric (A) is available. Due to the diverse nature of input parameter distributions and the complexity of the quality evaluation models, estimating prior distribution is hard and computationally expensive, since it would have to be repeated for each architecture evaluation.

5.3 Non-parametric estimation

Non-parametric estimation , has the major advantage of not requiring any assumptions about the probability distribution of the population (A). Non-parametric methods lend themselves to providing a generic estimation for flexible percentile estimates ($\hat{A}$).

Instantaneous objective values for each MC run ($A = a_1, a_2, a_3, ..., a_N$) are stored and sorted into ascending or descending order. Percentile estimates can be obtained from retrieving the correct position in the array.

Example ▷ Assume the quantitative predictions reliability of an architecture X for each MC run (A) have been inserted to a ascending-sorted array $S = s_1, s_2, s_3, ..., s_N$. The 95^{th} percentile of reliability for architecture X is easily obtained calculating index $i = N * 95/100$ of the required entry. ◁

6. DYNAMIC STOPPING CRITERION

All of the estimation techniques discussed above sample from appropriate distributions and obtain a desired statistical index of a quality attribute $\hat{a}$. However, the accuracy of the estimate $\hat{a}$ strongly depends on the sample size, i.e. on the number of MC trials carried out. One important characteristic of the problem is that the actual value of the estimate ($\hat{a}$) or the distribution of A is not known. Large numbers of MC runs cannot be conducted because given the large number of candidate architectures produced in stochastic optimisation, the computational expense is prohibitive.

6.1 Sequential Statistical Significance Test

To solve this issue, we propose a dynamic stopping criterion based on accuracy monitoring. In this approach, the assumptions on the monitored distribution (A) are relaxed by transforming the monitoring phase to the estimate $\hat{A}$. A statistical significance test is carried out on the samples of the statistical index ($\hat{A}$).

- A minimum of k MC executions ($a_1, .., a_k$) are conducted before estimating the desired index $\hat{A}$. After k repeats, one of the methods discussed in Section 5.3 can be used to obtain each $\hat{a}$.
- The variation of the estimate $\hat{A} = \{\hat{a}_1, \hat{a}_2, \hat{a}_3, ..., \hat{a}_k\}$ is monitored for a sliding window of size k. Only the last k samples of the estimate $\hat{A}$ are monitored, as the accuracy of the estimation is a changing property. The objective is to detect if sufficient accuracy is obtained.
- The statistical significance is calculated for the last k estimates [33]:

$$w_r = \frac{2z_{(1-\alpha/2)}}{\sqrt{k}} \frac{\sqrt{\overline{\hat{a}^2} - \left(\overline{\hat{a}}\right)^2}}{\overline{\hat{a}}} \tag{10}$$

where : w_r is the relative error, $\overline{\hat{a}}$ is the average of last k estimates, $\overline{\hat{a}^2}$ is the mean-square of the last k estimates,

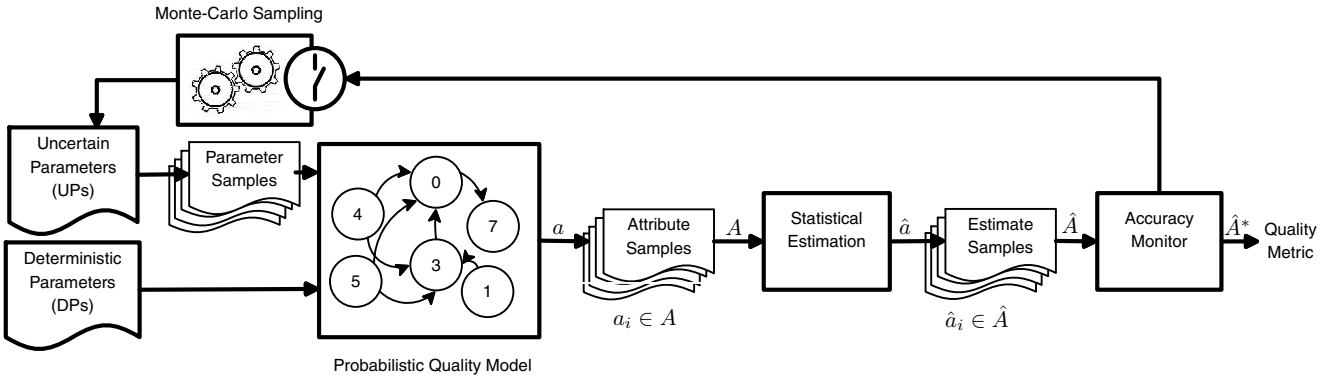


Figure 4: Monte Carlo simulation

α is the desired significance of the test and z refers to the inverse cumulative density value of the standard normal distribution. The relative error w_r of the estimate $\hat{A}$ is checked against a tolerance level, e.g. 0.05. The complete algorithm is explained in Algorithm 1.

Algorithm 1: Monte Carlo simulation with dynamic stopping criterion

```
1  i = 1, j = 1;
2  while w_r > tolerable w_r do
3      a_i := conduct one MC execution();
4      if i ≥ k then
5          â_j := non-parametric estimate using
           (a_1, a_2, a_3, ..., a_i);
6          if j ≥ k then
7              ā = (Σ_{p=j-k}^{j} â_p) / k;
8              â²‾ = (Σ_{p=j-k}^{N} â_p^2) / k;
9              w_r = (2 z_{(1-α/2)} / √k) · √(â²‾ − (ā)²) / ā;
10         end
11         j++;
12     end
13     i++;
14 end
15 â* = â_j;
```

It should be noted that the parameters of the above algorithm, epoch size (k) and significance (α) can be set independently from the architecture evaluation problem.

6.2 Illustration Using The Example

The above algorithm can be applied on the running example as the following:

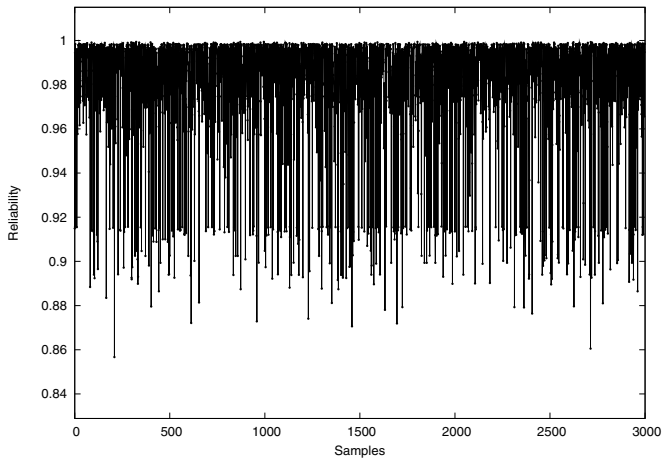
1. The reliability model is constructed from the architecture specification. The parameters are sampled according to the specifications in Table 2.
2. Model-specific parameter dependencies are resolved. In the DTMC-based reliability model, model parameters are normalised to satisfy the following properties.
   - The sum of all outgoing transitions (p_{ij}'s) from any non-absorbing node should be equal to 1.
   - The sum of execution initialisation probabilities(q_0's) should be equal to 1.
3. These values are subjected to the equation (3) and used to solve the matrix formulae (7). Then the expected visits for the links are calculated using equation (8).
4. From the sampled failure rates of ECUs and buses, reliabilities can be calculated by using formulae (1) and (2). Consequently, the system reliability estimate for the samples obtained in the step (1) can be obtained applying the equation 9. This process represents a single MC run, which yields a reliability value $a_i \in A$.
5. The steps (1) to (4) are repeated $k = 10$ times, after which the accuracy estimation process starts. Assuming a goal of 90% significance, for k samples, the initial estimate of reliability ($\hat{a_1}$) is computed using non-parametric percentile estimation(5^{th} percentile) described in Section 5.3.
All existing samples are used to estimate $\hat{a}$ after each MC run. When k number of estimates $\hat{a}$ are available, the dynamic stopping criterion in equation 10 is applied.
6. If w_r is less than a threshold, the last $\hat{a}$ is considered as the 5^{th} percentile reliability of the architecture. Otherwise, the process repeats from step 1. When the stopping criterion is reached, the final estimate of $\hat{a}$ is taken as the quality metric $\hat{a}^* \in \hat{A}^*$.

7. EXPERIMENTS

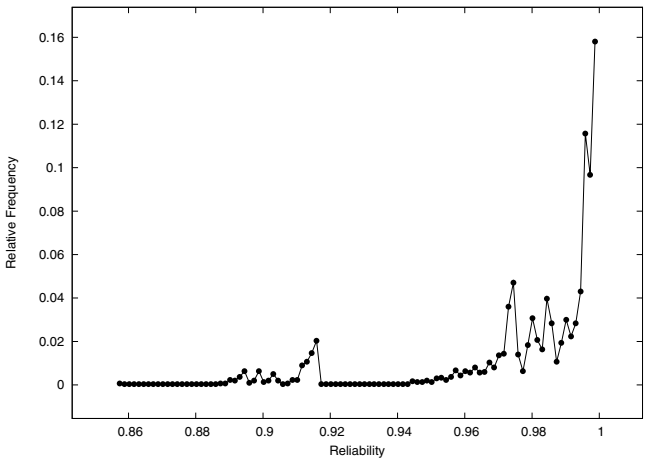
7.1 Experiments on the Example

The MC simulation process has been explored using the case study example. The results of 3000 MC trials are presented in Figure 5. The samples for each MC run, taken as described in step 4 in Section 6.2, are scattered as depicted in Figure 5a. It can be seen that the values for the reliability vary from 0.85 to 0.999, which is a significant variation with respect to the notion of reliability. The histogram of the reliability obtained from 3000 samples in Figure 5b shows that the conventional assumption of a normal or Weibull distribution of the system reliability is incompatible with the actual characteristics of the sample distribution obtained from MC simulation.

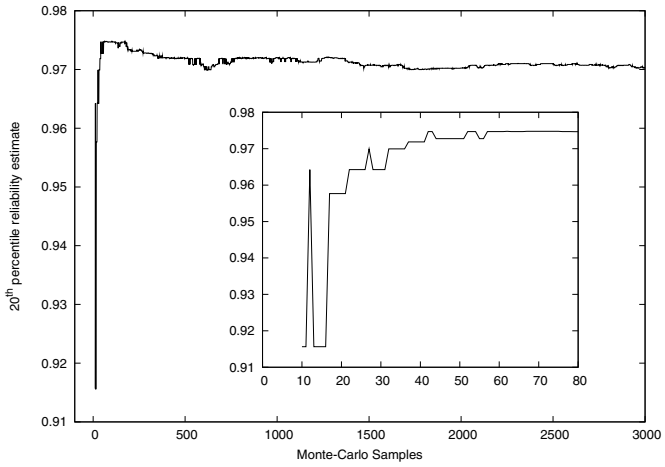
The goal of the MC run was set to the 20^{th} percentile, i.e. the reliability of the system will be greater than the estimate for 80% of the cases. The epoch size k was set to 10. The values of each estimation is presented in Figure 5c (note that the graph starts at sample size of 10). Considerable variations can be observed at the early estimations, while an observable stabilisation is achieved after around 100 MC runs. These variations are reflected in the error values (Figure 5d). The stopping criterion uses a significance test configuration of $\alpha = 0.05$ and $w_r = 0.0005$, which leads to the MC simulation achieving the required accuracy



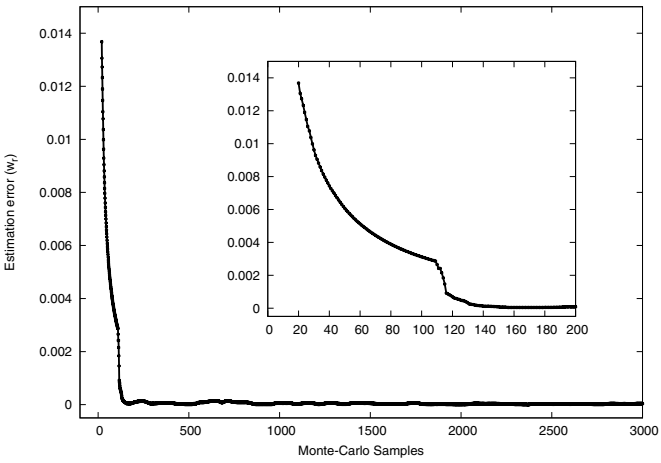
(a) Instantanious samples of reliability in MC trials



(b) Histogram of reliability samples



(c) Variation of the accuracy of the estimation with MC runs



(d) Relative error of sequential accuracy monitoring

Figure 5: Results of the experiments with ABS system example

at 156 runs, with the estimate for 20^{th} percentile reliability of 0.974527.

7.2 Experiments on Generated Problems

7.2.1 Experiment setup

A series of experiments have been conducted to investigate the presented architecture evaluation approach under uncertainty. The scalability of the approach is explored by generating instances from a range of problem sizes. The objective of each problem is estimation of reliability.

• To represent uncertainties in component reliability figures, each component's reliability is specified as a random distributions in the range $0.99 - 0.9999$. For a single problem, the distributions remain unchanged throughout the experiments to maintain the comparability of the results.

• Additional distributions are introduced to represent uncertainties associated with other parameters. The number of additional uncertainties are varied from 0 to 10 for each problem size in order to investigate the level or uncertainty in different instances. Parameters with uncertainty draw their values from Normal, Uniform, Beta, Shifted Beta, Gamma, Exponential, Weibull and Discrete distributions.

• In order to represent diversity in architecture to model relationship, the DTMC is constructed using random relationships between components. Therefore, a parameter may have an effect on randomly selected transition probabilities in the generated DTMC.

• The support for different levels of compromise in the estimation process is captured by optimising each problem instance for median, 25^{th} percentile (75% pessimistic), 5^{th} percentile (95% pessimistic) of the reliability. Dynamic stopping criteria is set to $\alpha = 0.05$, $w_r = 0.005$ and $k = 10$.

The configurations for the problem instances are given in Table 3.

7.2.2 Results

Table 4 lists the results for the expected value, 25^{th} percentile (75% pessimistic), 5^{th} percentile (95% pessimistic) of the reliability using the 16 problem instances and 3 classes of tolerance described in Table 3. N in the table refers to the MC runs that first satisfied the stopping criterion. The estimation of the quality at the stopping condition is listed in the columns $\hat{a}^*$.

The MC simulations are carried out for a large number of runs (10000), even after the stopping criterion has been met. The final estimation a_f obtained from 10000 runs is compared with the estimation achieved at the stopping condition. The column d_r indicates the relative difference be-

Case ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DTMC Nodes	10	10	10	10	20	20	20	20	50	50	50	50	100	100	100	100
Additional Uncertainties	0	2	5	10	0	2	5	10	0	2	5	10	0	2	5	10

Table 3: Experiment configurations

tween $\hat{a}^*$ and a_f calculated as:

$$d_r = \left(\frac{\hat{a}^* - a_f}{a_f} \right)^2 \quad (11)$$

In all cases, the relative difference d_r is less than the relative error w_r calculated by Equation 10. The results show that the approach is applicable to different sizes of the problem as well as diverse levels of uncertainty. The accuracy of the MC simulations comply with the specified tolerance levels. It should be noted that the novel stopping criterion controls the process with the effect of saving large number of computational resources. For example in 5^{th} percentile estimations, many cases have achieved sufficient accuracy with a small number of MC runs, while cases like 12 are automatically continued for longer to obtain an accurate estimation.

7.3 Discussion of The Results

Internal validity. The validity of the approach has been illustrated with respect to the example case study as well as with a series of random experiments. The new framework's capability of handling a diverse range of probability distributions has been validated with the experiments using random distributions. In the experiments, the DTMC-based reliability evaluation model takes on a stochastic relationship to the architecture. It has been shown that the new approach can successfully evaluate a number of very diverse problem instances, indicating versatile applicability. The percentiles estimated by the MC-simulator have been chosen to cover moderate and more pessimistic requirements with the quality attributes in practise. The novel dynamic stopping criterion has been tested for the 16 random cases and for three different percentile estimations, and accuracy of the tests has been validated under the specified tolerance levels. The experiments have been generated to represent a maximum possible degree of randomness.

External validity. It should be noted that all the experiments in this paper explore the model of a single system attribute, reliability. Validity against the spectrum of models and architecture parameters cannot be claimed without further experiments. The framework presented in this paper is deliberately generic and treats the probabilistic evaluation model as a *black box*, avoiding a dependency on the internal complexity of the model. We suggest that the contribution presented can be applied to any architecture-based evaluation model, even though it has been validated only with regards to a specific reliability model.

8. CONCLUSIONS

In this paper, we have addressed the high-level problem of evaluating reliability based on software architectures in the presence of uncertainty. The evaluation framework introduced in this work provides support for heterogeneous software architecture parameters. Probabilistic parameter values and their evaluation have been accommodated through the use of an MC simulation. A nonparametric significance test as a stopping criterion has significantly reduced the number of trial runs and function evaluations necessary to achieve the desired confidence level. The functionality of the framework has been illustrated using a practical case study. In our future work, we aim to investigate further on general applicability of the approach over the other probabilistic properties that are evaluated based on the architecture. Furthermore, we are currently working on integrating the uncertainty analysis with design space exploration, towards robust architecture optimisation.

Acknowledgement. This original research was proudly supported by the Commonwealth of Australia, through the Cooperative Research Center for Advanced Automotive Technology (projects C4-501: Safe and Reliable Integration and Deployment Architectures for Automotive Software Systems). Furthermore, the research was supported by the Center for Mathematical and Computational Modelling $(CM)^2$ at the University of Kaiserslautern.

9. REFERENCES

[1] I. Assayad, A. Girault, and H. Kalla. A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints. In *Dependable Systems and Networks*, pages 347–356. IEEE, 2004.

[2] J. Axelsson. Cost Models with Explicit Uncertainties for Electronic Architecture Trade-off and Risk Analysis. *Current Practice*, 2006.

[3] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni. Model-based performance prediction in software development: A survey. *Software Engineering, IEEE Transactions on*, 30(5):295–310, 2004.

[4] M. Basseur and E. Zitzler. A preliminary study on handling uncertainty in indicator-based multiobjective optimization. In *Appl. of Evol. Computing*, pages 727–739. Springer, 2006.

[5] S. Becker, L. Grunske, R. Mirandola, and S. Overhage. Performance prediction of component-based systems - a survey from an engineering perspective. In *Architecting Systems with Trustworthy Components*, volume 3938 of *LNCS*, pages 169–192. Springer, 2006.

[6] H. Beyer and B. Sendhoff. Robust optimization - A comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196(33-34):3190–3218, 2007.

[7] A. Bhunia, L. Sahoo, and D. Roy. Reliability stochastic optimization for a series system with interval component reliability via genetic algorithm. *Applied Mathematics and Computation*, 216(3):929–939, 2010.

[8] A. Birolini. *Reliability engineering: theory and practice*. Springer-Verlag, 2010.

[9] R. Cheung. A user-oriented software reliability model. *Software Engineering, IEEE Transactions on*, 6(2):118–125, 1980.

[10] D. Coit, T. Jin, and N. Wattanapongsakorn. System Optimization With Component Reliability Estimation Uncertainty: A Multi-Criteria Approach. *IEEE Transactions on Reliability*, 53(3):369–380, 2004.

[11] D. W. Coit and A. E. Smith. Genetic algorithm to maximize a lower-bound for system time-to-failure with uncertain component Weibull parameters. *Computers & Industrial Engineering*, 41, 2002.

[12] V. Cortellessa and V. Grassi. A modeling approach to analyze the impact of error propagation on reliability of component-based systems. In *Component-Based Software Engineering*, pages 140–156. Springer-Verlag, 2007.

[13] L. Fiondella and S. S. Gokhale. Software reliability with architectural uncertainties. In *Parallel and Distributed Processing*, pages 1–5. IEEE, 2008.

Case ID	Median (50^{th} percentile)				25^{th} percentile (75% pessimistic)				5^{th} percentile (95% pessimistic)			
	N	$\hat{a}^*$	a_f	d_r	N	$\hat{a}^*$	a_f	d_r	N	$\hat{a}^*$	a_f	d_r
1	19	0.952560	0.954734	0.002277	19	0.945523	0.948031	0.002646	19	0.935858	0.002277	0.003700
2	19	0.965903	0.962922	0.003097	19	0.954604	0.957312	0.002828	19	0.949370	0.003097	0.000109
3	19	0.966705	0.968018	0.001356	19	0.962980	0.961934	0.001088	19	0.934353	0.001356	0.018801
4	19	0.942744	0.932468	0.011020	19	0.918319	0.918192	0.000138	23	0.903614	0.011020	0.013062
5	19	0.897277	0.902449	0.005732	19	0.890973	0.892493	0.001703	19	0.880296	0.005732	0.002291
6	19	0.913447	0.907576	0.006469	19	0.900099	0.899102	0.001109	26	0.877142	0.006469	0.011810
7	19	0.912154	0.916944	0.005224	19	0.908060	0.908293	0.000256	19	0.888432	0.005224	0.007256
8	19	0.966131	0.965325	0.000834	19	0.961862	0.960805	0.001101	19	0.948812	0.000834	0.000323
9	19	0.784776	0.785679	0.001149	30	0.767856	0.773189	0.006897	19	0.762634	0.001149	0.007971
10	28	0.739862	0.736462	0.004617	21	0.721135	0.721837	0.000972	19	0.698968	0.004617	0.003520
11	19	0.783287	0.778369	0.006317	19	0.774306	0.762080	0.016042	133	0.725934	0.006317	0.012663
12	19	0.771074	0.748191	0.030584	125	0.727915	0.722666	0.007264	257	0.691504	0.030584	0.013951
13	19	0.592317	0.594764	0.004113	19	0.579529	0.580552	0.001761	48	0.562311	0.004113	0.003324
14	64	0.593832	0.593420	0.000693	34	0.572371	0.579086	0.011595	19	0.545807	0.000693	0.021654
15	33	0.584036	0.589625	0.009480	19	0.576236	0.573726	0.004375	21	0.563660	0.009480	0.020882
16	269	0.536075	0.530330	0.010832	241	0.481548	0.483096	0.003204	19	0.438971	0.010832	0.042167

Table 4: Results of the randomly generated experiments against 16 problem instances and 3 classes of tolerance

[14] M. Förster and M. Trapp. Fault Tree Analysis of Software-Controlled Component Systems Based on Second-Order Probabilities. In *Int. Symp. on Software Reliability Engineering*, pages 146–154. IEEE, Nov. 2009.

[15] K. Goeva-Popstojanova and K. Trivedi. Architecture-based approach to reliability assessment of software systems. *Performance Evaluation*, 45(2-3):179–204, 2001.

[16] S. Gokhale and K. Trivedi. Reliability prediction and sensitivity analysis based on software architecture. In *Int. Sym. on Software Reliability Engineering*, pages 64–75. IEEE, 2003.

[17] K. Goseva-Popstojanova and M. Hamill. Architecture-based software reliability: Why only a few parameters matter? In *Computer Software and Applications Conference, 2007.*, volume 1, pages 423–430. IEEE, 2007.

[18] K. Goseva-Popstojanova, M. Hamill, and R. Perugupalli. Large empirical case study of architecture-based software reliability. In *Int. Sym. on Software Reliability Engineering*, volume 54, pages 10–52. IEEE, 2005.

[19] K. Goseva-Popstojanova, M. Hamill, and X. Wang. Adequacy, Accuracy, Scalability, and Uncertainty of Architecture-based Software Reliability: Lessons Learned from Large Empirical Case Studies. In *Int. Symp. on Software Reliability Engineering*, pages 197–203. IEEE, 2006.

[20] K. Goseva-Popstojanova and S. Kamavaram. Assessing uncertainty in reliability of component-based software systems. In *ISSRE 2003.*, pages 307–320. IEEE, 2003.

[21] K. Goseva-Popstojanova and S. Kamavaram. Software reliability estimation under uncertainty:generalization of the method of moments. *High Assurance Systems Engineering, 2004.*, pages 209–218, 2004.

[22] L. Grunske and J. Han. A Comparative Study into Architecture-Based Safety Evaluation Methodologies Using AADL's Error Annex and Failure Propagation Models. *High Assurance Systems Engineering Symposium*, pages 283–292, 2008.

[23] A. Jhumka, M. Hiller, and N. Suri. Component-Based Synthesis of Dependable Embedded Software. *Lecture Notes in Computer Science*, 2469:111–128, 2002.

[24] P. Kubat. Assessing reliability of modular software. *Operations research letters*, 8(1):35–41, 1989.

[25] P. Limbourg. Multi-objective optimization of generalized reliability design problems using feature models - A concept for early design stages. *Reliability Engineering & System Safety*, 93(6):815–828, June 2008.

[26] M. Marseguerra, E. Zio, L. Podofillini, and D. Coit. Optimal Design of Reliable Network Systems in Presence of Uncertainty. *IEEE Transactions on Reliability*, 54(2):243–253, 2005.

[27] I. Meedeniya, B. Buhnova, A. Aleti, and L. Grunske. Architecture-driven reliability and energy optimization for complex embedded systems. In *Quality of Software Architectures, QoSA 2010*, volume 6093 of *LNCS*, pages 52–67. Springer, 2010.

[28] I. Meedeniya, B. Bühnova, A. Aleti, and L. Grunske. Reliability-driven deployment optimization for embedded systems. *Journal of Systems and Software*, 84(5):835–846, 2011.

[29] D. Montgomery and G. Runger. *Applied statistics and probability for engineers*. Wiley-India, 2007.

[30] W. Oberkampf, J. Helton, C. Joslyn, S. Wojtkiewicz, and S. Ferson. Challenge problems: uncertainty in system response given uncertain parameters. *Reliability Engineering & System Safety*, 85(1-3):11–19, 2004.

[31] R. Roshandel, S. Banerjee, L. Cheung, N. Medvidovic, and L. Golubchik. Estimating software component reliability by leveraging architectural models. In *International conference on Software engineering*, page 853. ACM Press, 2006.

[32] R. Roshandel, N. Medvidovic, and L. Golubchik. A Bayesian Model for Predicting Reliability of Software Systems at the Architectural Level. *LNCS*, 4880:108–126, 2007.

[33] R. Rubinstein and D. Kroese. *Simulation and the Monte Carlo method*. Wiley-interscience, 2008.

[34] A. Sanchez, S. Carlos, S. Martorell, and J. Villanueva. Addressing imperfect maintenance modelling uncertainty in unavailability and cost based optimization. *Reliability Engineering & System Safety*, 94(1):22–32, Jan. 2009.

[35] K. Trivedi. *Probability & Statistics with Reliability, Queuing and Computer Science Applications*. Wiley-India, 2009.

[36] W. Wang, Y. Wu, and M. Chen. An architecture-based software reliability model. In *Dependable Computing, 1999. Proceedings. 1999 Pacific Rim International Symposium on*, pages 143–150. IEEE, 2002.

[37] N. Wattanapongskorn and D. W. Coit. Fault-tolerant embedded system design and optimization considering reliability estimation uncertainty. *Rel. Eng.*, 92:395–407, 2007.

[38] L. Yin, M. Smith, and K. Trivedi. Uncertainty analysis in reliability modeling. *Reliability and Maintainability Symposium*, pages 229–234, 2001.