Efficient Sensitivity Analysis of Reliability in Embedded Software

Indika Meedeniya
Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia
imeedeniya@swin.edu.au

Aldeida Aleti
Faculty of Information Technology, Monash University, Melbourne, Australia
aldeida.aleti@monash.edu

I. Moser
Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia
imoser@swin.edu.au

ABSTRACT

The reliability of software architectures is an important quality attribute to consider in the design of embedded software systems. Reliability is especially important in safety-critical systems where human health and safety is often at stake. Researchers have developed a number of models that can estimate the reliability of software systems at design time. These models use matrix operations of considerable complexity, such as matrix inversions. During the design process of an embedded system as used by the automotive industry, the reliability of a system has to be evaluated after each change in the design. Such safety-critical systems are often subjected to sensitivity analysis, where a single parameter is changed numerous times to estimate its effect on the safety of a system, with a subsequent re-evaluation of the system's reliability.

In this paper, we introduce an efficient sensitivity analysis technique which does not require the complete re-evaluation of the system reliability. The approach uses the Δ Evaluation technique, which computes the change in reliability of the part of the system architecture which was affected by a change. Results from experiments based on a real case-study from the automotive industry indicate a significant improvement in time of the proposed approach compared to traditional sensitivity analysis methods.

Keywords

Sensitivity analysis, reliability, delta evaluation.

1. INTRODUCTION

Reliability is one of the crucial quality concerns in the design of embedded software. Design-time reliability prediction has to consider a great number of details related to the hardware and software to be used as well as the communication and usage profile of the prospective system. Probabilistic models have been developed to abstract relevant aspects and to provide quantitative reliability metrics.

In software reliability prediction, Discrete Time Markov Chain (DTMC) based models are a common formalisation technique [20, 25]. Since its first introduction by Cheung [7], this probabilistic model has been used in many different architectural styles [43] and as the basis of hierarchical reliability evaluations [21]. Goševa-Popstojonava et al. [25] illustrated the applicability of the model in various scenarios by analysing the state-of-the-art in reliability evaluation. The authors also discussed the usefulness of the model in uncertainty analysis when the 'method of moments' [24] and Monte Carlo simulation [23] are applied.

The design phase of embedded software systems generally comprises a sensitivity analysis, bottleneck identification, exploration of design options and uncertainty analysis. State-of-the-art techniques require then generation of reliability models which have to be re-evaluated in their entirety even when a small change is made to the model. This applies in run-time architecture evaluations when a component or the operational profile has been changed [14, 27].

Probabilistic model evaluation is a computation- and memory-intensive process. Extensive re-evaluations of practical-sized problems are often impossible due to complex matrix operations. Matrix inversions in DTMC based evaluations have a complexity of $\mathcal{O}(n^3)$. In such models, the time required to evaluate an architecture increases significantly with the size of the architectural model. Having to re-evaluate the entire model with every change of the architecture is particularly time-consuming in sensitivity analysis.

Sensitivity analysis investigates the relationship between model parameters and architecture-based quality evaluation. Sensitivity graphs are constructed to analyse the response characteristics of a quality attribute of interest with respect to possible variations of one or more parameters. The sensitivity analysis techniques can help analyse the effects of parameter uncertainties on specific quality attributes in detail. Cheung [7] presented a method to obtain the sensitivity of DTMC-based reliability evaluation model to component reliabilities. This method is purely analytical, and consists of 2^{nd} and 3^{rd} order partial derivatives of system reliabilities which are hard to estimate in real cases. Goševa-Popstajanova et al. [24] proposed using the method of moments to calculate the sensitivity of the system reliability to component reliabilities and transition probabilities. In this approach, variances and covariances of model parameters are employed. A similar approach is proposed by Cortellessa et

al. [10], who analyse the sensitivity of system reliability on failure propagation.

All of the above methods have taken an analytical approach to quantifying the sensitivity of system reliability. Analytical sensitivity analysis methods are hard to generalise, whereas the sensitivity models may be difficult to compute in real case scenarios. Goševa-Popstojanova et al. [22] have shown that analytical methods of uncertainty analysis do not scale well. To address this issue, they proposed a Monte Carlo simulation-based method, demonstrating that it scales better than the previous approaches [22]. In the safety evaluation domain, Förster *et al.* [18] have introduced a similar approach to obtain the sensitivity of hazard probabilities to the probability uncertainties of low level failure events, which also applies Monte Carlo simulations to obtain a probability distribution of the output hazard probability.

Monte Carlo simulation-based methods are more scalable compared to analytical methods [22]. However, these methods require the re-evaluation of the quality attributes for every sampled parameter value, which can be computationally expensive. In this paper, we introduce a novel architecture-based sensitivity analysis method for reliability, which takes advantage of Δ Evaluation [37]. Instead of re-evaluating the entire system, Δ Evaluation uses previous evaluation results to compute the changed reliability by estimating the impact of the architectural change on the overall system reliability. Only the changed part of the system Δ has to be re-evaluated and incorporated into the whole system reliability. This incremental evaluation is achieved by applying matrix operations from linear algebra.

In this paper, we integrate Δ Evaluation into the sensitivity analysis of software systems with respect to reliability and investigate the efficiency and applicability of the approach. A significant computational advantage is obtained using the new method, leveraging comprehensive reliability analysis techniques into practice. We illustrate the approach on a case study from the automotive industry.

2. STATE-OF-THE-ART IN RELIABILITY EVALUATION

Many of the probabilistic models used in software architecture evaluation are mathematical abstractions of certain aspects of the system which are relevant to an attribute of interest. Some probabilistic properties can be quantified by solving the model analytically. Probabilistic model checking [31, 26, 15, 13, 11, 30] is one way of verifying probabilistic properties from the system model. In cases where the quality attributes are given as algebraic functions of architectural parameters, the attributes can be computed by analytical methods. Usually, probabilistic models are hard to solve analytically. Model simulation techniques become useful in obtaining the quality attributes from such models

The research in this paper complements existing simulation-based reliability evaluation approaches [20, 25, 28] by introducing an efficient technique for re-evaluating software architectures in the event of changes in the model parameters or re-configurations. The approach extends the Δ Evaluation method introduced by Meedeniya and Grunske [37].

The majority of reliability evaluation methods [6, 10, 29, 38, 39] are based on Cheung's model [7]. These studies focus mainly on component-based systems where the architecture is modelled using architecture description languages. Cheung's model uses a first order partial derivative of system reliability with respect to component reliability (i.e. $\frac{\partial R}{\partial R_i}$). 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.

Cortellessa et al. [10] used Cheung's model to consider the effects of error propagation among components, whereas Wang et al. [42, 43] presented the use of Cheung's model for reliability evaluation in different architectural styles. Gokhale et al. [21] have used the model as the basis for their extension to hierarchical reliability evaluation. Goševa-Popstojonava et al. [25] illustrated extensive use of the model in their reliability evaluation survey.

These reliability evaluation methods provide only an indication of how the system reliability would change for a small change in component reliability from its original prediction. It does not produce the range of system reliability, which is a common practical requirement. To address this problem, Goševa-Popstojanova et al. [24] introduced the use of the method of moments. The approach calculates the sensitivity of a system's reliability to component reliabilities and transition probabilities. This sensitivity analysis method can help identify the most critical system components.

Coit *et al.* [44, 8] used the means and variances of the reliability estimates of software components to derive the mean and variance of the reliability of a redundancy allocation analytically. With the assumption of normally distributed input parameters, Finodella *et al.* [16] derived the distribution of system reliability from a multinomial distribution. Coit *et al.* [9] presented an analytical approach to obtain the lower bound of the reliability values in series-parallel systems. A similar analytical approach can be seen in evaluation of reliability bounds [4].

The methods described above use an analytical approach to quantifying the sensitivity, which limits their applicability to analytically solvable models, which are hard to generalise. All discussed approaches assume the parameter distributions to be normal and variations to be characterised by the mean and variance alone. To address this issue, Gošva-Popstojanova et al. [22] investigated the use of a combined analytical and simulation-based method for uncertainty analysis, and confirmed that Monte-Carlo-simulation-based methods scale better than the analysis using the method of moments. One major drawback of the Monte Carlo simulation based uncertainty analysis is that it requires the creation of a large number of sample variations of an architecture from the probability distributions of parameters, which can be very computationally expensive [3, 22, 32]. The reliability models have to be re-evaluated for each variant.

3. DTMC-BASED RELIABILITY PREDICTION

A Discrete Time Markov Chain (DTMC) is a tuple (S, P) where S is a finite set of states, and $P : S \times S \rightarrow [0, 1]$ is the transition probability matrix. A DTMC is *absorbing* when at least one of its states has no outgoing transition [41].

The program flow graph of a terminating application has a single entry and a single exit node. A terminating application is an application that operates on demand, and a single run of software that corresponds to a terminating execution can be clearly identified. This model can easily be extended to support multiple initial nodes and multiple final states by

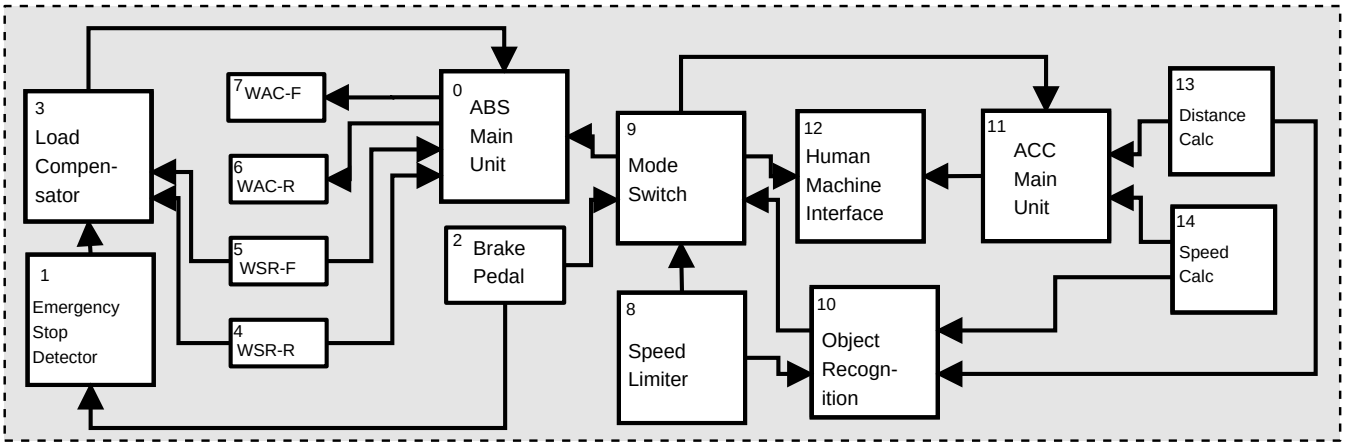


Figure 1: Automotive composite system

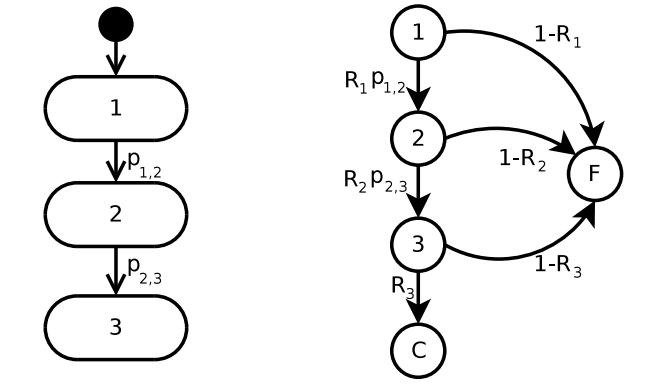
introducing *super-initial*, *super-final* states [43]. The transfer of control among modules can be described by an absorbing DTMC with transition probability matrix $P = [p_{ij}]$, where p_{ij} denotes the probability of j^{th} module being called after executing the i^{th} module.

An assumption of Cheung's model is that the components fail independently. According to this model, the reliability of component i is characterized by the probability R_i that the component performs its function correctly, i.e., the component produces the correct output and transfers control to the next component without a failure.

Two absorbing states C and F are added, representing the correct output and failure respectively. The transition probability matrix P is modified to $\hat{P}$, which incorporates these two states. The original transition probability p_{ij} between the components i and j is modified to $R_i p_{ij}$, which represents the probability that the module i produces the correct result and the control is transferred to component j. From the final (exit) state n, a directed edge to state C is added with transition probability R_n to represent the correct execution.

The failures of a component i are considered by creating a directed edge to failure state F with transition probability $(1 - R_i)$. This process integrates the failure behaviour of the components into the functional behaviour described in the original control flow. Thus, a DTMC defined by the transition probability matrix $\hat{P}$ is considered as a composite model of the software system [21]. Figure 2 illustrates a control flow graph of a software system and the corresponding DTMC when the two states C and F are added. The reliability of the system is the probability of reaching the absorbing state C of the DTMC.

Let Q be the matrix obtained from $\hat{P}$ by deleting rows and columns corresponding to the absorbing states C and F. $Q^k_{(1,n)}$ 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 proved that the infinite summation converges as follows [7]:



(a) Control Flow Graph

(b) DTMC when the states C and F are added

Figure 2: Control flow graph and corresponding DTMC for Cheung's model.

$$S = I + Q + Q^2 + Q^3 + ... = \sum_{k=0}^{\infty} Q^k = (I - Q)^{-1} \quad (1)$$

The matrix S is called the *fundamental matrix* of the DTMC, and $S_{(i,j)}$ represents the expected number of visits to the state j starting from state i before it is absorbed. Cheung [7] introduced an architecture based reliability prediction method in which the reliability of the overall system can be computed from S as follows:

$$R_s = S_{(1,n)} R_n \quad (2)$$

4. AUTOMOTIVE CASE STUDY

An embedded system from the automotive domain is used as a case study for the demonstration of the approach, which is described in the following sections. In the automotive do-

main, reliability is an important quality characteristic, because specific functions (e.g. brake assistance) are safety critical. The case study we use in this section has been designed based on already published models [19, 35] and represents a subsystem which implements an *Anti-lock Brake System (ABS)* and *Adaptive Cruise Control (ACC)* functionality. The components in the system refer to Electronic Control Units (ECU)s, which are microprocessors with built-in software. The system parameters required for the model, such as the component failure rates [2] and the estimated execution times per visit [17], have been chosen as closely to a real system as possible given the exact information is not in the public domain.

Anti-lock Brake System (ABS): ABS is currently used in the majority of modern cars to minimise hazards associated with skidding and loss of control due to locked wheels during braking. Proper rotation during brake operations allows better manoeuvrability and decreases the speed more effectively.

Adaptive Cruise Control (ACC): Apart from the usual automatic cruise control functionality, the main aim of the ACC is to avoid crashes by reducing speed once a slower vehicle is detected ahead.

The main components used by the composite system and their interaction diagram are presented in Figure 1. The *ABS Main Unit* is the major decision-making component regarding the braking levels of individual wheels, while the *Load Compensator* unit assists with computing adjustment factors from the wheel load sensor inputs. Components 4 and 5 represent the components that communicate with wheel sensors, whereas components 7 and 8 represent the electronic control units that control the brake actuators. *Brake Pedal* is the component that reads from the pedal sensor and sends the data to the *Emergency Stop Detection* unit.

Execution initialisation is possible at the components that communicate with the sensors and user inputs. In this case study the *Wheel Sensors*, *Speed Limiter*, *Object Recognition*, *Mode Switch* and *Brake Pedal* components contribute to the triggering of the service. The data from the sensors is processed by a number of components in the system and triggers are generated for the actuators like *Brake Actuators* and *Human Machine Interface*. The software components are characterised by three externally observable parameters:

(a) Failure Rate (λ_c), the failure intensity of the exponential distribution of failure behaviour of a component [40]. Component failures in the model are assumed independent.
(b) Estimated Time per Visit (t_c), the estimated time taken by component execution within a single visit of the component measured in milliseconds (ms).
(c) Execution Initiation Probability (q_0), the probability of the program execution starting at this component.

The interaction between two components $\mathcal{C}_i$ and $\mathcal{C}_j$ have the following observable characteristics:

(a) Failure Rate (λ_l), the failure intensity of the exponential distribution of failure behaviour of a communication link between two components, assumed independent for different links and given per time unit.
(b) Transfer time for a link per visit (t_l), the communication time between two components measured in a model with no redundancy, given in time units (ms).
(c) Next-step probability (p), the probability that a service calls component $\mathcal{C}_j$ after component $\mathcal{C}_i$. Parameters of the elements of the considered system, and probabilities of transferring execution from one component to another are illustrated in Table 2.

Comp ID ID	q_0	λ_c	t_c (ms)
0	0	$4 \cdot 10^{-6}$	33
1	0	$6 \cdot 10^{-6}$	30
2	0.01	$5 \cdot 10^{-6}$	10
3	0	$8 \cdot 10^{-6}$	33
4	0.17	$8 \cdot 10^{-6}$	10
5	0.17	$8 \cdot 10^{-6}$	10
6	0.17	$8 \cdot 10^{-6}$	10
7	0.17	$8 \cdot 10^{-6}$	10
8	0.01	$5 \cdot 10^{-6}$	20
9	0	$5 \cdot 10^{-6}$	20
10	0	$5 \cdot 10^{-6}$	33
11	0	$4 \cdot 10^{-6}$	28
12	0	$7 \cdot 10^{-6}$	28
13	0.15	$3 \cdot 10^{-6}$	33
14	0.15	$3 \cdot 10^{-6}$	33

Table 1: Probabilities of a process starting at a given component, component failure rates and estimated execution time.

Trans $c_i \to c_j$	Prob. $p(c_i, c_j)$	λ_l	t_l
$0 \to 7$	0.5	$4 \cdot 10^{-5}$	40
$0 \to 6$	0.5	$5 \cdot 10^{-5}$	40
$1 \to 3$	1	$6 \cdot 10^{-5}$	10
$2 \to 1$	0.75	$5 \cdot 10^{-5}$	30
$3 \to 0$	1	$4 \cdot 10^{-5}$	30
$4 \to 0$	0.7	$4 \cdot 10^{-5}$	30
$4 \to 3$	0.3	$5 \cdot 10^{-5}$	30
$5 \to 0$	0.7	$3 \cdot 10^{-5}$	30
$5 \to 3$	0.3	$5 \cdot 10^{-5}$	40
$2 \to 9$	0.25	$6 \cdot 10^{-5}$	40
$8 \to 9$	0.6	$8 \cdot 10^{-5}$	30
$8 \to 10$	0.4	$12 \cdot 10^{-5}$	30
$9 \to 0$	0.2	$4 \cdot 10^{-5}$	10
$9 \to 11$	0.4	$5 \cdot 10^{-5}$	10
$9 \to 12$	0.6	$5 \cdot 10^{-5}$	10
$10 \to 9$	1	$6 \cdot 10^{-5}$	20
$11 \to 12$	1	$8 \cdot 10^{-5}$	20
$13 \to 10$	0.5	$10 \cdot 10^{-5}$	40
$13 \to 11$	0.5	$12 \cdot 10^{-5}$	40
$14 \to 10$	0.5	$4 \cdot 10^{-5}$	40
$14 \to 11$	0.5	$5 \cdot 10^{-5}$	40

Table 2: Probabilities of communication between components, duration and failure rates of communication.

5. EFFICIENT RELIABILITY ANALYSIS

In order to obtain quantitative metrics on the reliability of a software-intensive system, an appropriate reliability model has to be constructed. Several aspects of the sys-

tem such as execution model, usage behaviour and failure characteristics have to be captured to assess the degree of reliability. Markov models are a powerful technique for the analysis of complex probabilistic systems based on the notion of states and transitions between states. They provide a comprehensive reliability modelling technique with the ability to include both structural and behavioural abstraction of a software system.

5.1 Reliability Model Construction

In this work, we assume a DTMC-based reliability model which has been widely-accepted in the field. We consider two types of possible failures in the reliability evaluation [36].

Execution failures: Components are considered as self-contained micro-computers with the software to fulfil a specific functionality. Failures may occur during the execution of a process in a software component. This type of failure is expressed in the *failure rate* values.

Communication failures: A failure of a data communication bus at a time when a software component communicates with another over the bus leads to a failure of the service that depends on this communication.

Figure 3 shows the DTMC for the case study described in Section 4. The digits in the node labels point to the corresponding nodes in Figure 1. Single digit nodes represent execution of a software component, and nodes labelled $l_{i,j}$ denote the communications between software components. A super-initial node [43] has been added to represent the start of the software execution, and arcs originating at the node have been annotated with relevant execution initialization probabilities(q_0). Two new absorbing states C and F have been added, representing the *correct output* and *failure* states respectively.

Failures during execution are mapped to arcs from each execution node c_i to F state with a probability $(1 - R_i)$ and communication failures are mapped to arcs from each communication node l_{ij} to F with a probability $(1 - R_{ij})$, where R_i, R_{ij} represent component reliability and execution link reliability respectively. Note that only a few of these failure transitions have been added for the clarity of the figure.

The failure probabilities during the execution of a software component (c_i) can be obtained from it's estimated failure rate parameter and the execution time as follows:

$$R_i = e^{-\lambda_{c_i} \times t_{c_i}} \tag{3}$$

A similar computation can be employed to establish the reliability of the communication elements, which, in our model, is characterised by the failure rates of the link, and the time taken for inter-component communication. Therefore, the reliability of the communication between component c_i and c_j is defined according to Equation 4.

$$R_{l_{ij}} = e^{-\lambda_{l_{ij}} \times t_{l_{ij}}} \tag{4}$$

5.2 Incremental Reliability Evaluation

Many of the analyses needed at design-time require re-evaluating the reliability model. For instance, *Sensitivity Analysis*, which is widely used as a technique to determine the impact of parameters on the behaviour of the composite system. Parameter sweeps are most commonly applied for this purpose and quality evaluations are necessary at each change of parameter.

The relationship between architecture to probabilistic quality model maps architectural elements (*e.g.* software components, interactions) to the elements of probabilistic model (*e.g.* nodes, transition probabilities). When a change is made to the architecture, the change is identified and propagated without a complete reconstruction of the quality evaluation model.

The model evaluation process can be enhanced by applying the change through the model instead of a complete model evaluation using Δ Evaluation [33, 37]. With the annotations in Figure 4 and using $\Rightarrow$ to represent derivation, the approach can be formally defined as $(A, \Delta A, M) \Rightarrow \Delta M \Rightarrow M'$ and $(M, \Delta M, R) \Rightarrow R'$.

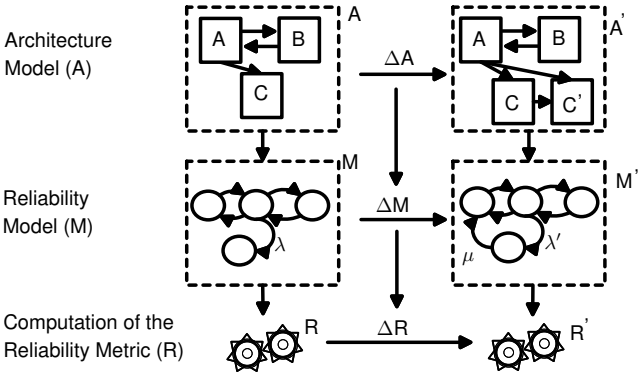


Figure 4: Outline of the proposed approach

5.3 Δ Evaluation for Sensitivity Analysis

Δ Evaluation [33, 37] can be applied in the architecture-based reliability evaluation by using it for the calculation of the probability of reaching the correct execution completion state C in the DTMC-based model. Suppose the transitions of the system have been changed and the change is expressed by ΔQ such that the new matrix Q' is $Q' = Q + \Delta Q$. The new system reliability is $R'_s = S'_{1,n} R_n$, where $S' = (I - Q')^{-1}$.

The change matrix ΔQ can be expressed as a row vector r and a column vector c such that $\Delta Q = cr$.

With the notation of $A = (I - Q)$ and B to denote the matrix obtained by deleting the n^{th} row and the 1^{st} column from A, $A' = A + \Delta A = A + c'r'$ where $c'r' = I - cr$

Similarly, $B' = B + c''r''$ where c'', r'' are obtained by removing the n^{th} row and the 1^{st} column from $c'r'$.

Meedeniya et al.[33] have shown that,

$$S'_{1,n} = \frac{(-1)^{n+1}|B'|}{|A'|} = \frac{(-1)^{n+1}|B|(1 + r''B^{-1}c'')}{|A|(1 + r'A^{-1}c')} \tag{5}$$

$$= S_{1,n} \frac{1 + r''B^{-1}c''}{1 + r'A^{-1}c'} \tag{6}$$

The results obtained in above derivation can be generalized to any single element modification (δ_{ij}) in the transition matrix Q,

$$S'_{(1,n)} = S_{(1,n)} \frac{1 - \delta_{ij} \times B^{-1}_{(j-1,i)}}{1 - \delta_{ij} \times A^{-1}_{(j,i)}} \tag{7}$$

In summary, if only one element of the transition matrix is changed, the updated reliability can be obtained simply

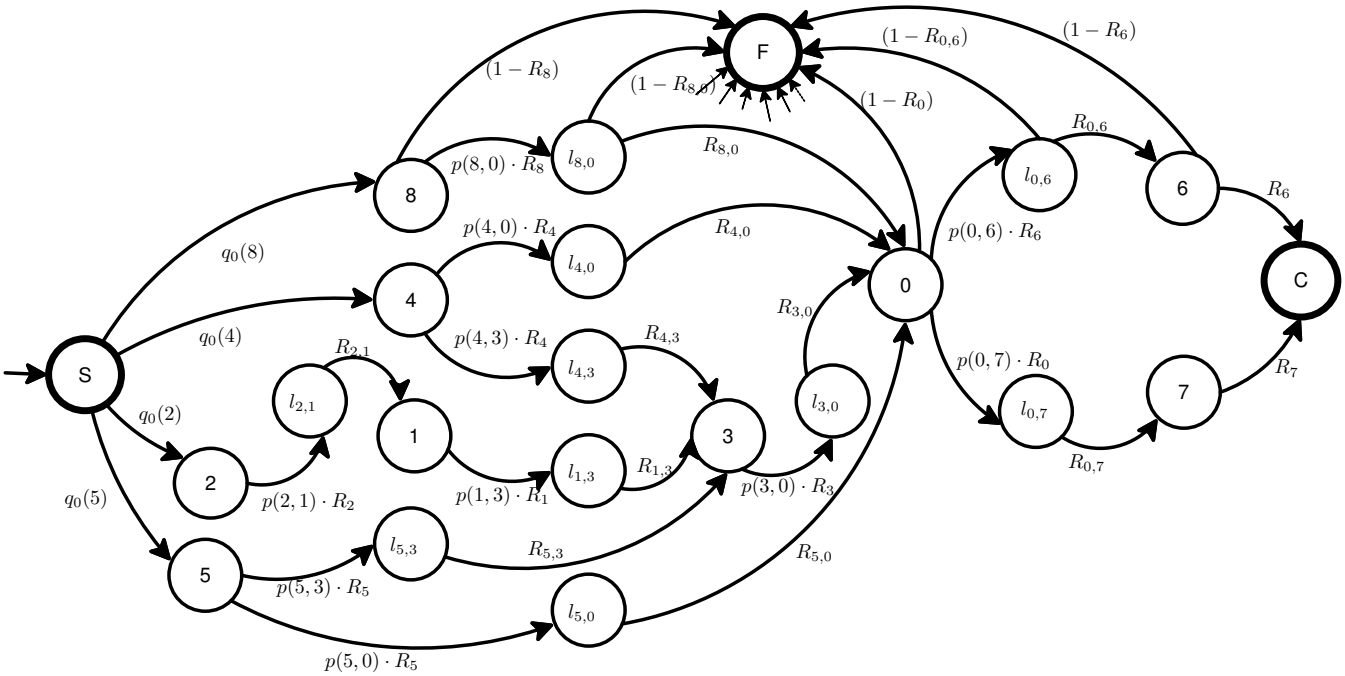


Figure 3: Annotated DTMC for service reliability evaluation. Note that only a few of the possible failure transitions are included in the diagram as an illustration. Similar arcs exist from each transient state to the final state F.

by computing Equation 7. If more than one element in one row or column in the matrix is changed, Equation 6 is applicable which requires a significantly less computation power than the computation of $(I-Q)^{-1}$. The next section further illustrates the significance of the computational gain in various analysis activities.

6. VALIDATION

The proposed sensitivity analysis of reliability using the Δ Evaluation technique has been implemented within the ArcheOpterix [1] framework, which is able to extract the architecture specification of an embedded system to analyse and optimise a variety of quality attributes. In order to validate the computational advantage of the Δ Evaluation in reliability analysis, we have conducted a series of analyses to compare the results obtained using:

- Full evaluation of the reliability model by Cheung [7] after each change,
- Δ Evaluation, and
- PRISM tool for reachability formulae evaluation.

6.1 Objectives of Reliability Analysis

A number of reliability-related aspects have to be analysed during the design stage of automotive software. The most relevant are:

- What is the predicted reliability of the automotive software system for the given configuration?
- Which are the most sensitive software components that affect the system's reliability?
- What are the best architectures to add redundancy/improve reliability?
- Which component contributes the most to the worst-case reliability?
- How sensitive is the system reliability to behavioural model estimation?

In this work, we show how the proposed method can help in answering these questions quicker and with less computational effort.

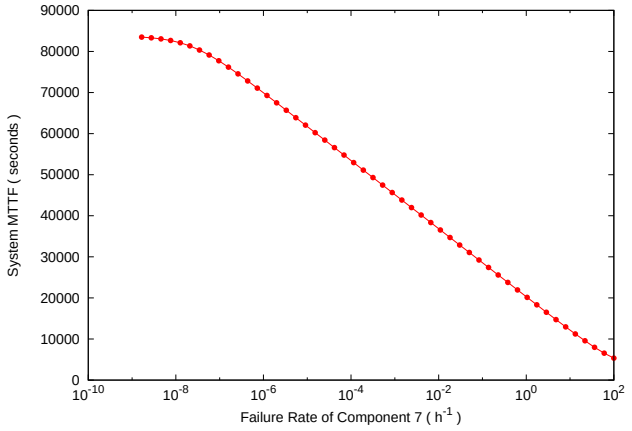


Figure 5: Sensitivity of MTTF with respect to failure rate of component 7.

6.2 Scenario 1: Sensitivity Analysis with Respect to the Failure Rate

Estimating the failure rate of software and hardware elements is a difficult task, especially in embedded systems where environment conditions may affect failures in the system [5]. For example, the operating temperature of a hardware host directly affect its failure rate [12]. Therefore, it is

important to analyse how the system reliability would vary if the component failure rate changed. We conducted an experiment to analyse the sensitivity of the mean time to failure (MTTF) when the failure rate of the brake actuator is changed (component ID 7 in Figure 1).

Figure 5 is a sensitivity graph, which depicts the variation of system MTTF against a parameter sweep of the failure rate for the range $[10^{-9} - 10^{2}]$, i.e. sensitivity of the the system MTTF to failure rate of component 7. For the purpose of this experiment, only one parameter (failure rate of component 7) was changed in 50 homogeneous steps across the range $[10^{-9} - 10^{2}]$ while all other parameters were fixed. This change affects the changes in two probabilities in the reliability model given in Figure 3: transitions $7 \rightarrow SF$, and $7 \rightarrow F$. Since the rows and columns corresponding to state F are removed to obtain the modified matrix Q, only one value is changed.

The sensitivity analysis is conducted by using the simplified formulae in Equation 6 instead of the complete model evaluation. It can be observed that the system MTTF is very sensitive to the failure rate of component 7. The impact on MTTF is not significant when component 7's failure rate is below $10^{-7}h^{-1}$ where as the MTTF drops rapidly with increasing failure rate from about $10^{-7}h^{-1}$.

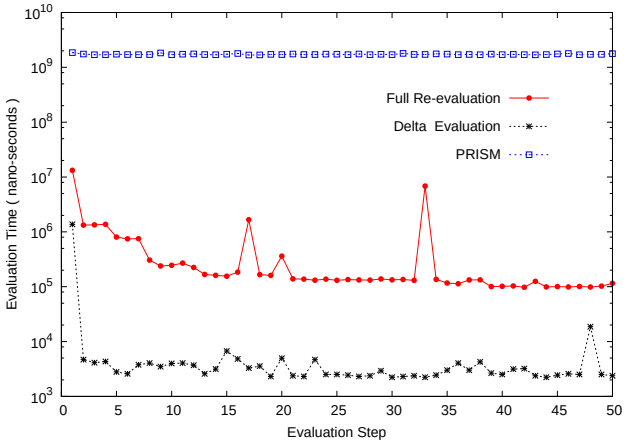


Figure 6: The computation time for each evaluation step in the analysis.

Figure 6 illustrates the computation time for each evaluation step in the sensitivity analysis experiment. It can be observed that Δ Evaluation usually takes less than 10^4 nanoseconds, whereas the complete model evaluation requires more than 10^5 nanoseconds. The extremely high evaluation time arising from the PRISM evaluation compared to the Cheung [7] model evaluation is largely explained by calls to third party applications.

6.3 Scenario 2: Sensitivity Analysis with Respect to Redundancy Allocation Parameters

A diversity of identical or similar units (hardware or software) are commonly employed in software-intensive systems in order to to gain additional tolerance against faults. As the redundant units have negative consequences such as additional cost, impact on performance, additional weight and higher energy consumption, the problem of finding appropriate redundancy levels in the architecture design has gained significant interest of the industry. For the illustration, we conducted a set of experiments by changing the redundancy levels of components in a hot-spare, homogeneous redundancy configuration. In this configuration, the reliability of a component $c_i$can be obtained from $R^a_{c_i} = (1-(1-R_i)^{a_{c_i}+1})$ where a_{c_i}denotes the redundancy level of component c_i. Meedeniya et al. [34] provided a detailed explanation of the relationship between reliability and redundancy allocation.

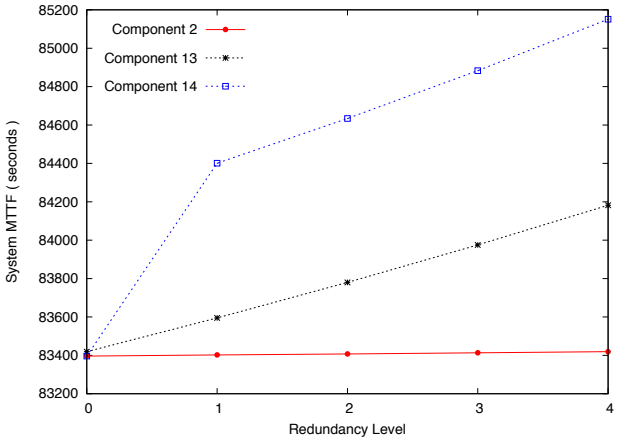


Figure 7: The variation of system reliability for different redundancy levels of brake pedal sensor, distance calculator and speed calculator.

Figure 7 depicts the variation of system reliability for different redundancy levels of brake pedal sensor, distance calculator and speed calculator. It can be seen that component 14 produces the highest gain for redundancy level 1.

These values were obtained by evaluating the reliability model for different redundancy levels. However, as changing the redundancy level of a single component changes the reliability of a logical component (R_i), this change only affects the outgoing transition probabilities from the corresponding node in DTMC model in Figure 3.

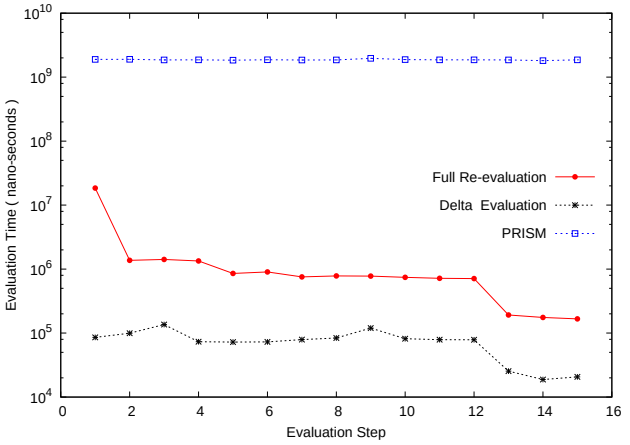


Figure 8: The computation time for each evaluation step in the analysis.

Consequently, only one row in the transition matrix is affected by a change, and equation 2 can be applied. Figure 8

illustrates the average computation times for each evaluation model. It is evident that Δ Evaluation requires significantly less computation time compared to either full model evaluation.

Table 3: The minimum, maximum, average, median and standard deviation of the time in nanoseconds taken by the three sensitivity analysis schemes.

Time	Complete	Δ Eval.	PRISM
Minimum	166,711	18,787	1,812,691,769
Average	1,957,982	75,070	1,870,216,668
Median	777,124	78,781	1,862,849,752
Maximum	18,462,611	135,981	1,970,328,002
Standard dev.	4,582,946	32,933	34,281,780

Table 3 shows the minimum, maximum, average, median and standard deviation of the time taken by the three sensitivity analysis schemes. The maximum evaluation time is high for all three methods, although it is considerably lower for the Δ Evaluation technique. The first time the model is evaluated the Δ methods has to perform a complete evaluation, hence the high computation time. The subsequent evaluations are very efficient, since only part of the model is re-evaluated. This is evidence by the very low minimum computational time for Δ Evaluation.

6.4 Scenario 3: Sensitivity Analysis with Respect to Behaviour of Model Parameters

One key advantage of Markov Chain-based reliability models is that they are capable of combining many reliability related aspects of the system, including systems execution behaviour, usage profile and failure characteristics. At the design phase of an embedded system, it is important to analyse how the system's reliability would vary according to various user profiles. Considering this analysis requirement, we conducted a sensitivity analysis experiment with respect to a set of user behaviour parameters.

For illustration, we will consider the estimated call probabilities of break pedal sensor component. The original ratio given in Table 2 is varied in the range [0,1]. Figure 9 presents the values of system reliability (MTTF) for different parameter values.

For the purpose of this experiment, only two transition probabilities were changed in the reliability model: $2 \rightarrow 1$ and $2 \rightarrow 9$. The new Δ Evaluation formula described in Equation 7 was applied and compared to the process of re-evaluating the whole model for each parameter value.

Similar to the other two scenarios, Figure 10 presents considerably lower computation time for each change in comparison to conventional model evaluation. Table 4 summarises the results. Similar to the previous scenario, the computational time of the Δ Evaluation technique is high for the first evaluation of the system reliability, hence the high maximum value, and very low for the subsequent re-evaluations of the system.

7. THREATS TO VALIDITY

The results presented in the paper may be affected by the implementation of the approach, which may be erroneous or biased towards favourable results. To avoid this,

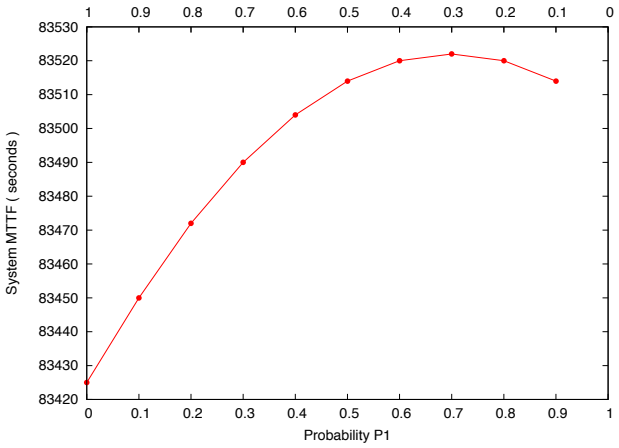


Figure 9: System reliability (MTTF) change for different parameter values.

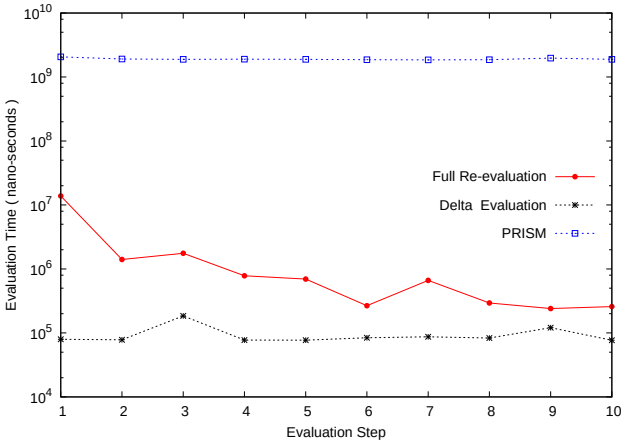


Figure 10: The computation time for each evaluation step in the analysis.

Table 4: The minimum, maximum, average, median and standard deviation of the time in nanoseconds taken by the three sensitivity analysis schemes.

Time	Complete	Δ Eval.	PRISM
Minimum	240,952	76,965	1,854,119,309
Average	2,020,277	94,942	1,909,975,390
Median	678,962	81,470	1,889,220,665
Maximum	13,841,141	184,731	2,062,513,326
Standard dev.	4,185,132	34,181	63,140,203

we have consulted experienced programmers and followed regular code-review sessions in several occasions. Due measures were taken to cross check the implementation and the conceptual design. By taking these precautions, we have minimised the possibility of having erroneous or misleading results in the experiments.

Another threat to validity of the results is that the applicability of the contribution may be limited to a specific scenario. To reduce the risk, three distinct scenarios were created, which consider very different aspects of sensitivity

analysis.

8. CONCLUSION AND FUTURE WORK

Models for reliability evaluation are computationally expensive due to the complex matrix operations involved. The time-consuming aspect of model calculations is especially crucial in sensitivity analysis, where several values of model parameters have to be analysed. The reliability of the system has to be re-evaluated for every small change made to the architecture or the parameters of the model. The approach introduced in this paper helps reduce the computational complexity of sensitivity analysis by using Δ Evaluation, which builds on previous evaluation results by evaluating a change in the architecture to estimate its reliability rather than re-evaluating the architecture in its entirety. Experiments on a practical demonstrate significant economies in computation time compared to state-of-the-art sensitivity analysis methods. In the future, we would like to extend the approach to analyse the sensitivity of reliability with respect to other design aspects, such as component deployment.

9. REFERENCES

[1] Aldeida Aleti, Stefan Björnander, Lars Grunske, and Indika Meedeniya. Archeopterix: An extendable tool for architecture optimization of AADL models. In *ICSE 2009 Workshop on Model-Based Methodologies for Pervasive and Embedded Software, MOMPES 2009*, pages 61–71. IEEE Computer Society, 2009.

[2] Ismail Assayad, Alain Girault, and Hamoudi Kalla. A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints. In *Dependable Systems and Networks (DSN'04)*, pages 347–356. IEEE Computer Society, 2004.

[3] Hans-Georg Beyer and Bernhard Sendhoff. Robust optimization - a comprehensive survey. *Computer Methods in Applied Mechanics and Eng.*, 196(33-34):3190 – 3218, 2007.

[4] A. K. 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.

[5] Alessandro Birolini. *Reliability Engineering: Theory and Practice, Fourth Edition*. Springer-Verlag, 6th edition edition, 2010.

[6] Franz Brosch and Barbora Zimmerova. Design-Time Reliability Prediction for Software Systems. In *Proceedings of the International Workshop on Software Quality and Maintainability (SQM'09)*, pages 70–74, 2009.

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

[8] David W. Coit, Tongdan Jin, and Naruemon Wattanapongsakorn. System optimization with component reliability estimation uncertainty: a multi-criteria approach. *IEEE Transactions on Reliability*, 53(3):369–380, 2004.

[9] David W. Coit and Alice E. Smith. Genetic algorithm to maximize a lower-bound for system time-to-failure with uncertain component weibull parameters. *Computers & Industrial Engineering*, 41(4):423 – 440, 2002.

[10] Vittorio Cortellessa and Vincenzo Grassi. A modeling approach to analyze the impact of error propagation on reliability of component-based systems. In *Component-Based Software Engineering CBSE'07*, volume 4608, pages 140–156, 2007.

[11] Conrado Daws. Symbolic and parametric model checking of discrete-time markov chains. In *Theoretical Aspects of Computing (ICTAC'04)*, volume 3407 of *LNCS*, pages 280–294. Springer Berlin / Heidelberg, 2004.

[12] Department of Defense. *MIL-HDBK-217,Military Handbook, Reliability Prediction of Electronic Equipment*. Department of Defense, United States of America, 1990.

[13] Matthew B. Dwyer, John Hatcliff, Robby Robby, Corina S. Pasareanu, and Willem Visser. Formal software analysis emerging trends in software model checking. In *Future of Software Engineering (FOSE'07)*, pages 120–136. IEEE Computer Society, 2007.

[14] Ilenia Epifani, Carlo Ghezzi, Raffaela Mirandola, and Giordano Tamburrelli. Model evolution by run-time parameter adaptation. In *International Conference on Software Engineering (ICSE'09)*, pages 111–121. IEEE Computer Society, 2009.

[15] Antonio Filieri, Carlo Ghezzi, and Giordano Tamburrelli. Run-time efficient probabilistic model checking. In *Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, Waikiki, Honolulu, HI, USA, May 21-28, 2011*, pages 341–350. ACM, 2011.

[16] Lance Fiondella and Swapna S. Gokhale. Software reliability with architectural uncertainties. In *IEEE International Symposium on Parallel and Distributed Processing*, pages 1–5. IEEE Computer Society, 2008.

[17] Bastian Florentz and Michaela Huhn. Embedded systems architecture: Evaluation and analysis. In *QoSA:Quality of Software Architectures, Second International Conference on Quality of Software Architectures, (QoSA'06)*, volume 4214, pages 145–162. Springer, 2006.

[18] Marc Förster and Mario Trapp. Fault tree analysis of software-controlled component systems based on second-order probabilities. In *ISSRE '09*, pages 146–154. IEEE Computer Society, 2009.

[19] Johan Fredriksson, Thomas Nolte, Mikael Nolin, and Heinz Schmidt. Contract-based reusable worst-case execution time estimate. In *The International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 39–46, 2007.

[20] Swapna S. Gokhale. Architecture-based software reliability analysis: Overview and limitations. *IEEE Transactions Dependable and Secure Computing*, 4(1):32–40, 2007.

[21] Swapna S. Gokhale and Kishor S. Trivedi. Reliability prediction and sensitivity analysis based on software architecture. In *International Symposium of Software Reliability Engineering (ISSRE'02)*, pages 64–78. IEEE Computer Society, 2002.

[22] Katerina Goševa-Popstojanova, Margaret Hamill, and Xuan Wang. Adequacy, accuracy, scalability, and uncertainty of architecture-based software reliability: Lessons learned from large empirical case studies. In *International Symposium of Software Reliability Engineering (ISSRE'06)*, pages 197–203. IEEE Computer Society, 2006.

[23] Katerina Goševa-Popstojanova and Sunil Kamavaram. Assessing uncertainty in reliability of component-based software systems. In *ISSRE*, pages 307–320. IEEE Computer Society, 2003.

[24] Katerina Goševa-Popstojanova and Sunil Kamavaram. Software reliability estimation under uncertainty: Generalization of the method of moments. In *High-Assurance Systems Engineering (HASE'04)*, pages 209–218. IEEE Computer Society, 2004.

[25] Katerina Goševa-Popstojanova and Kishor S. Trivedi. Architecture-based approach to reliability assessment of software systems. *Performance Evaluation*, 45(2-3):179–204, 2001.

[26] Lars Grunske, Robert Colvin, and Kirsten Winter. Probabilistic Model-Checking Support for FMEA. In *Proc. 4th International Conference on the Quantitative Evaluation of Systems, QEST 07*, pages 119–128. IEEE Computer Society, 2007.

[27] Lars Grunske and Pengcheng Zhang. Monitoring probabilistic properties. In *Proceedings of the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE/ESEC 2009*, pages 183–192. ACM, 2009.

[28] Anne Immonen and Eila Niemelä. Survey of reliability and availability prediction methods from the viewpoint of software architecture. *Software and System Modeling*, 7(1):49–65, 2008.

[29] Heiko Koziolek and Franz Brosch. Parameter dependencies for component reliability specifications. In *6th International Workshop on Formal Engineering approaches to Software Components and Architectures (FESCA'09)*, volume 253 of *Electronic Notes in Theoretical Computer Science*, pages 23 – 38. Elsevier, 2009.

[30] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: Probabilistic symbolic model checker. In *Computer Performance Evaluation: Modelling Techniques and Tools*, volume 2324 of *Lecture Notes in Computer Science*, pages 113–140. Springer Berlin / Heidelberg, 2002.

[31] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Probabilistic model checking in practice: case studies with prism. *SIGMETRICS Performance Evaluation Review*, 32(4):16–21, 2005.

[32] Marzio Marseguerra, Enrico Zio, and Luca Podofillini. Multiobjective spare part allocation by means of genetic algorithms and monte carlo simulation. *Reliability Engineering & System Safety*, 87(3):325 – 335, 2005.

[33] Indika Meedeniya. An incremental methodology for quantitative software architecture evaluation with probabilistic models. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2, ICSE 2010, Cape Town, South Africa, 1-8 May 2010*, pages 339–340. ACM, 2010.

[34] Indika Meedeniya, Aldeida Aleti, and Barbora Bühnova. Redundancy allocation in automotive systems using multi-objective optimisation. In *Symposium of Avionics/Automotive Systems Engineering (SAASE'09), San Diego, CA*, 2009.

[35] Indika Meedeniya, Barbora Buhnova, Aldeida Aleti, and Lars Grunske. Architecture-driven reliability and energy optimization for complex embedded systems. In *6th International Conference on the Quality of Software Architectures, QoSA 2010*, volume 6093 of *Lecture Notes in Computer Science*, pages 52–67. Springer, 2010.

[36] Indika Meedeniya, Barbora Bühnova, Aldeida Aleti, and Lars Grunske. Reliability-driven deployment optimization for embedded systems. *Journal of Systems and Software*, 84(5):835–846, 2011.

[37] Indika Meedeniya and Lars Grunske. An Efficient Method for Architecture-Based Reliability Evaluation for Evolving Systems with Changing Parameters. In *IEEE International Symposium on Software Reliability Engineering (ISSRE'10)*, pages 229–238. IEEE, 2010.

[38] Ralf Reussner, Heinz W. Schmidt, and Iman Poernomo. Reliability prediction for component-based software architectures. *Journal of Systems and Software*, 66(3):241–252, 2003.

[39] Vibhu Saujanya Sharma and Kishor S. Trivedi. Quantifying software performance, reliability and security: An architecture-based approach. *Journal of Systems and Software*, 80(4):493–509, 2007.

[40] Sol M. Shatz, Jia-Ping Wang, and Masanori Goto. Task allocation for maximizing reliability of distributed computer systems. *IEEE Trans. Computers*, 41(9):1156–1168, 1992.

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

[42] Wen-Li Wang, Dai Pan, and Mei-Hwa Chen. Architecture-based software reliability modeling. *Journal of Systems and Software*, 79(1):132–146, 2006.

[43] Wen-Li Wang, Ye Wu, and Mei-Hwa Chen. An architecture-based software reliability model. In *Pacific Rim International Symposium on Dependable Computing (PRDC'99),*, pages 143–150. IEEE Computer Society, 1999.

[44] Naruemon Wattanapongskorn and David W. Coit. Fault-tolerant embedded system design and optimization considering reliability estimation uncertainty. *Reliability Engineering & System Safety*, 92(4):395–407, 2007.