Uncertainty analysis in software reliability of component-based systems

Sunil Kumar Kamavaram
West Virginia University

Follow this and additional works at: https://researchrepository.wvu.edu/etd

Recommended Citation
Kamavaram, Sunil Kumar, "Uncertainty analysis in software reliability of component-based systems" (2004). Graduate Theses, Dissertations, and Problem Reports. 1441.
https://researchrepository.wvu.edu/etd/1441

This Thesis is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself. This Thesis has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.
Uncertainty Analysis in Software Reliability of Component-Based Systems

Sunil Kumar Kamavaram

Thesis submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Master of Science
in
Electrical Engineering

Katerina Goseva-Popstojanova, Ph.D., Chair
Hany H. Ammar, Ph.D.
Donald A. Adjeroh, Ph.D.

Department of Computer Science and Electrical Engineering
Morgantown, West Virginia
2004

Keywords: Software reliability, operational profile, component based software applications, uncertainty analysis, Markov chain, entropy, perturbation theory, method of moments and Monte Carlo simulation.
ABSTRACT

Uncertainty Analysis in Software Reliability of Component-Based Systems

Sunil Kumar Kamavaram

Several architecture-based software reliability models were proposed in the past. These models compute the point estimate of system reliability by plugging point estimates of unknown parameters into the model. These models however, discard the uncertainty of the parameters, that is, they do not attempt to answer the question how parameter uncertainties propagate into overall system reliability. Therefore, the traditional way of estimating software reliability by plugging point estimates may not be appropriate. This thesis is focused on uncertainty analysis of architecture-based software reliability models. In particular, we present uncertainty analysis using the following methods: entropy, perturbation theory, method of moments and Monte Carlo simulation. The choice of the most appropriate method is based upon data requirements, reliability measures, accuracy of the solution and scalability criteria. Entropy and perturbation theory methods study uncertainty analysis of software operational profile, while the method of moments and Monte Carlo simulation enable us to study how the uncertainty of parameters propagates into the reliability estimate. Each method for uncertainty analysis is applied and analyzed on various case studies.

Keywords: Software reliability, operational profile, component based software applications, uncertainty analysis, Markov chain, entropy, perturbation theory, method of moments and Monte Carlo simulation.
DEDICATION

I am honored to dedicate this publication to the members of my family, who have encouraged and supported me during my career. I express my love and gratefulness to my parents, Surekha and Murali Mohan, who inspired and guided me throughout my life and who serve as shining examples of parental love. I also thank my brother Satish and my sister Savitha for their affection.
ACKNOWLEDGEMENTS

I first express my sincere thanks and appreciation to Dr. Katerina Goseva-Popstajonova for providing me with an opportunity to work with her, and for being my advisor and friend. Her guidance and comments have aided me in my college career. I would like to thank my committee members, Dr. Hany Ammar and Dr. Donald Adjeroh, for their support and contribution in the course of this study.

I gratefully acknowledge the financial support for my research provided by the NASA Office of Safety and Mission Assurance (OSMA) Software Assurance Research Program (SARP) managed through the NASA Independent Verification and Validation (IV&V) Facility in Fairmont, West Virginia and by grant from the NASA West Virginia Space Grant Consortium, Research Institute Grant Program.
Contents

1 Introduction

1.1 Motivation .................................................................................................................. 1

1.2 Related Work and Uniqueness of the Research .......................................................... 4

1.3 Organization of the Thesis .......................................................................................... 6

2 Architecture-Based Methodology for Uncertainty Analysis

2.1 Methodology for Uncertainty Analysis ........................................................................ 8

2.1.1 Software Architecture .......................................................................................... 8

2.1.2 Components Failure Behavior .............................................................................. 10

2.1.3 Combining Software Architecture with Failure Behavior ..................................... 11

3 Methods for Uncertainty Analysis ................................................................................. 13
3.1 Uncertainty Analysis Based on Entropy ................................................. 14
3.2 Uncertainty Analysis Based On Perturbation Theory ............................... 15
3.3 Uncertainty Analysis Based on Method of Moments ................................ 18
3.4 Uncertainty Analysis Based on Monte Carlo Simulation ............................ 23
3.5 Comparison of Methods ................................................................. 25

4 Description of the Case Studies .......................................................... 27

4.1 European Space Agency Software ....................................................... 27
4.2 Real-Time Control System .................................................................. 32
4.3 E-commerce Application ..................................................................... 37
4.4 Cast Study Adopted From Literature .................................................... 40

5 Uncertainty Analysis on Case Studies ...................................................... 43

5.1 Uncertainty Analysis based on Entropy .................................................. 43
  5.1.1 Uncertainty of ESA operational profile ........................................... 43
  5.1.2 Uncertainty of ESA software reliability .......................................... 44
  5.1.3 Uncertainty of ESA Hypothetical Example ..................................... 48
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.4</td>
<td>Uncertainty Analysis of Real-Time Control System</td>
<td>50</td>
</tr>
<tr>
<td>5.2</td>
<td>Uncertainty Analysis based on Perturbation Theory</td>
<td>56</td>
</tr>
<tr>
<td>5.2.1</td>
<td>European Space Agency Software</td>
<td>56</td>
</tr>
<tr>
<td>5.2.2</td>
<td>E-commerce application</td>
<td>58</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Real-Time Control System: Scenario-1</td>
<td>60</td>
</tr>
<tr>
<td>5.3</td>
<td>Uncertainty Analysis based on Method of Moments</td>
<td>62</td>
</tr>
<tr>
<td>5.3.1</td>
<td>European Space Agency Software</td>
<td>62</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Case Study Adopted From Literature</td>
<td>65</td>
</tr>
<tr>
<td>5.4</td>
<td>Uncertainty Analysis based on Monte Carlo Simulation</td>
<td>68</td>
</tr>
<tr>
<td>5.4.1</td>
<td>European Space Agency Software</td>
<td>68</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Case Study Adopted From Literature</td>
<td>72</td>
</tr>
</tbody>
</table>

6 Conclusion 75
List of Figures

2.1 Methodology for uncertainty analysis of software reliability .......................... 9

3.1 Methods for Uncertainty Analysis ................................................................. 14

4.1 Special case of the methodology used for the ESA case study ....................... 28

4.2 Software architecture for the ESA case study ............................................... 29

4.3 Architecture-based software reliability model for the ESA case study ............ 30

4.4 Software architecture for the hypothetical example .................................... 31

4.5 Special case of the methodology used for the Real-time control system ....... 32

4.6 Sequence diagram of the scenario-1 ......................................................... 33

4.7 DTMC for scenario-1 .................................................................................. 34

4.8 DTMC for scenario-2 .................................................................................. 35
5.10 Execution rates and uncertainties of components in scenario-1 . . . . . . . . . . . . 52

5.11 Uncertainty for the operational profile of scenario-2 . . . . . . . . . . . . . . . . . . . . . 53

5.12 Execution rates and uncertainties of components in scenario-2 . . . . . . . . . . . . 53

5.13 Uncertainty for the operational profile of the scenario-3 . . . . . . . . . . . . . . . . . 55

5.14 Execution rates and uncertainties of components in scenario-3 . . . . . . . . . . . . 56

5.15 Parameters contribution to the variance of the system reliability for ESA . . . . 65

5.16 Parameters contribution to the variance of system reliability . . . . . . . . . . . . . . 67

5.17 Uncertainty analysis for version A . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 68

5.18 Uncertainty analysis for version B . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70

5.19 Uncertainty analysis ($p_{21} = 0$) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71

5.20 Uncertainty analysis ($p_{21} = 0.95$) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72

5.21 Uncertainty analysis for case study from Cheung’s paper . . . . . . . . . . . . . . . 73
# List of Tables

4.1 Transition probabilities and component reliabilities for versions A and B .................. 30

4.2 Comparison of the results ......................................................................................... 31

4.3 Mean values of transition probabilities and component reliabilities ......................... 41

5.1 Execution rates and uncertainties of components for operational profiles A and B .... 45

5.2 Expected execution rates and uncertainties of components for the software reliability model, versions A and B .................................................. 47

5.3 Execution rates and uncertainties of components for operational profiles C and D .... 49

5.4 Execution rates, uncertainties, and reliabilities of components for software reliability model, versions C and D .................................................. 51

5.5 Execution rates and uncertainties of components in scenario-1 ............................... 52

5.6 Execution rates and uncertainties of components in scenario-2 ............................... 54
Chapter 1

Introduction

1.1 Motivation

Software reliability is defined as the probability that the software product will work without failure in a specified environment for a specified exposure period. The exposure period can be either execution time or software runs. The environment usually is characterized by a set of input states along with their probabilities of occurrence. This probability distribution over the input space that represents the frequencies of occurrence of possible input states in the operation of software application is known as operational profile. An operational profile describes how users employ a system. Making a good reliability estimate depends on testing the product as if it is in the field. The operational profile, which is a quantitative characterization of how a system will be used, is thus essential in software reliability engineering. Moreover, the operational profile shows how to increase the reliability by allocating development resources to functions on the basis of use. Thus, the predictive quality of software reliability is affected by the ability to estimate the correct operational profile. However, building an operational profile is not an easy task, especially for a new product. Therefore, it is of critical importance to study the sensitivity of the software reliability with respect to the errors in the operational profile, particularly when
reliability estimates with high accuracy are required.

A detailed procedure for developing the operational profiles is reported in [30]. In general, operational profile is developed by some combination of system engineers, high-level designers and test planners with strong participation from product planning and marketing professionals. In order to determine the operational profile of a software, you need to look at usage of software from a progressively narrowing perspective - from customer down to operation- and, at each step, you quantify how often each of the functions in that step will be used. It has proven to be successful experience for the applications such as telecommunication systems, where the use of software is predictable because it is related to identifiable events due to human activity.

In general, the estimation of a trustworthy operational profile is difficult because the assessment requires anticipation of the usage of the software and prior information about the system application and its environment. A typical example would be a flight control system of a spacecraft in which every critical software components are activated by physical events whose frequencies during the field usage are totally unknown. Further, in process control applications, various software components are activated by complex sequences of events whose frequencies can hardly be estimated a priori. In some cases a single operational profile is not enough to explain the usage of software, which is used by different users in different ways. Because the effort required deriving an operational profile for each group of users is usually extremely high, the usual solution is to adopt an approximate operational profile that represents a rough average of the operational profiles of the different users. In addition to the above difficulties, problems could arise due to the changes in the operational profile during the development and usage of the software. Thus, software systems evolve because functions are added or modified. As a result, the way in which the software is used also evolves and the operational profile changes. This, of course, will invalidate any existing estimates of the operational profile. These reasons can easily lead to erroneous estimates of the operational profile, which will directly affect the reliability estimate. Studying the variations of the reliability estimate due to the inaccuracy in the operational profile is especially important for any real time applications which are designed to deal with events whose frequencies are difficult or impossible to predict accurately. The objective is to come up
with a methodology for computing the sensitivity of software reliability to the operational profile errors.

A number of analytical models have been proposed to address the problem of quantifying software reliability. One group of models is focused on modeling reliability growth during testing phase [9]. These so called black box models that treat the software as monolithic whole, considers only the software interactions with external environment, without an endeavor to model the internal structure. In black-box models usually no information other than failure data is used. With the growing emphasis on reuse of the software, a large number of organizations are developing and using software not only as all inclusive applications, as in the past, but also as component parts of larger applications. The existing black-box models are clearly inappropriate to model such large component-based systems. For these systems we need to take into account the information about the architecture of software made out of components, which is recognized as a white box approach. Architecture based approach has been used in the past for evaluation of the software reliability and for conducting sensitivity studies focused on component reliabilities, assuming fixed values for transition probabilities among the components. An extensive survey on architecture-based software reliability models, including their assumptions, usefulness and limitations is presented in [10].

Two important questions arise with respect to predications of software reliability based on models. The first question addresses the appropriateness of the model, since the model could be inappropriate because its assumptions may not hold in practice. The second question addresses the accuracy of parameter values. Parameters can be estimated using the field data obtained during testing or operational usage of the software, historical data for the products with similar functionality, or reasonable guesses based on the specification and design documentation.
1.2 Related Work and Uniqueness of the Research

Traditionally, the most common method for uncertainty analysis in software reliability is conducting sensitivity studies. Sensitivity of software reliability estimation to errors in the operational profile has been investigated in the context of black-box reliability growth models [4],[31],[33]. The author [31] suggests that multiple errors in the operational profile tend to have countervailing rather than cumulative effect on the failure intensity of software. From this observation, it was concluded that the failure intensity is very robust with respect to the operational profile errors. On the contrary, using the simulation approach in [4] and the experimental approach in [5] it was concluded that the inaccuracies in operational profile result in significant errors in reliability estimates. To contribute to this debate, the experimental results based on a case study were presented in [33]. These results show that the effect due to errors in the estimation of operational profile on the software reliability depends on the accuracy with which the software system is tested. Sensitivity to errors in operational profile estimate decreases greatly after the execution of thousands of test cases, because testing reaches a high coverage after the execution of so many test cases. Moreover distribution of new test cases does not affect the testability of exposing new faults.

Sensitivity studies of software reliability estimates obtained using architecture-based models have been presented in [6],[35]. In these studies the authors assumed fixed known values for the transition probabilities and derived the sensitivity of the system reliability with respect to the reliability of each component. However, any inaccuracy in the operational profile will directly affect the transition probabilities among the components. Therefore, the sensitivity studies of software reliability with respect to the operational profile (i.e., transition probabilities) and component reliabilities is presented in [11].

In addition to sensitivity studies, there have been several attempts to quantify the variability of software reliability. In [28] authors used black-box approach and assumed that the failure probability has prior Beta distribution. Using Bayesian approach, they derived the mean and
the variance of the failure probability for a software system that, in its current version, has not failed. The same problem was considered in [1] for the software with partitioned input domain. However, in this work it was admitted that there is uncertainty in the estimation of the reliability for each partition (using Beta prior distribution), as well as uncertainty in the probability of using each partition (using Dirichlet distribution). In [36] the mean and the variance of software failure probability were estimated using Bayesian approach and assuming Beta prior distributions for component failure probabilities. In another related work [25], three optimization models for software reliability allocation under an uncertain operational profile were formulated and solved. In this case the operational profile was characterized with the probabilities of function execution.

Several papers that use discrete time Markov chains to describe software usage are also relevant to our work, although they do not consider software reliability. Thus, in [38] Markov analysis of software specifications was presented and entropy was used as a measure of uncertainty. In [37] the impact of uncertainties in the operational profile on the usage coverage was analyzed. Uncertainties were specified as intervals of transition probabilities assuming a uniform distribution in the interval.

From the above it is obvious that uncertainty analysis was not used systematically and extensively in software reliability. However, it has a long tradition in other engineering applications. Thus, several methods for uncertainty analysis of system characteristics from uncertainties in component characteristics are presented in [15],[20].

Unlike the studies that take the black-box approach and treat the software as monolithic whole, we propose to take into account the information about the software architecture [10]. Architecture-based approach has been used in the past for evaluation of software reliability and for conducting sensitivity studies focused on components’ reliabilities, assuming fixed values for the transition probabilities among components. The aim of this research is to explore how the inaccuracy in the operational profile will directly affect the transition probabilities among components, which has not been considered in the studies on the architecture-based software reliability. Building on the earlier work [11], a methodology for uncertainty analysis of architecture-based
software reliability models suitable for large complex component-based applications, which is applicable throughout the software life cycle is being proposed. The methodology addresses the parameter uncertainty problem and enables us to study the sensitivity of the software reliability with respect to the erroneous estimates of the operational profile.

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

1.3 Organization of the Thesis

This thesis describes how the changes in the operational profile of the software will affect the overall system reliability. The current chapter presents motivation, related work and uniqueness of the research. Chapter 2 presents an overview of the methodology used for assessing the uncertainty in software reliability. Chapter 3 explains the various methods used to assess the uncertainty in software reliability of component based systems. In Chapter 4, a detailed description of each case study and its model is presented. Chapter 5 presents uncertainty analysis on the case studies and discusses the results that are based on different methods. The final chapter of this thesis presents the conclusion of the work.
Chapter 2

Architecture-Based Methodology for Uncertainty Analysis

Architectural approach considers the utilization and reliabilities of the components, thus allowing insight into the dynamic behavior of software executions. Therefore, the architecture-based methodology is particularly suitable for large component-based applications. Sensitivity studies of the software reliability to the operational profile errors are of essential importance for applications which, on one hand requires reliability estimates with high accuracy, while on the other hand deals with events whose frequencies are difficult to predict. For such applications plugging point estimates of the unknown parameters in the model to compute the software reliability is not appropriate because there exists a lot of uncertainty around the parameters. The proposed methodology can be used to assess the effects of the uncertainty in parameters on software reliability estimations, thus leading to more realistic reliability predictions.
2.1 Methodology for Uncertainty Analysis

The architecture-based approach to software reliability assessment considers the utilization and the reliability of components, thus allowing insight into the dynamic behavior of software executions. In order to estimate the system reliability using architecture-based model we need to know the architecture of the software (i.e., the structure of the component interactions), usage of the software described by the operational profile (i.e., relative frequencies of component interactions determined by transition probabilities) and failure behavior of the software (i.e., component reliabilities or failure rates). In this chapter, we describe the methodology for uncertainty analysis presented in Figure 2.1[13].

2.1.1 Software Architecture

Software architecture defines the software behavior with respect to the manner in which different components interact with each other. In this context, interaction of components is nothing but the transfer of execution control of the software. In case of any sequential software, at each instant, control lies in one and only one of the components. The architecture of an application may not always be readily available. In such cases, it has to be extracted from the source code or the object code of the application.

Here, a state-based approach is used to build the architecture-based software reliability model [10],[11]. This approach uses the control flow graph to represent the software architecture. The states represent active components and the arcs represent the transfer of control. Based on the assumption that the transfer of control between components has a Markov property, the architecture is modeled with discrete time Markov chain (DTMC) with a transition probability matrix $P = [p_{ij}]$, where $p_{ij} = Pr\{\text{program transits from component i to component j}\}$. The Markov chain has a two-phase construction. The structural phase involves the establishment of the static software architecture. The static software architecture can be built using different
abstraction levels as defined by the specification, or obtained using parser-based or lexically based tools. The dynamic statistical phase involves the estimation of the relative frequencies of the components interactions, i.e., transition probabilities that are clearly dependent on the operational profile. During the early phases of software development, dynamic software behavior can be captured using Unified Modelling Language (UML) use cases and sequence diagrams. During the integration phase profiles or test coverage tools can be used to obtain data necessary to describe dynamic behavior. Next, we briefly describe two different approaches that we use to build a DTMC that represents dynamic software architecture.

- **Intended Approach** is used in early phases of the software development. The estimates are based on historical data from similar products or on high-level information about software architecture obtained from specification and design documents. Since, UML is rapidly becoming a standard for software development, the UML annotations such as use cases and sequence diagrams are looked at in intended approach [3]. Use case diagrams provide
graphic description of how external entities interact with the system. Sequence diagrams depict how groups of components interact in a use case. Each sequence diagram shows a number of components and how many times the messages are exchanged between them. The general expression to estimate the transition probability from component $i$ to component $j$ is given by $p_{ij} = \frac{n_{ij}}{n_i}$, where $n_{ij}$ is the number of times messages are transmitted from component $i$ to component $j$ and $n_i$ is the total number of messages from component $i$ to all other components that are present in the sequence diagram.

- **Informed Approach** is used in the later phases of the software development life cycle. It implies the availability of the source code and an access to the code. The estimates are based on the field data for the software obtained when testing becomes available. So, the availability of code and testing tools helps us to study the dynamic behavior of the software. Thus, component traces obtained using profilers [40] and test coverage tools [41] can be used to obtain a set of execution paths and establish the frequency count of the transition arcs.

### 2.1.2 Components Failure Behavior

The next step in architecture-based methodology is to consider the failure behavior of the components, i.e., estimation of the reliability of each component $R_i$. Here, we assume that the components fail independently. The reliability of a component $i$ is the probability that the component performs its function correctly. Assessing the reliability of software components clearly depends on various factors, such as whether or not component code is available, how well the component has been tested, and whether it is a reused or a new component.

Several techniques have been proposed for estimating component’s reliability. Software reliability growth models can be applied to each software component exploiting component’s failure data obtained during testing [9]. But, due to scarcity of failure data it is not always possible to use software reliability growth models. However, there is another possibility of esti-
mating component’s reliability, i.e., from explicit consideration of non-failed executions, possibly together with failures [28],[32]. In this context, testing is not an activity for discovering faults, but an independent validation activity. With these models we may come across the problem of large number of executions necessary to establish a reasonable statistical confidence in the reliability estimate. Finally, one can use fault injection technique to estimate component’s reliability [11]. However, fault-based techniques are only as powerful as the range of fault classes that they simulate. Regardless of the technique, the estimates of component reliabilities may be inaccurate, which further motivates the use of uncertainty analysis.

2.1.3 Combining Software Architecture with Failure Behavior

Three different types of models - state-based models, path-based models and additive models - can be used in combining the software architecture with the failure behavior [10],[11].

- **State-based models** use the control flow graph to represent software architecture and estimate software reliability analytically. These models assume that the transfer of control between the components has a Markov Property, that is, model software architecture with a discrete time Markov chain (DTMC), continuous time Markov chain (CTMC), or semi Markov process (SMP). These can be further classified into absorbing and irreducible. Depending on the solution method, state-based models can be further classified as either composite or hierarchical. The former method combines the software architecture with the failure behavior into a composite model, which is then solved to predict reliability of the application. The latter method takes hierarchical approach, that is, to solve first the architectural model and then to superimpose the failure behavior on the solution of the architectural model in order to predict reliability.

- **Path-based models** compute the reliability of the software by considering the possible execution paths of the program. The sequence of components along different paths is obtained either experimentally or algorithmically. The reliability of each path is computed
by multiplying the reliabilities of the components along that path. Then, the overall system reliability is estimated by averaging path reliabilities over all paths.

- **Additive models** assume that the reliability of each component can be modeled by non-homogenous Poisson process (NHPP). Then, system failure process is also NHPP with the cumulative number of failures and failure intensity function that are the sums of the corresponding functions for each component.

In this research we use the model presented in [6] which uses state-based composite method to combine software architecture with failure behavior. Every program in general is assumed to have a single entry node and a single exit node. All the nodes in the program are considered as states of the Markov process. Two absorbing states C and F are added to the DTMC, representing the correct output and failure respectively. The transition probability matrix $P$ is modified to $\hat{P}$ as follows. The original transition probability $p_{ij}$ between the components $i$ and $j$ is modified into $R_i p_{ij}$, which represents the probability that the component $i$ produces the correct result and the control is transferred to component $j$. From the exit state $n$, a direct edge to state $C$ is created with transition probability $R_n$ to represent the correct execution. The failure of a component $i$ is considered by creating a direct edge to failure state $F$ with transition probability $(1 - R_i)$. The reliability of the program is the probability of reaching the absorbing state $C$ from the initial state 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 shown that $S = \sum_{k=0}^{\infty} Q^k = (I - Q)^{-1}$, so it follows that the overall system reliability is $R = S(1, n) R_n$. The presented methodology for uncertainty analysis can be applied to any architecture-based software reliability model that has a close form solution for the system reliability.
Reliability of a software system is estimated using the model described in Chapter 2. The expression for the system reliability is a function of transition probabilities $p_{ij}$ and component reliabilities $R_i$. These parameters are required to have input values so that the software reliability can be computed from the model. Regardless of the accuracy of the mathematical model used to model software reliability, if considerable uncertainty exists in the usage of the software and components failure data (as it usually does) then a significant uncertainty exists in the calculated system reliability. The traditional approach of computing the point estimate of the system reliability by plugging point estimates of input parameters into the model is not appropriate, because the input parameters disregards any variance due to uncertainty in input parameters. Therefore we need to study the uncertainty analysis of operational profile as well as software reliability model which quantifies the uncertainty of system reliability. Different methods can be applied for synthesizing uncertainty in system reliability from uncertainties in component reliabilities and transition probabilities (see Figure 3.1). These methods are classified into two categories:

- Methods that assess the uncertainty of operational profile and software reliability model
3.1 Uncertainty Analysis Based on Entropy

Entropy is a well-known concept used in Information Theory, which is closely tied to source coding. The extent to which a source can be compressed is related to its entropy. In 1948, Claude E. Shannon introduced a theorem which related the entropy to the number of bits per second required to represent a source without much loss. Entropy has a direct relationship to randomness, and in a sense measures the amount of uncertainty inherent in a Markov source. Intuitively, the source entropy quantifies the uncertainty present in a stochastic source $S$, and is computed by [2],[38]

$$ H(S) = - \sum_i \pi_i \sum_j p_{ij} \log p_{ij} \quad (3.1) $$

where $\pi_i$ represents the steady state probability of state $i$ and $p_{ij}$ are the transition probabilities of the stochastic source. This single value is related to the number of typical paths that are
statistically typical of the Markov chain. Thus, higher the value implies exponentially greater number of typical paths, i.e., more number of paths exists because of the uncertainty present in the source. The entropy value is maximum when all the transitions that are exit arcs from each state are equiprobable. Source entropy serves as a comparative measure for Discrete Time Markov Chain with the same structure but different transition probabilities. Thus, source entropy allows us to compare different Operational profiles. The range of entropy for a Markov chain with \( n \) states is \( 0 \leq H(S) \leq \log(n) \).

In our research, we used the concept of source entropy to quantify the uncertainty of the operational profile and architecture-based software reliability models. In addition, we quantify the uncertainty of components using the conditional entropy [23]. Thus, the uncertainty of component \( i \) (i.e., state \( i \)) is given by equation 3.2

\[
H_i = -\sum_j p_{ij} \log p_{ij}
\]  

(3.2)

In general, uncertainty of component \( i \) will be higher if it transfers the control to more components (i.e. more states are directly reachable from state \( i \)) and the transition probabilities are equiprobable. Further, we compute the steady state probabilities \( \pi = [\pi_i] \) by solving the system of equations \( \pi = \pi P \), where \( P \) is the transition probability matrix of the DTMC. Since \( \pi \) can be interpreted as the expected execution rate of component \( i \) in the long run, it represents a measure of component usage which in addition to component uncertainty \( H_i \) can be used to identify critical components.

### 3.2 Uncertainty Analysis Based On Perturbation Theory

Perturbation theory provides a mathematical means to study how the stationary distribution of an irreducible discrete time Markov chain changes due to the variation of transition probabilities
In our research we define the operational profile of a system as a discrete time Markov chain with transition probability matrix $P$. The stationary distribution vector of $P$ is a positive vector $\pi^T = (\pi_1, \pi_2, \ldots, \pi_n)$, $\Sigma_{j=1}^n \pi_j = 1$ satisfying equation

$$\pi^T P = \pi^T$$ \hspace{1cm} (3.3)

Since $\pi_i$ can be interpreted as the expected execution rate of component $i$ in the long run, it represents a measure of component usage which can be used to identify critical components. Suppose $P$ is perturbed to a matrix $\bar{P}$, which is the transition probability matrix of an $n$ state finite irreducible, homogeneous Markov chain. Sensitivity results concerning absolute perturbations have been phrased in terms of the perturb matrix $E = P - \bar{P}$ as

$$\|\pi^T - \bar{\pi}^T\| \leq k \|E\|_{\infty}$$ \hspace{1cm} (3.4)

where $\|E\|_{\infty}$ is the norm of the perturb matrix and $k$ is the condition number used as measure of sensitivity. In [19],[7],[8] eight different condition numbers $k_1, \ldots, k_8$, which give different bounds are discussed. Most of the condition numbers are expressed in terms of either the fundamental matrix $Z \equiv (A + e\pi^T)^{-1}$ [27] of the underlying Markov chain or the group inverse of $A \equiv I - P$ [34]. Several condition numbers provide good numerical measure of the maximal extent to which the magnitude of the perturbation can be amplified. However some numbers suffer from certain short comings and are not satisfying because of the following two reasons. First, irreducible chains exist for which the bounds are not tight, so the condition number $k$ may seriously overestimate the sensitivity to perturbations. Second, the bounds generally provide very little information about the relative error in individual stationary probabilities. The condition number $k_8$, expressed in terms of mean first passage times in the Markov chain, provides the tightest bound on the stationary probability vector. Moreover, viewing sensitivity in terms of mean passage time times can sometimes help practitioners decide whether or not to expect sensitivity by merely observing the structure of the chain without computing or estimating the condition numbers.
Let $P$ and $\bar{P}$ denote two transition probability matrices of a software which are represented by two irreducible $n$ state Markov chains with respective stationary probability vectors $\pi^T$ and $\bar{\pi}^T$. Let $M_{ij}$ denote the mean first passage time from state $i$ to state $j$ and $M_{jj}$ denote the mean return time for the state $j$ in the unperturbed chain. In [16],[17],[18] a standard procedure for computing the mean first passage times of a finite irreducible discrete time Markov chains has been developed. The matrix $M$ (i.e., mean first passage time matrix) is computed using Meyer’s group inverse matrix [27] given by

$$A^\# \equiv Z - \Pi$$  \hfill (3.5)

where $Z \equiv [I - P + \Pi]^{-1}$ and $\Pi = e\pi^T$.

The diagonal elements of the matrix $M$ specify the mean return time of the states and non-diagonal elements specify the mean first passage time from state $i$ to state $j$. If group inverse matrix is estimated then matrix $M$ [17] is given by

$$M = [m_{ij}] = [I - A^\# + LA^\#_d]D$$  \hfill (3.6)

where $I$ is the Identity Matrix, $A^\#$ is Meyer’s group inverse matrix given by equation 3.5, $L = ee' = [1]$ ($e$ is a column vector with all elements equal to 1, $e'$ is a row vector with all elements equal to 1), $A^\#_d$ is Meyer’s group inverse matrix with non-diagonal elements being zero and $D = (\Pi_d)^{-1}$.

The absolute change in the $j^{th}$ stationary probability [7] is given by

$$|\pi_j - \bar{\pi}_j| \leq \max_{i \neq j} M_{ij} - \min_{i \neq j} M_{ij} \|E\|_\infty$$  \hfill (3.7)

which leads to relative change in $\pi_j$ is

$$\left|\frac{\pi_j - \bar{\pi}_j}{\pi_j}\right| \leq \frac{\max_{i \neq j} M_{ij} \|E\|_\infty}{2}$$  \hfill (3.8)
3.3 Uncertainty Analysis Based on Method of Moments

Method of moments is an approximate approach that allows us to generate the moments of the system reliability from the moments of component reliabilities and transition probabilities, that is, to quantify the uncertainty in software reliability due to the uncertainty in the operational profile and the component reliabilities. Parameter moments may be determined from historical or test data or from assumed distributions. Once the parameter moments have been calculated, the system moments are obtained by combining parameter moments in a Taylor series expansion of the function that relates the system reliability to the parameters. The method is limited to those cases where the Taylor series expansion about the mean exist, and further results in a tractable number of partial derivative terms.

Method of moments involves the following steps in uncertainty analysis of software:

1. Obtain the expression for the system reliability using the architecture-based software reliability model explained in Chapter 2.
2. Expand the expression for system reliability using Taylor series.
3. Determine the moments of the component reliabilities and transition probabilities.
4. Evaluate Taylor series coefficients.
5. Estimate the mean and variance of the system reliability using the parameter moments and Taylor series coefficients.
6. Select an empirical distribution function for system reliability based on the computed moments.

Using the model presented in Chapter 2 we obtain the relationship between system reliability $R$ and the component reliabilities $R_i$ and the transition probabilities $p_{ij}$ is given by the function $R = f(R_i, p_{ij})$. The actual relationship between system reliability and components
reliabilities depends on specific software architecture. If we treat every parameter on the right hand side of this expression as a random variable then the system reliability $R$ is also a random variable. Note that the uncertainty analysis is general and the use of the method of moments in particular are not limited to this model. They can be applied to any architecture-based model that provides a closed form solution.

Let $E[R_i]$ be the mean value of the $i^{th}$ component reliability and let $\mu_k[R_i]$ denote its $k^{th}$ central moment (or moment about the mean). Further, $E[p_{ij}]$ represents the mean value of the $j^{th}$ transition coming out of the $i^{th}$ component and let $\mu_k[p_{ij}]$ denote its $k^{th}$ central moment.

The method of moments allows us to obtain the estimates of the expected value and $k^{th}$ central moments for system reliability based on (1) knowledge of the system structure $R = f(R_i, p_{ij})$ (2) data on the components reliabilities from which estimates of $E[R_i]$ and $\mu_k[R_i]$ for $i = 1, 2, \cdots, n$ can be obtained and (3) usage profile of the system from which estimates of $E[p_{ij}]$ and $\mu_k[p_{ij}]$ for $i, j = 1, 2, \cdots, n$ can be obtained.

Initially we used method of moments to quantify the uncertainties propagating into overall system reliability due to independent parameters (i.e., component reliabilities)[12],[13]. We treated each component reliability $R_i$ on the right-hand side of the system function as a random variable. Later, we generalized the method of moments. We derived expression for system reliability mean and variance, which considered both sources of uncertainty in software reliability: dependent parameters (i.e., transition probabilities) and independent parameters (i.e., component reliabilities)[14].

Generating the system reliability moments using method of moments is based on expanding the system function $R = f(R_i, p_{ij})$ in a multivariable Taylor series expansion around the statistically expected values of each of the component reliabilities $R_i$ and transition probabilities $p_{ij}$. Deriving the expression for system reliability and estimating the Taylor series coefficients by hand is cumbersome and can be done only for small systems. Therefore, the generation of system reliability moments using method of moments is a natural candidate for automation. Matlab is used to derive the symbolic expression for the system reliability where as Mathematica is used to
estimate the system reliability moments by estimating the Taylor series coefficients (i.e., partial derivatives for the Taylor series expansion).

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

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

(3.9)

where

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

(3.10)

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

(3.11)

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

(3.12)

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

\[ E[R] \sim a_0 \]  

(3.13)

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

(3.14)

In [13] we have considered the uncertainty only due to component reliabilities which resulted in only the first term in equation (3.14). Generalization of method of moments in [14] resulted in second and third terms in equation (3.14), which are due to uncertainty in transition probabilities (i.e., operational profile). Estimation of variance of software reliability in addition to the mean value is important because it is a measure of confidence in the reliability estimate. Thus, smaller values of the variance correspond to increased confidence.
The accuracy of the $E[R]$ and $\text{Var}[R]$ can be improved by including higher order terms in the Taylor series expansion. We have also derived the expression for the second order Taylor series expansion

$$R \sim a_0 + \sum_{i=1}^{n} a_{R_i} (R_i - E[R_i]) + \frac{1}{2} \sum_{i=1}^{n} a_{R_i^2} (R_i - E[R_i])^2$$
$$+ \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{R_i R_j} (R_i - E[R_i]) (R_j - E[R_j])$$
$$+ \sum_{i=1}^{n} \sum_{j=1}^{n} a_{p_{ij}} (p_{ij} - E[p_{ij}]) + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{p_{ij}^2} (p_{ij} - E[p_{ij}])^2$$
$$+ \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=j+1}^{n} a_{p_{ij} p_{ik}} (p_{ij} - E[p_{ij}]) (p_{ik} - E[p_{ik}])$$  \hspace{1cm} (3.15)$$

where $a_0$, $a_{R_i}$, and $a_{p_{ij}}$ are given by equations (3.10), (3.11), and (3.12) respectively, and

$$a_{R_i^2} = \left. \frac{\partial^2 R}{\partial R_i^2} \right|_{R_i = E[R_i], p_{ij} = E[p_{ij}]} \quad \text{for } i,j=1,2,...n$$  \hspace{1cm} (3.16)$$
$$a_{R_i R_j} = \left. \frac{\partial^2 R}{\partial R_i \partial R_j} \right|_{R_i = E[R_i], p_{ij} = E[p_{ij}]} \quad \text{for } i,j=1,2,...n$$  \hspace{1cm} (3.17)$$
$$a_{p_{ij}^2} = \left. \frac{\partial^2 R}{\partial p_{ij}^2} \right|_{R_i = E[R_i], p_{ij} = E[p_{ij}]} \quad \text{for } i,j=1,2,...n$$  \hspace{1cm} (3.18)$$
$$a_{p_{ij} p_{ik}} = \left. \frac{\partial^2 R}{\partial p_{ij} \partial p_{ik}} \right|_{R_i = E[R_i], p_{ij} = E[p_{ij}]} \quad \text{for } i,j=1,2,...n$$  \hspace{1cm} (3.19)$$

Using equation (3.15) we derive the mean of the system reliability for the second order Taylor approximation, retaining terms up to second order\(^1\)

$$E[R] \sim a_0 + \frac{1}{2} \left[ \sum_{i=1}^{n} a_{R_i^2} \text{Var}[R_i] + \sum_{k=1}^{n} \sum_{i=1}^{n} a_{p_{k_i}^2} \text{Var}[p_{k_i}] \right]$$
$$+ \sum_{k=1}^{n} \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{p_{k_i} p_{k_j}} \text{Cov}(p_{k_i}, p_{k_j}).$$  \hspace{1cm} (3.20)$$

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

\(^1\)“Retaining terms up to $k$th order” means that in the derivation all terms whose powers of the expected value sum to $k$ or less are retained and those whose powers sum to more than $k$ are dropped.
by

\[
\text{Var}[R] \sim \sum_{i=1}^{n} a_{R_i}^2 \text{Var}[R_i] + \sum_{k=1}^{n} \sum_{i=1}^{n} a_{p_{ki}}^2 \text{Var}[p_{ki}]
\]

\[+ 2 \sum_{k=1}^{n} \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{p_{ki}} a_{p_{kj}} \text{Cov}(p_{ki}, p_{kj})
\]

\[+ \sum_{i=1}^{n} a_{R_i} a_{R_i}^2 E[(R_i - E[R_i])^3]
\]

\[+ \sum_{k=1}^{n} \sum_{i=1}^{n} a_{p_{ki}} a_{p_{ki}^2} E[(p_{ki} - E[p_{ki}])^3]
\]

\[+ \sum_{k=1}^{n} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{l=1}^{n} a_{p_{ki}} a_{p_{kj}} E[(p_{ki} - E[p_{ki}])(p_{kj} - E[p_{kj}])^2]
\]

\[i \neq j
\]

\[+ 2 \sum_{k=1}^{n} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{p_{ki}} a_{p_{kj}} E[(p_{ki} - E[p_{ki}])^2(p_{kj} - E[p_{kj}])]
\]

\[i \neq j
\]

\[+ 2 \sum_{k=1}^{n} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{p_{ki}} a_{p_{kj}} a_{p_{kl}} E[(p_{ki} - E[p_{ki}])]
\]

\[i \neq j \neq l
\]

\[\cdot (p_{kj} - E[p_{kj}](p_{ki} - E[p_{ki}]))].
\]

Of course, equations for the mean (3.20) and the variance (3.21) of system reliability which take into account both sources of uncertainty in software reliability (i.e., component reliabilities and the operational profile) are far more complex that the one we derived in [13].

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

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

3.4 Uncertainty Analysis Based on Monte Carlo Simulation

Monte Carlo simulation is an approximate, but powerful method for estimating system reliability when the parameters of the software reliability model can be represented by well-defined distributions. Direct sampling method consists of the repeated generation of random variables from parameter distributions and their combination according to derived equation for system reliability. Essentially, this is equivalent to constructing many experiments or running many tests on identical systems. A direct Monte Carlo simulation consists of the following steps:

1. Derive the expression for the system reliability \( R = f(R_i, p_{ij}) \).

2. Assign probability distributions to transition probabilities \( P_{ij} \) and component reliabilities \( R_i \).

3. Estimate the parameters of these probability distributions from the engineering judgment, historical or test data.

4. Sample the probability distributions of the parameters.

5. Compute the system reliability using the expression determined in Step 1 and the values of the parameters sampled in Step 4.

6. Repeat Steps 4 & 5 until the desired number of system reliability values has been generated.
7. Calculate the moments, frequency chart and percentiles for the system reliability; do the
distribution fitting.

Here, the reliability expression (Step 1) is derived using the architecture-based software
reliability model described in Chapter 2. In Step 2 we assign probability distribution functions
to component reliabilities \( R_i \) and transition probabilities \( p_{ij} \). These distribution functions can
be based on theoretical assumption or on observed data. Component reliabilities are assumed to
be random variables with Beta distribution with pdf given by equation 3.22

\[
f(R_i) = \frac{\Gamma(a_i + b_i)}{\Gamma(a_i)\Gamma(b_i)} R_i^{a_i-1} (1 - R_i)^{b_i-1}
\]  

(3.22)

where \( 0 \leq R_i \leq 1 \). Furthermore the rows in the transition probability matrix are assumed
to be independent and distributed accordingly to Dirichlet distribution. This distribution is
commonly used for a set of proportions adding up to one and has been used in connection with
Markov transition probability matrices [26]. Thus, the joint density for the \( i^{th} \) row in transition
probability matrix has the form given by equation 3.23

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

(3.23)

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

For the simulation of Dirichlet distribution, transformation approach has been used [21]
based on the following property [22]. The standard Dirichlet distribution is defined as the distribu-
tion of \((Y_1, Y_2, \ldots, Y_n)\) where \( Y_k = \frac{Z_k}{\sum_{j=1}^{n} Z_j} \) and \( Z_j, j = 1, 2, \ldots, n \) are independent, standard
Gamma distributed random variables with shape parameters \( \alpha_j \). The Dirichlet distribution has
two properties that make it attractive. First, with the selection of different parameters it can
take a wide variety of shapes. Second, if the prior distribution is a Dirichlet distribution, then
the posterior distribution is also Dirichlet. Even in cases where the use of the Dirichlet distribu-
tion is not implied by theory, due to its variety shapes it may prove useful as an approximation.
However, our method is not restricted to Dirichlet distribution. For instance, in some cases it might be assumed that parameters vary by some fixed amount (e.g., $0.1 \pm 0.05$) and they are uniformly distributed in the interval.

### 3.5 Comparison of Methods

Source entropy is used to quantify the uncertainty of the operational profile and architecture-based software reliability models. In addition, conditional entropy is used to determine the uncertainty at component level. Perturbation theory is another method in uncertainty studies which is used to study how the changes in the operational profile will affect the expected execution rates of software components. The results obtained from these methods are useful for verification and validation of component based software systems, but these methods do not provide a software reliability estimate. The methods that provide software reliability estimate and assess the uncertainty in estimated reliability are Method of moments and Monte Carlo simulation.

Method of moments is an approximate approach which generates system reliability moments from the moments of component reliabilities and transition probabilities. The method of moments has several advantages. First, it requires only the knowledge of the moments of components reliabilities and transition probabilities that are estimated from available data, that is, no distribution function must be specified. Second, method of moments can be applied to software systems that do not fail during testing, providing estimates for the mean and the variance of the software reliability (instead of a point estimate equal to 1). Of course, this is very valuable information for high assurance systems. Finally, method of moments is an analytical approach and therefore generation of random numbers is not required, that is, there is no sampling error. However, the method of moments is an approximate method and a finite error is associated with the use of only up to first (second) order terms in the Taylor series expansion. A limitation of the method of moments is that accuracy may be increased only by including higher order
terms in the Taylor series expansion with a direct implication on computational cost. Monte Carlo simulation is also an approximate method, but it is a powerful method for assessing uncertainty and estimation of software reliability when the parameters of the model are represented by well-defined probability distributions. In the case of Monte Carlo simulation, in principle, the accuracy may be arbitrary increased simply by increasing the number of simulations.
Chapter 4

Description of the Case Studies

We apply and validate our methodology and methods on case studies from the European Space Agency, Real-Time Control System, an E-commerce application, and a case study from Cheung’s paper[6]. In this chapter the above case studies are described briefly.

4.1 European Space Agency Software

The application from the European Space Agency provides language-oriented user interface, which allows the user to describe the configuration of an array of antennas. Its purpose is to prepare a data file in accordance with a predefined format and characteristics from a user, given the array antenna configuration described using the Array Definition Language. The program was developed in C language and consists of almost 10,000 lines of code. It is divided into three subsystems: the Parser subsystem, the Computational subsystem and the Formatting subsystem. The choice of this program as a case study was based on the following:[11]

- The program is real and of typical size for this kind of application.
- The programming language is widely used.
- The program has been extensively used after the last fault removal without failures. This gold version is used as an oracle in the experiment.
- A set of test cases is generated randomly accordingly to the known operational profile determined by interviewing the users of the program.
- Component traces obtained during the testing are used for building the software architecture and estimating transition probabilities.
- Component reliabilities are estimated using fault injection. Faults reinserted in the code during the experiment are the real faults discovered during integration testing and operational use of the program.

Figure 4.1 presents the special case of our methodology for uncertainty analysis used for the European Space Agency case study.

![Figure 4.1: Special case of the methodology used for the ESA case study](image)

Discrete Time Markov Chain that describes software architecture of European Space Agency software is shown in figure 4.2. Components 1, 2 and 3 correspond to the Parser,
Computational and Formatting subsystems respectively. State $E$ represents the completion of execution of the software. The choice for decomposition was made in order to reach a tradeoff between the number of components, their size and the ability to collect data needed for use in the model. In the experiment, two faulty versions of the program were constructed. Faulty version A consists of fault-free component 3 and faulty components 1 and 2, while faulty version B consists of fault-free components 1 and 3 and faulty component 2. Each faulty version of the program and the oracle were executed on the same test cases generated randomly on the basis of the operational profile. Component traces obtained during testing were used for estimating transition probabilities $p_{ij} = \frac{n_{ij}}{n_{i}}$, where $n_{ij}$ is the number of times control was transferred form component $i$ to component $j$, and $n_{i} = \sum_j n_{ij}$.

![Figure 4.2: Software architecture for the ESA case study](image)

Failure of a component is determined by comparing the outputs of the faulty version and the oracle of the ESA. Identification of the fault responsible for the failures was only aimed at determining which component has failed. Faults have not been removed and the number of failures includes recurrences due to the same fault. Component reliabilities are estimated as $R_i = 1 - \lim_{n_i \to \infty} \frac{f_i}{n_i}$, where $f_i$ is the number of failures and $n_i$ is the number of executions of component $i$ in $N$ randomly generated test cases accordingly to the operational profile. Estimated values for transition probabilities and component reliabilities for both faulty versions are given in Table 4.1.
Table 4.1: Transition probabilities and component reliabilities for versions A and B

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
\text{Version} & p_{12} & p_{23} & R_1 & R_2 & R_3 \\
\hline
A & 0.5933 & 0.7704 & 0.8428 & 0.8346 & 1 \\
B & 0.7364 & 0.6866 & 1 & 0.8346 & 1 \\
\hline
\end{array}
\]

DTMC presented in Figure 4.3 is a composite state based software reliability model of the European Space Agency application.

The expression for system reliability obtained using the model described in section 2.1.3 is given by

\[
R = (1 - p_{12})R_1 + p_{12}(1 - p_{23})R_1 R_2 + p_{12}p_{23}R_1 R_2 R_3. \quad (4.1)
\]

In [11], the reliability estimated from the model is compared with the actual reliability estimate. As it can be seen from Table 4.2, the architecture-based software reliability model gives accurate estimations compared to the actual reliability for each of the faulty versions,
<table>
<thead>
<tr>
<th>Faulty version</th>
<th>Actual reliability</th>
<th>Estimated reliability</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.7393</td>
<td>0.7601</td>
<td>2.81%</td>
</tr>
<tr>
<td>B</td>
<td>0.8782</td>
<td>0.8782</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table 4.2: Comparison of the results

which validates the appropriateness of this model for software reliability estimation.

In addition to the real case study, we also consider a hypothetical example of software architecture given in Figure 4.4 which has an additional transition from component 2 to component 1. This example is meant to illustrate how the components executed within a loop affect software reliability.

For the example in Figure 4.4 the system reliability obtained using the model described in section 2.1.3 is given by

\[
R = \frac{(1 - p_{12})R_1 + p_{12}(1 - p_{21} - p_{23})R_1R_2 + p_{12}p_{23}R_1R_2R_3}{1 - p_{12}p_{21}R_1R_2}. \tag{4.2}
\]
4.2 Real-Time Control System

Real-time control system is a real world application. Due to confidentiality issues, much details about this application is not provided. For this case study, UML use case diagram and sequence diagrams for each case are available. Therefore, in order to build the software architecture for this case study intended approach is used. Discrete Time Markov Chains are built using UML sequence diagrams that present the software components used for given scenario and how many times the messages are exchanged between these components. The expression used to estimate the transition probability from component \( i \) to component \( j \) is given by \( p_{ij} = \frac{n_{ij}}{n_i} \), where \( n_{ij} \) is the number of times messages are transmitted from component \( i \) to component \( j \), and \( n_i \) is the total number of messages that are present in the sequence diagram from component \( i \) to all other components.

Data for the components failure behavior were not available for the Real-time control system. Also, historical failure data for similar project is also not available. Therefore, the Real-time control system case study is used to study the uncertainty analysis of the operational profile based on entropy. Figure 4.5 presents the special case of our methodology used for this case study.

![Figure 4.5: Special case of the methodology used for the Real-time control system](image-url)
Figure 4.6: Sequence diagram of the scenario-1
Out of the available sequence diagrams for the use cases in Real-time control system study we are considering three different scenarios: scenario-1, scenario-2 and scenario-3. Figure 4.6 shows the sequence diagram of the scenario-1.

DTMC of the scenario-1 is constructed by analyzing the sequence diagram given in Figure 4.6. The DTMC that represents the software execution behavior of the scenario-1 is shown in Figure 4.7.

![Figure 4.7: DTMC for scenario-1](image)

The transition probability matrix for scenario-1 is as shown below where 1,2,3,4,5,6,7 and 8 denote the start state S, components C1, C2, C3, C4, C5, C6 and end state E respectively.
\[ P = \begin{bmatrix}
1 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
4 & 0 & 1/9 & 1/9 & 0 & 1/9 & 2/9 & 4/9 & 0 \\
5 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 
\end{bmatrix} \]  \hspace{1cm} (4.3)

DTMC that represents the software execution behavior of scenario-2 is shown in Figure 4.8

![Figure 4.8: DTMC for scenario-2](image)

The transition probability matrix of scenario-2 is as shown below where 1,2,3,4,5,6,7,8,9 and 10 denote the start state S, components C6, C3, C1, C2, C7, C10, C8, C9 and end state E.
respectively.

\[
P = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 1/12 & 3/12 & 3/12 & 0 & 1/12 & 2/12 & 2/12 & 0 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
9 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

(4.4)

DTMC that represents the software execution behavior of scenario-3 is shown in Figure 4.9.

Figure 4.9: DTMC for scenario-3
The transition probability matrix for scenario-3 is given by

\[
P = \begin{bmatrix}
1 & 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
4 & 0 & 1/6 & 0 & 0 & 3/6 & 2/6 & 0 \\
5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
6 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\] (4.5)

where 1, 2, 3, 4, 5, 6 and 7 denote the start state S, components C1, C8, C3, C6, C4 and end state E respectively.

4.3 E-commerce Application

In this section we discuss the example of an E-commerce application presented in [29]. In E-commerce applications the users interact with the Web sites through sessions that consist of consecutive request to execute e-business functions (search, add to cart, pay and so on) during a single visit to the site. In the example presented in [29], the user’s navigation pattern within a session is captured by so called Customer Behavior Model Graph (CBMG). The CBMG describes how the users navigate through the site, which functions they use and the frequency of transitions form one function to the other function. The CBMG of the example presented in [29] is shown in Figure 4.10.

The transition probability matrix for occasional buyer scenario and heavy buyer scenario
are given below

\[
P_{\text{Occasional}} = \begin{bmatrix}
0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\
0 & 0.4 & 0.35 & 0 & 0.2 & 0 & 0.05 \\
0 & 0.35 & 0.4 & 0 & 0.2 & 0 & 0.05 \\
0 & 0.2 & 0.2 & 0.05 & 0.2 & 0.3 & 0.05 \\
0 & 0.425 & 0.425 & 0.05 & 0 & 0 & 0.1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\] (4.6)
In general construction of the DTMC includes constructing the structure first and then assigning transition probabilities. For Web applications, there is usually a close resemblance between navigation patterns and the underlying Web design and code because Web sites are designed to support directly such navigations. Consequently, the basic structure can be easily identified from product specification, related design documents and other information sources, which corresponds to our intended approach. Also one might use the informed approach such as for example to extract the architecture from the HTML code or the Web access logs. Similarly, transition probabilities can be obtained using either intended approach or informed approach. Intended approach includes several methods. These include educated guesses based on expert opinions, survey of target customers or historical data from similar products. However, informed approach based on measurement of the actual usage patterns is the more accurate, but the approach is also more expensive and difficult to perform. Unique to Web applications is the fact that the users accesses to Web functions can be easily recorded at the server side, without legal or other complications for on-site installations at the users sites. In fact, server logs are routinely kept to track Web usages and problems. From these logs, we can extract the information to estimate the frequencies of usage of different Web functions.

Here, we have considered the structure of the DTMC and the values of the transition probabilities from the literature [29], that is, the intended approach. Component failure data is not available for E-commerce application. Therefore, the E-commerce application is used to study the uncertainty analysis of the operational profile based on entropy. Figure 4.11 presents the special case of our methodology used for E-commerce application.
4.4 Case Study Adopted From Literature

In this section, we illustrate the example adopted from [6]. The application represents the control structure of a program by a directed graph where every node represents a program module and a directed branch between nodes represent the transfer of control. The application has 10 components and its architecture is nothing but a DTMC as shown in Figure 4.12.

As discussed before, construction of the DTMC includes constructing the structure first and then assigning transition probabilities. The DTMC for the Cheung model can be estimated using the Methodology explained in 2.1.1. The user oriented reliability model of Cheung’s example is shown in Figure 4.13. The mean values of non-zero transition probabilities $p_{ij}$ and mean component reliabilities $R_i$ are given in Table 4.3.
Figure 4.12: Software architecture of the Cheung model

<table>
<thead>
<tr>
<th>Transition Probability</th>
<th>Component Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_{12} = 0.60$</td>
<td></td>
</tr>
<tr>
<td>$p_{13} = 0.20$</td>
<td>$R_1 = 0.999$</td>
</tr>
<tr>
<td>$p_{23} = 0.70$</td>
<td></td>
</tr>
<tr>
<td>$p_{25} = 0.30$</td>
<td>$R_2 = 0.980$</td>
</tr>
<tr>
<td>$p_{35} = 1.00$</td>
<td></td>
</tr>
<tr>
<td>$p_{45} = 0.40$</td>
<td></td>
</tr>
<tr>
<td>$p_{46} = 0.60$</td>
<td>$R_3 = 0.990$</td>
</tr>
<tr>
<td>$p_{57} = 0.40$</td>
<td></td>
</tr>
<tr>
<td>$p_{58} = 0.60$</td>
<td>$R_4 = 0.970$</td>
</tr>
<tr>
<td>$p_{63} = 0.30$</td>
<td></td>
</tr>
<tr>
<td>$p_{67} = 0.30$</td>
<td>$R_5 = 0.950$</td>
</tr>
<tr>
<td>$p_{68} = 0.10$</td>
<td></td>
</tr>
<tr>
<td>$p_{69} = 0.30$</td>
<td></td>
</tr>
<tr>
<td>$p_{72} = 0.50$</td>
<td>$R_6 = 0.995$</td>
</tr>
<tr>
<td>$p_{79} = 0.50$</td>
<td></td>
</tr>
<tr>
<td>$p_{84} = 0.25$</td>
<td>$R_7 = 0.985$</td>
</tr>
<tr>
<td>$p_{8,10} = 0.75$</td>
<td></td>
</tr>
<tr>
<td>$p_{98} = 0.10$</td>
<td>$R_8 = 0.950$</td>
</tr>
<tr>
<td>$p_{9,10} = 0.90$</td>
<td></td>
</tr>
<tr>
<td>$p_{98} = 0.10$</td>
<td>$R_9 = 0.975$</td>
</tr>
<tr>
<td></td>
<td>$R_{10} = 0.985$</td>
</tr>
</tbody>
</table>

Table 4.3: Mean values of transition probabilities and component reliabilities
Figure 4.13: Software Reliability Model of Cheung's case study
Chapter 5

Uncertainty Analysis on Case Studies

5.1 Uncertainty Analysis based on Entropy

5.1.1 Uncertainty of ESA operational profile

Uncertainty analysis of an operational profile determined using source entropy is illustrated on the Markov model (see Figure 4.2) built for European Space Agency Application. In order to estimate the uncertainty of the operational profile of ESA we consider multiple software executions by adding in Figure 4.2 a transition from state $E$ to starting state 1 with transition probability equal to 1. Using source entropy equation 3.1, the variation in the uncertainty of the operational profile as a function of $p_{12}$ and $p_{23}$ is plotted in Figure 5.1.

In general, when transition probabilities are close to 0 or 1 the number of statistically typical paths will be small and the uncertainty will be low. The maximum uncertainty 0.5514 is obtained when both $p_{12}$ and $p_{23}$ are equal to 0.5. The uncertainty of the two operational profiles defined by the empirical values of transition probabilities for versions A and B given in
Figure 5.1: Uncertainty of the Operational profile as a function of $p_{12}$ and $p_{23}$

Table 4.1 are 0.4707 and 0.4604 respectively. Thus, operational profile A is more uncertain than operational profile B, although the difference is not significant.

Expected execution rates $\pi_i$ and uncertainties $H_i$ for components in the operational profiles A and B are shown in Table 5.1 and Figure 5.2. Component 1 in operational profile A has the highest uncertainty since transition probability $p_{12}$ is close to 0.5. The uncertainty of component 3 is zero because there is only one transition out of state 3, i.e., we are certain that the control will be transferred to component E.

5.1.2 Uncertainty of ESA software reliability

So far we have considered the uncertainty of the operational profile. Next, we consider the uncertainty of software reliability. As in case of the operational profile, in order to estimate
Table 5.1: Execution rates and uncertainties of components for operational profiles A and B

<table>
<thead>
<tr>
<th>States</th>
<th>$\pi_i$</th>
<th>$H_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VersionA</td>
<td>VersionB</td>
</tr>
<tr>
<td>1</td>
<td>0.3278</td>
<td>0.3085</td>
</tr>
<tr>
<td>2</td>
<td>0.1945</td>
<td>0.2271</td>
</tr>
<tr>
<td>3</td>
<td>0.1498</td>
<td>0.1560</td>
</tr>
<tr>
<td>$E$</td>
<td>0.3278</td>
<td>0.3085</td>
</tr>
</tbody>
</table>

Figure 5.2: Execution rates and uncertainties of components for operational profiles A and B

the source entropy of a DTMC given in Figure 4.3, we consider multiple software executions by adding transitions from both states $E$ and $F$ to starting state 1. The addition of failure state $F$ to the DTMC and the change in the transition probability matrix will affect the source entropy as a whole. Figure 5.3 illustrates how the uncertainty $H$ and system reliability $R$ vary as a function of $p_{12}$ and $p_{23}$ for versions A and B. As shown in these figures, consideration of components failure behavior increases the uncertainty of both versions compared to the uncertainty due to the operational profile (see Figure 5.1). Note that version B, which is more reliable, is less uncertain than version A.
Expected execution rates $\pi_i$ and uncertainties $H_i$ for the components in the software reliability model given in Figure 4.3 are shown in Table 5.2 and Figure 5.4.

If we look at the results in Table 5.1 and Table 5.2 (i.e., Figure 5.2 and Figure 5.4), we see that the uncertainty of component 1 (i.e., State 1) in version B remains the same because $R_i = 1$ results in no transition to the failure state $F$ from component 1. For all other components (i.e., components 1 and 2 in version A and component 2 in version B) the uncertainty increases due to $R_i < 1$ which leads to additional transitions to the failure state $F$. In concise, we say that components that have higher expected execution rates, higher component uncertainty and moderate reliability should be allocated more testing effort.
<table>
<thead>
<tr>
<th>States</th>
<th>$\pi_i$</th>
<th>$H_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>VersionA</td>
<td>VersionB</td>
</tr>
<tr>
<td>1</td>
<td>0.3544</td>
<td>0.3166</td>
</tr>
<tr>
<td>2</td>
<td>0.1772</td>
<td>0.2332</td>
</tr>
<tr>
<td>3</td>
<td>0.1139</td>
<td>0.1336</td>
</tr>
<tr>
<td>$E$</td>
<td>0.2694</td>
<td>0.2781</td>
</tr>
<tr>
<td>$F$</td>
<td>0.0851</td>
<td>0.0386</td>
</tr>
</tbody>
</table>

Table 5.2: Expected execution rates and uncertainties of components for the software reliability model, versions A and B

Figure 5.4: Expected execution rates and uncertainties of the components for the software reliability model, versions A and B
5.1.3 Uncertainty of ESA Hypothetical Example

In this section we illustrate the uncertainty analysis for the hypothetical example presented in Figure 4.4. We chose two operational profiles C and D with different values for the transition probability associated with the arc forming a loop in the model (0.25 and 0.75 respectively) and same values for other parameters except \( p_{12} = 0.8, p_{23} = 0.25 \). The operational profile C is slightly less uncertain than the operational profile D. Figure 5.5 shows the variation of the operational profiles C and D as a function of \( p_{12} \) and \( p_{23} \).

![Figure 5.5: Variation of the operational profiles C and D as a function of \( p_{12} \) and \( p_{23} \)](image)

Note that the value of \( p_{21} \) is fixed on 0.25 and 0.75 for the operational profiles C and D respectively, which means the range of variation for \( p_{23} \) will be 0 to 0.75 and 0 to 0.25 for operational profiles C and D respectively. Furthermore, the range of variation of the uncertainty of the operational profiles C and D are [0.0441, 0.7219] and [0.0443,0.7272] respectively.

The execution rates and the uncertainty of the components for the operational profiles C and D are shown in Table 5.3 and Figure 5.6. It can be observed that the execution rates of components 1 and 2 in version D are significantly higher than in version C. This is due to the fact that components 1 and 2 are invoked within a loop higher number of times for higher value
of \( p_{21} \). On the other side, the high value of \( p_{21} = 0.75 \) in version D results in the uncertainty of component 2 almost twice smaller than in version C.

<table>
<thead>
<tr>
<th>State</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version C</td>
<td>( \pi_i )</td>
<td>0.3571</td>
<td>0.2857</td>
<td>0.0714</td>
</tr>
<tr>
<td></td>
<td>( H_i )</td>
<td>0.7219</td>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td>Version D</td>
<td>( \pi_i )</td>
<td>0.4167</td>
<td>0.3333</td>
<td>0.0833</td>
</tr>
<tr>
<td></td>
<td>( H_i )</td>
<td>0.7219</td>
<td>0.8113</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.3: Execution rates and uncertainties of components for operational profiles C and D

Next we analyze the uncertainty of software reliability for the hypothetical example. Figure 5.7 presents the uncertainties and reliabilities of versions C and D as a function of \( p_{12} \) and \( p_{23} \). As in the case of the original ESA case study, considering components failure behavior introduces additional source of uncertainty, thus leading to higher uncertainty compared to the uncertainty of the operational profile given in Figure 5.5. We know that the software reliability for \( p_{21} = 0.75 \) is significantly lower than for \( p_{21} = 0.25 \). In addition, from the uncertainty analysis we observe the version D which is less reliable more uncertain than version C.

Expected execution rates \( \pi_i \), uncertainties \( H_i \), and component reliabilities \( R_i \) for the components in the software reliability model of the hypothetical example for both versions C and
Figure 5.7: Uncertainty and reliability for versions C and D as functions of $p_{12}$ and $p_{23}$.

D are shown in Table 5.4 and Figure 5.8. Again, we notice that considering components failure behavior (additional source of uncertainty) increases the values of components uncertainties.

5.1.4 Uncertainty Analysis of Real-Time Control System

In this section we study the uncertainty analysis using entropy on Real-Time Control System for three different scenarios - scenario-1, scenario-2 and scenario-3.

Discrete Time Markov Chain that describes the software architecture of scenario-1 shown in Figure 4.7 consists of eight components, including the starting and terminating states. The uncertainty of the operational profile defined by its transition probability matrix is 0.7505. The
Table 5.4: Execution rates, uncertainties, and reliabilities of components for software reliability model, versions C and D

<table>
<thead>
<tr>
<th>State</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\pi_i$</td>
<td>0.3739</td>
<td>0.2521</td>
<td>0.0526</td>
<td>0.2208</td>
</tr>
<tr>
<td>Version C</td>
<td>$H_i$</td>
<td>1.236</td>
<td>1.899</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$R_i$</td>
<td>0.8428</td>
<td>0.8346</td>
<td>1</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\pi_i$</td>
<td>0.4179</td>
<td>0.2818</td>
<td>0.0588</td>
<td>0.1292</td>
</tr>
<tr>
<td>Version D</td>
<td>$H_i$</td>
<td>1.236</td>
<td>1.3242</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$R_i$</td>
<td>0.8428</td>
<td>0.8346</td>
<td>1</td>
<td>NA</td>
</tr>
</tbody>
</table>

Figure 5.8: Execution rates, uncertainties, and reliabilities of components for software reliability model, versions C and D

value of uncertainty is very low when compared to the maximum uncertainty of the chain ($\log_2 8 = 3$). This is due to the fact that the DTMC of this scenario is not highly connected, that is, the transition probability matrix is sparse with many transition probabilities equal to zero. Using the source entropy equation we plotted the variation of the uncertainty of the operational profile as a function of transition probabilities $p_{12}$ and $p_{47}$, shown in Figure 5.9.

Table 5.5 and Figure 5.10 shows the component uncertainty and expected execution rate for scenario-1
Figure 5.9: Uncertainty for the operational profile for the scenario-1

Table 5.5: Execution rates and uncertainties of components in scenario-1

<table>
<thead>
<tr>
<th>States</th>
<th>S</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_1$</td>
<td>0.1334</td>
<td>0.1</td>
<td>0.1</td>
<td>0.2999</td>
<td>0.0333</td>
<td>0.0666</td>
<td>0.1334</td>
<td>0.1334</td>
</tr>
<tr>
<td>$H_i$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2.0579</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 5.10: Execution rates and uncertainties of components in scenario-1
It is evident from Table 5.5 and Figure 5.10 that the component 3 is executed most often and its uncertainty is the highest. Clearly, component C3 is the most critical component in scenario-1 and would require significantly more testing effort than the other components.

![Figure 5.11: Uncertainty for the operational profile of scenario-2](image1)

Figure 5.11: Uncertainty for the operational profile of scenario-2

Let us now consider scenario-2 with a DTMC shown in Figure 4.8. This scenario consists of 10 components including the start and the end state of the application. The uncertainty of this scenario estimated using the entropy equation is 1.093. The uncertainty of the scenario-2 is more compared to scenario-1 because more components exists in scenario-2 which leads to an increase in the number of connectivity among the components. The variation of the uncertainty of scenario-2 as a function of transition probabilities $p_{34}$ and $p_{39}$ is presented in Figure 5.11.

![Figure 5.12: Execution rates and uncertainties of components in scenario-2](image2)

Figure 5.12: Execution rates and uncertainties of components in scenario-2

Let us now consider scenario-2 with a DTMC shown in Figure 4.8. This scenario consists of 10 components including the start and the end state of the application. The uncertainty of this scenario estimated using the entropy equation is 1.093. The uncertainty of the scenario-2 is more compared to scenario-1 because more components exists in scenario-2 which leads to an increase in the number of connectivity among the components. The variation of the uncertainty of scenario-2 as a function of transition probabilities $p_{34}$ and $p_{39}$ is presented in Figure 5.11.
Table 5.6: Execution rates and uncertainties of components in scenario-2

<table>
<thead>
<tr>
<th>States</th>
<th>$\pi_i$</th>
<th>$H_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0.037</td>
<td>0</td>
</tr>
<tr>
<td>C7</td>
<td>0.037</td>
<td>0</td>
</tr>
<tr>
<td>C6</td>
<td>0.4445</td>
<td>2.4591</td>
</tr>
<tr>
<td>C1</td>
<td>0.111</td>
<td>0</td>
</tr>
<tr>
<td>C2</td>
<td>0.1111</td>
<td>0</td>
</tr>
<tr>
<td>C8</td>
<td>0.0741</td>
<td>0</td>
</tr>
<tr>
<td>C9</td>
<td>0.0741</td>
<td>0</td>
</tr>
<tr>
<td>C10</td>
<td>0.037</td>
<td>0</td>
</tr>
<tr>
<td>C3</td>
<td>0.037</td>
<td>0</td>
</tr>
<tr>
<td>T</td>
<td>0.037</td>
<td>0</td>
</tr>
</tbody>
</table>

The expected execution rates and the uncertainties of the components in scenario-2 are presented in Table 5.6 and Figure 5.12. In case of scenario-2 components C6 is executed most often and has the highest uncertainty. On the other hand if we look at the component C3, which is the most critical component in scenario-1, is not the critical component in scenario-2. Firstly, the component C3 execution rate 0.037 is significantly lower than that 0.2999 in scenario-1. Moreover, the uncertainty of component C3 in scenario-2 is zero (i.e., it transfers the control only to the end state).

Let us now consider scenario-3 of the Real-time control system. The DTMC for scenario-3 is shown in Figure 4.9. This scenario has 7 components including the starting state S and the end state E. The uncertainty of the operational profile defined by scenario-3 transition probability matrix is 0.5599. This is less uncertain when compared to scenario-1. The variation of the uncertainty of the operational profile as a function of transition probabilities $p_{12}$ and $p_{45}$ is shown in Figure 5.13.

The expected execution rates and uncertainties of components in scenario-3 are presented
Figure 5.13: Uncertainty for the operational profile of the scenario-3

in Table 5.7 and Figure 5.14.

<table>
<thead>
<tr>
<th>States</th>
<th>S</th>
<th>C1</th>
<th>C8</th>
<th>C3</th>
<th>C6</th>
<th>C4</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\pi_i$</td>
<td>0.1429</td>
<td>0.1191</td>
<td>0.0714</td>
<td>0.2857</td>
<td>0.1429</td>
<td>0.0951</td>
<td>0.1429</td>
</tr>
<tr>
<td>$H_i$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1.4595</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.7: Execution rates and uncertainties of components in scenario-3

It is obvious from Table 5.7, Figure 5.14 and perhaps Figure 4.9 that component C3 is the most critical component for scenario 3 as well as for scenario 1. The expected execution rates of component C3, which is one and the same in both the scenarios, are close (0.2857 and 0.2999 for scenario-3 and scenario-1 respectively). However, the component uncertainty for scenario-3 (1.4595) is lower than for scenario-1 (2.0579). This is due to the fact that component C3 passes the control to the smaller number of components in scenario-3 than in scenario-1. As we see that component C6 is most critical component in scenario-2 and whereas in case of scenario-3 and scenario-1 component C3 is the most critical one. C3 and C6 are the most critical components in real-time control system application and these components need most testing efforts.
5.2 Uncertainty Analysis based on Perturbation Theory

5.2.1 European Space Agency Software

DTMC that represents the software operational profile is shown in Figure 4.2. The stability of the above Markov chain is studied by perturbing the transition probability matrix, (i.e., operational profile). We consider the transition probability matrix of operational profile A denoted by $P_A$ and the perturbation matrix $E$ which leads to the transition probability matrix $P_B = P_A - E$ of the operational profile B.

$$P_A = \begin{bmatrix} 0 & 0.5933 & 0 & 0.4067 \\ 0 & 0 & 0.7704 & 0.2296 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (5.1)$$
Using the equations 3.7, 3.8 and 3.6 we estimate the absolute and the relative changes in components execution rates. As it can be seen from Table 5.8, the operational profile A is absolutely stable since each stationary probability is insensitive in the absolute sense to perturbations in $P_A$.

Next, we consider the hypothetical example based on the European Space Agency application which has a loop back from component 2 to 1 [11]. The architecture of the software for hypothetical example is shown in Figure 4.4. Here, we assume that the transition probability matrix $P_C$ of the hypothetical example is perturbed by matrix $E$ which results into a new operational profile $P_D = P_C - E$.

$$P_C = \begin{bmatrix} 0 & 0.8 & 0 & 0.2 \\ 0.25 & 0 & 0.25 & 0.5 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$  \hspace{1cm} (5.3)
\[ E = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & -0.5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \] (5.4)

From the results given in Table 5.9 it can be seen that the operational profile \( C \) is not stable in relative sense to the perturbations. In particular, the component 3 (i.e., the Formatting subsystem), which has the smallest execution rate, is the most sensitive in relative sense to the changes made to the transition probability matrix \( P_C \).

<table>
<thead>
<tr>
<th>States</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution rate</td>
<td>0.3571</td>
<td>0.2857</td>
<td>0.0714</td>
<td>0.2857</td>
</tr>
<tr>
<td>Absolute change</td>
<td>0.3214</td>
<td>0.3571</td>
<td>0.4643</td>
<td>0.3571</td>
</tr>
<tr>
<td>Relative change</td>
<td>0.9</td>
<td>1.25</td>
<td>6.5</td>
<td>1.25</td>
</tr>
</tbody>
</table>

Table 5.9: Perturbation analysis of the hypothetical example

5.2.2 E-commerce application

In this section we analyze the sensitivity of a software usage in a typical e-commerce application adopted from [29]. DTMC that describes the typical e-commerce application is shown in Figure 4.10. Due to a large number of diverse users, a single operational profile is not sufficient to describe the use of the Web site by different users. Thus, in [29] two operational profiles are given showing the usage of the same e-commerce site by two different types of users: an occasional buyer and a heavy buyer. Here, we consider the operational profile \( P \) typical for the occasional
buyer and the perturbation matrix $E$ that leads to operational profile typical for the heavy buyer.

$$
P = 
\begin{bmatrix}
0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\
0 & 0.4 & 0.35 & 0.2 & 0 & 0.05 \\
0 & 0.35 & 0.4 & 0 & 0.2 & 0 & 0.05 \\
0 & 0.2 & 0.2 & 0.05 & 0.2 & 0.3 & 0.05 \\
0 & 0.425 & 0.425 & 0.05 & 0 & 0 & 0.1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

(5.5)

$$
E = 
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -0.05 & -0.05 & 0 & 0 & 0 & 0.1 \\
0 & -0.05 & -0.05 & 0 & 0 & 0 & 0.1 \\
0 & 0.025 & 0.025 & 0.05 & -0.1 & 0 & 0 \\
0 & -0.1 & -0.1 & 0.25 & 0 & 0 & -0.05 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

(5.6)

As explained before the components with lowest execution rates are more sensitive relatively to changes in the operational profile. Thus, it can be observed from the results shown in Table 5.10 that the components ‘Add’ and ‘Pay’ which are visited rarely in the operational profile of an occasional buyer exhibits excessive relative change in execution rate due to the changes in the operational profile. In particular, the relative change of the expected execution rate of the component ‘Add’ is one order of magnitude higher and the relative change of the expected execution rate of component ‘Pay’ is two orders of magnitude higher than the execution rates of other components in the e-commerce case study.
<table>
<thead>
<tr>
<th>States</th>
<th>Execution rate</th>
<th>Absolute change</th>
<th>Relative change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry</td>
<td>0.0542</td>
<td>0.2365</td>
<td>4.3615</td>
</tr>
<tr>
<td>Browse</td>
<td>0.3666</td>
<td>0.3149</td>
<td>0.8589</td>
</tr>
<tr>
<td>Search</td>
<td>0.3666</td>
<td>0.3149</td>
<td>0.8589</td>
</tr>
<tr>
<td>Add</td>
<td>0.0078</td>
<td>0.2652</td>
<td>34</td>
</tr>
<tr>
<td>Select</td>
<td>0.1428</td>
<td>0.2779</td>
<td>1.875</td>
</tr>
<tr>
<td>Pay</td>
<td>0.0023</td>
<td>0.2494</td>
<td>106.58</td>
</tr>
<tr>
<td>Exit</td>
<td>0.0542</td>
<td>0.2365</td>
<td>4.3615</td>
</tr>
</tbody>
</table>

Table 5.10: Perturbation analysis of the e-commerce case study

5.2.3 Real-Time Control System : Scenario-1

We consider a Real-time control system: scenario-1, whose operational profile is shown in Figure 4.7. The sensitivity of the operational profile of this scenario is analyzed by perturbing the transition probability matrix $P$ (see equation 4.3) with perturb matrix $E$. The resulting matrix $\tilde{P}$ represents a different usage of the Real-time control application.

\[
P = \begin{bmatrix}
0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1/9 & 1/9 & 0 & 1/9 & 2/9 & 4/9 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 
\end{bmatrix}
\] (5.7)

60
Table 5.11 shows the results of the perturbation analysis of the Real-time control system obtained using equations 3.7 and 3.8. We observe from the table that the execution rates of components C1, C2, and C3 have the same absolute change. However, the execution rate of the component C1 and C2 are more sensitive in relative sense to the perturbations than component C3. The components C4 and C5 which are visited rarely are more sensitive to changes in the relative sense. This observation is very important for software verification and validation due to the fact that rarely executed components usually handle critical functionality such as for example exception handling or recovery.

<table>
<thead>
<tr>
<th>States</th>
<th>Execution rate</th>
<th>Absolute change</th>
<th>Relative change</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>0.1333</td>
<td>0.2889</td>
<td>2.1667</td>
</tr>
<tr>
<td>C1</td>
<td>0.1</td>
<td>0.3</td>
<td>3</td>
</tr>
<tr>
<td>C2</td>
<td>0.1</td>
<td>0.3</td>
<td>3</td>
</tr>
<tr>
<td>C3</td>
<td>0.3</td>
<td>0.3</td>
<td>1</td>
</tr>
<tr>
<td>C4</td>
<td>0.0333</td>
<td>0.3556</td>
<td>10.667</td>
</tr>
<tr>
<td>C5</td>
<td>0.0667</td>
<td>0.3778</td>
<td>5.6667</td>
</tr>
<tr>
<td>C6</td>
<td>0.1333</td>
<td>0.2889</td>
<td>2.1667</td>
</tr>
<tr>
<td>E</td>
<td>0.1333</td>
<td>0.2889</td>
<td>2.1667</td>
</tr>
</tbody>
</table>

Table 5.11: Perturbation analysis of the Real-time control system - Scenario 1
5.3 Uncertainty Analysis based on Method of Moments

5.3.1 European Space Agency Software

In this section, we apply the method of moments (see section 3.3) on ESA case study. The mean and variances of the transition probabilities and component reliabilities are given in Table 5.12. In addition to the mean $E[R]$ and the variance $\text{Var}[R]$ of the system reliability, we estimate the coefficient of variation $C_R = \sqrt{\text{Var}[R]/E[R]}$ which is a relative measure of the spread of the distribution and allows us to compare different distributions.

<table>
<thead>
<tr>
<th></th>
<th>$p_{12}$</th>
<th>$p_{23}$</th>
<th>$R_1$</th>
<th>$R_2$</th>
<th>$R_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Version A</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>0.5933</td>
<td>0.7704</td>
<td>0.8428</td>
<td>0.8346</td>
<td>0.9995</td>
</tr>
<tr>
<td>Variance</td>
<td>0.02974</td>
<td>0.02579</td>
<td>0.00571</td>
<td>0.00568</td>
<td>0.00001</td>
</tr>
<tr>
<td><strong>Version B</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean</td>
<td>0.7364</td>
<td>0.6866</td>
<td>0.9995</td>
<td>0.8346</td>
<td>0.9995</td>
</tr>
<tr>
<td>Variance</td>
<td>0.02452</td>
<td>0.02556</td>
<td>0.00001</td>
<td>0.00568</td>
<td>0.00001</td>
</tr>
</tbody>
</table>

Table 5.12: Transition probabilities and Component Reliabilities for versions A and B

Table 5.13 compares the values obtained for the mean, variance, and coefficient of variation of the system reliability for versions A and B using first and second order Taylor series expansion. As we already knew from the point estimates, version B has higher mean reliability than version A. The uncertainty analysis provides an additional information about the variance of the system reliability estimate. Thus, the reliability of version B has a smaller variance, that is, the distribution is less spread than the distribution for version A. The smaller value of the variance means that we have a higher confidence in the reliability estimate of version B. As it can be seen from Table 5.13, the second order approximation does not improve the accuracy for this example. This is due to the fact that all second and higher order partial derivates are zero since the system reliability given by equation (4.1) is a linear function of components reliabilities and transition probabilities.
Table 5.13: The mean and variance of the system reliability for the ESA case study

Next we consider the uncertainty analysis for the hypothetical example. In this case we choose four versions, C, D, E and F with different values for the transition probability $p_{21}$ associated with the arc forming a loop in the model. Table 5.15 shows the mean and variances of the transition probabilities of versions, C, D, E and F respectively. Component reliabilities for versions C, D, E and F are the same as for version A given in Table 5.14.

<table>
<thead>
<tr>
<th></th>
<th>$p_{12}$</th>
<th>$p_{23}$</th>
<th>$p_{21}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version C</td>
<td>Mean 0.8</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Variance 0.01164</td>
<td>0.01831</td>
<td>NA</td>
</tr>
<tr>
<td>Version D</td>
<td>Mean 0.8</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td>Variance 0.01164</td>
<td>0.01831</td>
<td>0.01831</td>
</tr>
<tr>
<td>Version E</td>
<td>Mean 0.8</td>
<td>0.25</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>Variance 0.01164</td>
<td>0.01831</td>
<td>0.02441</td>
</tr>
<tr>
<td>Version F</td>
<td>Mean 0.8</td>
<td>0.25</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td>Variance 0.01164</td>
<td>0.01831</td>
<td>0.01831</td>
</tr>
</tbody>
</table>

Table 5.14: Transition probabilities and Component Reliabilities for versions A and B

In general, higher order Taylor series expansion increases the accuracy, as it can be seen from Table 5.16 which presents the results obtained for the hypothetical example. In view of
Table 5.15: Transition probabilities for versions C and D

Table 5.15 shows the transition probabilities for versions C and D. We observe that the mean system reliability decreases for higher values of transition probability $p_{21}$. In addition, we see that for higher values of $p_{21}$ the coefficient of variation (i.e., the spread of the distribution) is increasing.

Table 5.16: The mean and variance of the system reliability for the hypothetical example

Next, we study the parameters contribution to the variance of system reliability. As it can be seen from the Figure 5.15, in the case of version A 91.28% of the variance is due to component reliabilities and only 8.72% to transition probabilities. In case of version B the system reliability is still more sensitive to the variation of the component reliabilities, although
with smaller contribution to the variance (82.12%). Further, it is obvious that the parameter \( p_{21} \) affects significantly the parameters contribution to the variance of system reliability. Thus, in case of version C (i.e., \( p_{21} = 0 \)) component reliabilities contribute 96.82\% to the variance of system reliability, while in case of version F (i.e., \( p_{21} = 0.75 \)) they contribute 55.09\%. We see that in Figure 5.15 as \( p_{21} \) is increasing (i.e. from version C to version F), which is the arc forming a loop in the model, the contribution of transition probabilities to the system reliability is increasing. These results clearly illustrate the usefulness of the uncertainty analysis and motivate its systematic use for software reliability prediction.

### 5.3.2 Case Study Adopted From Literature

In this section, we illustrate the method of moments on the example adopted from [6]. As discussed earlier, the application has 10 components and its architecture is described by the
| $E[p_{12}] = 0.60$ | $E[p_{13}] = 0.20$ | $E[p_{14}] = 0.20$ | $E[R_1] = 0.999$ |
| Var$[p_{12}] = 0.038003$ | Var$[p_{13}] = 0.025336$ | Var$[p_{14}] = 0.025336$ | Var$[R_1] = 0.000069$ |
| $E[p_{23}] = 0.70$ | $E[p_{25}] = 0.30$ | $E[R_2] = 0.980$ | Var$[R_2] = 0.012078$ |
| Var$[p_{23}] = 0.030849$ | Var$[p_{25}] = 0.030849$ | Var$[R_3] = 0.990$ | Var$[R_4] = 0.003850$ |
| $E[p_{35}] = 1.00$ | $E[p_{46}] = 0.60$ | $E[R_3] = 0.950$ | Var$[R_5] = 0.000015$ |
| Var$[p_{35}] = 0$ | Var$[p_{46}] = 0.032968$ | Var$[R_5] = 0.000451$ | Var$[R_6] = 0.001122$ |
| $E[p_{45}] = 0.40$ | $E[p_{58}] = 0.60$ | $E[R_6] = 0.995$ | Var$[R_6] = 0.000045$ |
| Var$[p_{45}] = 0.032968$ | Var$[p_{58}] = 0.031784$ | Var$[R_7] = 0.007605$ | Var$[R_8] = 0.000451$ |
| $E[p_{57}] = 0.40$ | $E[p_{68}] = 0.30$ | $E[R_8] = 0.975$ | Var$[R_9] = 0.017091$ |
| Var$[p_{57}] = 0.031784$ | Var$[p_{68}] = 0.0441874$ | Var$[R_9] = 0.007605$ | Var$[R_{10}] = 0.000015$ |
| $E[p_{63}] = 0.30$ | $E[p_{67}] = 0.30$ | $E[R_{10}] = 0.985$ | Var$[R_{10}] = 0.000045$ |
| Var$[p_{63}] = 0.0441874$ | Var$[p_{67}] = 0.0041874$ | Var$[R_{10}] = 0.000045$ | Var$[R_{10}] = 0.000045$ |
| $E[p_{72}] = 0.50$ | $E[p_{79}] = 0.50$ | $E[R_{11}] = 0.985$ | Var$[R_{11}] = 0.000045$ |
| Var$[p_{72}] = 0.038519$ | Var$[p_{79}] = 0.038519$ | Var$[R_{12}] = 0.000045$ | Var$[R_{12}] = 0.000045$ |
| $E[p_{84}] = 0.25$ | $E[p_{88}] = 0.75$ | $E[R_{13}] = 0.985$ | Var$[R_{13}] = 0.000045$ |
| Var$[p_{84}] = 0.020327$ | Var$[p_{88}] = 0.004088$ | Var$[R_{14}] = 0.000045$ | Var$[R_{14}] = 0.000045$ |
| $E[p_{88}] = 0.10$ | $E[p_{98}] = 0.90$ | $E[R_{15}] = 0.985$ | Var$[R_{15}] = 0.000045$ |
| Var$[p_{88}] = 0.004088$ | Var$[p_{98}] = 0.004088$ | Var$[R_{16}] = 0.000045$ | Var$[R_{16}] = 0.000045$ |

Table 5.17: Parameter values for the model

DTMC presented in Figure 4.12. The mean and variance of non-zero transition probabilities $p_{ij}$ and the mean and variance of component reliabilities $R_i$ are given in Table 5.17. The architecture-based reliability model for this application is presented in Figure 4.13. Table 5.18 presents the values of the mean $E[R]$ and the variance $Var[R]$ of the system reliability for the case study 2.

We also study the contribution of the parameters to the variance of system reliability. As it can be seen from Figure 5.16, 53.67% of the variance is due to only two parameters $R_{10}$ and $R_2$. In this case, transition probabilities contribute only 4.47% to the variance of system reliability, that is, 95.53% of the variance is due to component reliabilities.
Table 5.18: The mean and variance of the system reliability for the model

<table>
<thead>
<tr>
<th></th>
<th>First order Taylor series</th>
<th>Second order Taylor series</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.8299</td>
<td>0.8277</td>
</tr>
<tr>
<td>Variance</td>
<td>0.0041</td>
<td>0.0043</td>
</tr>
<tr>
<td>$C_R$</td>
<td>0.0770</td>
<td>0.0791</td>
</tr>
</tbody>
</table>

Figure 5.16: Parameters contribution to the variance of system reliability
5.4 Uncertainty Analysis based on Monte Carlo Simulation

5.4.1 European Space Agency Software

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

![Uncertainty analysis](image)

Figure 5.17: Uncertainty analysis for version A

In Figure 5.17 we present how the uncertainty in parameters of version A affects the system reliability. The estimation of the mean reliability converges in approximately 3000 iterations. In
addition to the mean reliability, we have estimated several other characteristics of the system reliability distribution [15]: coefficient of variability $C_R$ which is related to the spread of a distribution, skewness which is related to the lean of a distribution, and kurtosis which is related to the peakedness of a distribution. Note that these measures are relative which allows us to compare different distributions.

The frequency chart presented in Figure 5.17 gives the probability (frequency) of occurrence for different values of system reliability. The range of the reliability is $[0.3939, 0.9709]$ and the distribution is skewed to the left. We have also done a distribution fitting for system reliability. In this case Beta distribution with parameters $\alpha = 19.7994$, $\beta = 5.8986$ and scale = 0.9855 is the closest fit to the frequency data based on the chi–square fitness test. Further, we have estimated the percentiles, i.e., certainty bands. Another interesting observation can be made from the sensitivity chart in Figure 5.17. We calculate sensitivity by computing rank correlation coefficients between every parameter and system reliability. High correlation coefficient means that the parameter has a significant impact on software reliability (both through its uncertainty and its model sensitivity). Positive coefficients indicate that an increase in the parameter is associated with an increase in the reliability. Negative coefficients imply the reverse situation. In this example, even though the variation of component reliabilities is small, they play a critical role in the variation of system reliability. As can be seen from the sensitivity chart 94.8% of the reliability variation is due to reliabilities $R_1$ and $R_2$.

In Figure 5.18 we present the results for version $B$ obtained by varying transitions probabilities and component reliabilities. The reliability range in this case is $[0.5376, 0.9956]$ and the reliability distribution is also skewed to the left. As it can be seen from the values given in Table 5.19 the reliability distribution of version $B$ has higher mean and smaller variance. Further, it is more skewed to the left (that is, concentrated to the right), with higher peak. Also, certainty bands for version $B$ are narrower than for version $A$. The system reliability is still more sensitive to the variation of the component reliabilities, although with smaller contribution to

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

\[2\] For the sake of comparison the displayed range is $[0.5, 1]$
<table>
<thead>
<tr>
<th>Version</th>
<th>Mean</th>
<th>Coefficient of variability</th>
<th>Skewness</th>
<th>Kurtosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.7593</td>
<td>0.1056</td>
<td>-0.5155</td>
<td>3.2786</td>
</tr>
<tr>
<td>B</td>
<td>0.8746</td>
<td>0.0699</td>
<td>-0.8520</td>
<td>3.8429</td>
</tr>
</tbody>
</table>

Table 5.19: Characteristics of reliability distributions for versions A and B

the variance (88.6%).

Figure 5.18: Uncertainty analysis for version B

Our next numerical example illustrates the uncertainty analysis for the hypothetical example presented in Figure 4.4. Table 5.20 compares the characteristics of the system reliability distribution for three different values of transition probability $p_{21}$ associated with the arc forming a loop in the model. As can be seen from Table 5.20, the mean system reliability decreases for higher values of transition probability $p_{21}$. In addition, we see that for higher values of $p_{21}$ the coefficient of variability is increasing, distribution skewness is moving to the right, and the peak
<table>
<thead>
<tr>
<th>$p_{21}$</th>
<th>Mean</th>
<th>Coefficient of variability</th>
<th>Skewness</th>
<th>Kurtosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.7305</td>
<td>0.1156</td>
<td>-0.3954</td>
<td>2.9862</td>
</tr>
<tr>
<td>0.5</td>
<td>0.6202</td>
<td>0.1779</td>
<td>-0.2560</td>
<td>2.8662</td>
</tr>
<tr>
<td>0.95</td>
<td>0.4136</td>
<td>0.3439</td>
<td>0.0938</td>
<td>2.6479</td>
</tr>
</tbody>
</table>

Table 5.20: Characteristics of reliability distribution for the hypothetical example

It is obvious from Figures 5.19 and 5.20 that the characteristics of the system reliability distribution are very sensitive to the values of modeling parameters. We already knew from the point estimates [11] that the system reliability for $p_{21} = 0.95$ is significantly lower that for $p_{21} = 0$. In addition, from uncertainty analysis we observe that the reliability distribution for $p_{21} = 0.95$ is widely spread and has wider certainty bands compared to $p_{21} = 0$. Also, the parameters
contribution to the variance of system reliability changes significantly. Thus, in the case of $p_{21} = 0$ reliabilities $R_1$ and $R_2$ contribute 98.0% to the variance of system reliability, while in the case of $p_{21} = 0.95$ they contribute only 41.3%. Even more, when $p_{21} = 0.95$ the highest effect on the system reliability is coming from transition probability $p_{1E} = 1 - p_{12}$ which contributes 51.7% to the variance of system reliability. These results clearly illustrate the usefulness of uncertainty analysis and motivate its systematic use for software reliability prediction.

### 5.4.2 Case Study Adopted From Literature

Here, we consider model from Cheung’s paper [6] and study the effect of both sources of uncertainty (operational profile and components reliabilities) on system reliability. Frequency chart, distribution fitting, certainty bands, and sensitivity chart for this model are given in Figure 5.21. The reliability range in this case is $[0.4623, 0.9677]$ with a mean 0.8276, variance 0.0041 and
Figure 5.21: Uncertainty analysis for case study from Cheung’s paper
distribution skewed to the left. From the sensitivity chart it can be seen that 17 parameters contribute less than 1% each and 8 parameters contribute between 1% – 5% each to the variance in software reliability. Further, the 4 parameters (out of 29) with the highest sensitivity ranking \((R_5, R_8, R_2, R_{10})\) contribute 77.2% of the variance in the reliability. Among transition probabilities \(p_{84}\) has the highest sensitivity ranking with 4.9% contribution to the variance in software reliability.
Chapter 6

Conclusion

In this thesis an architecture-based methodology is presented for uncertainty analysis of software reliability throughout the software life cycle. Software architecture, software usage and software failure behavior are necessary to estimate software reliability using architecture-based models. Within this methodology, several different approaches are presented to build the operational profile, as well as several approaches to estimate component reliabilities. But, there exists a lot of uncertainty around these parameters. Since, architecture-based approach provides insight into the dynamic behavior of the software execution, we studied the effect of different parameters on the uncertainty of software reliability. Four proposed methods - entropy, perturbation theory, method of moments and Monte Carlo simulation are considered for uncertainty analysis.

The choice of the method for uncertainty analysis in software reliability that is most appropriate for a given application depends on the criteria such as data requirements, reliability measures derived, and accuracy of the solution. The basic characteristics of different methods for uncertainty analysis are summarized in Table 6.1.

Source entropy quantifies the uncertainty of the operational profile and architecture-based software reliability models. The method is applied on two different case studies: European space
<table>
<thead>
<tr>
<th>Method</th>
<th>Data requirements</th>
<th>Reliability measures</th>
<th>Accuracy of the solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy</td>
<td>Point estimate</td>
<td>N/A</td>
<td>Exact analytical solution</td>
</tr>
<tr>
<td>Perturbation Theory</td>
<td>Point estimate</td>
<td>N/A</td>
<td>Exact analytical solution</td>
</tr>
<tr>
<td>Method of moments</td>
<td>Moments of components reliabilities</td>
<td>Moments</td>
<td>Approximate method; analytical solution</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• accuracy may be increased by higher order Taylor series</td>
</tr>
<tr>
<td>Monte Carlo simulation</td>
<td>• Distribution functions of transition probabilities and component reliabilities</td>
<td>• Distribution</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Generation of random numbers</td>
<td>• Moments</td>
<td>•Moments</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Percentiles</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Parameters contribut</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ion to the variance</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: Comparison of methods for uncertainty analysis

agency and Real-time control system. The results show that software systems that have a uniform operational profile and moderate component reliabilities are more uncertain, and thus would require more testing efforts. Further, the architecture-based methodology helps us to estimate the expected execution rate and uncertainty of each component using the theory of Markov chains and conditional entropy respectively.

Perturbation theory is another method for uncertainty analysis that can be used to study how the change in the operational profile affects the expected execution rates of software components. This method is applied on three case studies: European space agency, an E-commerce application and Real-time control system. Results show that the stability of operational profile can be studied by looking at the small stationary probabilities. Thus, if a small stationary probability is relatively insensitive, then the operational profile is stable in both absolute and relative sense. Further, component with small execution rates are the ones which are most sensitive to
changes in the operational profile. This analysis is most valuable for software verification and validation, where rarely executed components usually handle critical functionality such as for example exception handling or recovery.

Uncertainty analysis using Entropy and Perturbation theory methods doesn’t provide any measures of software reliability. But, Method of moments and Monte carlo simulations are the two other methods that assess the uncertainty by providing richer measures of software reliability than the traditional point estimate. Method of moments and Monte carlo simulation are applied on the following case studies: European space agency application and Cheung model. It is obvious that the estimated values of the system reliability moments provide more information than point estimate. Thus, systems that have reliability with smaller variance will have higher confidence in the reliability estimates.

Uncertainty analysis of the operational profile and software reliability is not only important but also necessary, if we want to make predictions early in software life cycle, keep track of software evolution and certify the reliability of component-based systems. Another important contribution of the thesis is that, instead of the point estimate of the reliability equal to 1, it enables us to quantify the mean and variance of the reliability of the component-based systems that do not fail during testing. This analysis is very important for high assurance systems.

The main focus of our future work is to validate the methodology and methods by applying them on open source applications and compare the methods for uncertainty analysis. The comparison of methods for uncertainty analysis help us to develop sound guidelines for choosing the most appropriate method depending on data requirements, derived reliability measures, accuracy of the solution and scalability with respect to the number of components.
Bibliography


