

**FY2002 UNIVERSITY SOFTWARE INITIATIVE PROPOSAL
FOR THE
NASA SOFTWARE IV&V FACILITY**

**Initiative Title: Sensitivity of Software Reliability to Operational Profile
Errors: Architecture-Based Approach**

Initiative ID: Project 10000559, Task 24g, Award 1000625GR

September 2002 deliverable:

Report on

Application of the methodology for uncertainty analysis on the case studies

PI: Katerina Goseva – Popstojanova

Student: Sunil Kumar Kamavaram

*LANE Department of Computer Science and Electrical Engineering
West Virginia University*

Report Summary

In this report we first present a methodology for uncertainty analysis of software reliability that can be used throughout the software life cycle. Then, we describe in details several methods for uncertainty analysis that can be used within this methodology: entropy, method of moments, and Monte Carlo simulation. The methodology and all methods for uncertainty analysis are applied and validated on a case study based on the software developed for the European Space Agency. Also, we apply our methodology on the NASA's Hub Control System (HCS) from the International Space Station (ISS) based on the available software artifacts for this case study.

Table of Contents

1. Introduction	4
2. Related work and uniqueness of the research work	4
3. Methodology for uncertainty analysis	5
3.1. Software architecture	6
3.2. Components failure behavior	7
3.3. Combining software architecture with failure behavior	8
3.4. Uncertainty analysis	8
4. Description of the case studies	9
4.1. Case study from the European Space Agency	9
4.2. Case study from NASA: Hub Control Software	12
5. Uncertainty analysis based on entropy	18
5.1. Application of the entropy on the ESA case study	19
5.1.1. Uncertainty of ESA operational profile	19
5.1.2. Uncertainty of ESA software reliability	20
5.1.3. Uncertainty of ESA components	21
5.2. Application of the entropy on the NASA's HCS case study	22
6. Uncertainty analysis based on method of moments	26
6.1. Application of the method of moments on the ESA case study	28
7. Uncertainty analysis based on Monte Carlo simulation	29
7.1. Application of the Monte Carlo method on the ESA case study	31
8. Conclusion	37
References	37

1. Introduction

A number of analytical models have been proposed to address the problem of quantifying software reliability. One group of models is focused on modeling reliability growth during testing phase [Farr96]. These so called black - box models treat the software as monolithic whole, considering only its interactions with external environment, without an attempt to model the internal structure. Black - box models are clearly inappropriate for large component - based systems. For these systems, we need to use a white - box approach that takes into account the information about the architecture of software made out of components. An extensive survey on architecture -- based software reliability models, including their assumptions, usefulness, and limitations is presented in [Goseva01a].

Two important questions arise with respect to predications of software reliability based on models. The first question addresses the appropriateness of the model. Thus, the model could be inappropriate because its assumptions may not hold in practice. The second question addresses the accuracy of parameters values. Parameters can be estimated using the field data obtained during testing or operational usage of the software, historical data for products with similar functionality, or reasonable guesses based on the specification and design documentation. In practice, there is a lot of uncertainty around parameters because they rarely can be estimated accurately. The traditional way of estimating software reliability by plugging point estimates of the unknown parameters into the model [Farr96], [Goseva01a] is not appropriate since it discards any variance due to uncertainty of the parameters. In order to answer the question how parameters uncertainties propagate into reliability estimation, uncertainty analysis is necessary.

In this report we first present the methodology for uncertainty analysis of software reliability. Then, we apply and validate our methodology on the case studies from European Space Agency and NASA. The rest of the report is organized as follows. The discussion of the related work is presented in Section 2. The basic concepts of the proposed methodology for uncertainty analysis are presented in Section 3. The description of the case studies is given in Section 4. The entropy, method of moments, and Monte Carlo simulation as methods for uncertainty analysis are described and applied on the case studies in Sections 5, 6, and 7 respectively. Finally, the concluding remarks are presented in Section 8.

2. Related work and uniqueness of the research

Traditionally, the most common method for uncertainty analysis in software reliability is conducting sensitivity studies. Thus, sensitivity of the software reliability estimation to errors in the operational profile has been investigated in the context of black - box reliability growth models [Chen94], [Musa94], [Pasquini96]. Sensitivity studies of software reliability estimates obtained using architecture - based models have been presented in [Cheung80], [Siegrist88]. In these studies the authors assumed fixed known values for the transition probabilities and derived the sensitivity of the system reliability with respect to the reliability of each component. However, any inaccuracy in the operational profile directly will affect transition probabilities among components. Therefore, in [Goseva01b] we presented the sensitivity studies of software reliability with respect to the operational profile (i.e., transition probabilities) and component reliabilities.

In addition to sensitivity studies, there have been several attempts to quantify the variability of software reliability. In [Miller92] 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 [Adams96] for the software with partitioned input domain. However,

in this work it was recognized that there is uncertainty in the estimations of the reliability for each partition (using Beta prior distribution), as well as uncertainty in the probability of using each partition (using Dirichlet distribution). In [Singh01] 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 [Leung97] 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 [Whittaker93] Markov analysis of software specifications was presented and entropy was used as a measure of uncertainty. In [Wesslen00] 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 [Hahn94], [Jackson81], [Yin01].

In this report we propose a methodology for uncertainty analysis of architecture - based software reliability models suitable for large complex component - based applications and applicable throughout the software life cycle. The methodology addresses the parameter uncertainty problem and enables us to study how the uncertainty of parameters propagates in the system reliability. Within this methodology we are considering several different methods for uncertainty analysis. So far we have used entropy [Kamavaram02], methods of moments [Goseva02b], and Monte Carlo simulation [Goseva02c] for uncertainty analysis.

The general goal of our 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 our work, because with the growing emphasis on reuse developers can not afford to stay away from reliability certification.

3. Methodology for uncertainty analysis

The architecture - based approach for 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 software architecture (structure of component interactions), software usage described by the operational profile (relative frequencies of component interactions determined by transition probabilities), and software failure behavior (component reliabilities or failure rates). In [Goseva01b] we have shown that the architecture – based software reliability model presented in [Cheung80] provides system reliability estimates close to the actual measured reliability, that is, we have validated the model appropriateness. In this report we propose a methodology for uncertainty analysis (see Figure 1) and focus on the assessment of the uncertainty in software reliability due to uncertainty of modeling parameters. The basic concepts of our work were presented in [Goseva02a]. Here, we describe the proposed methodology in detail.

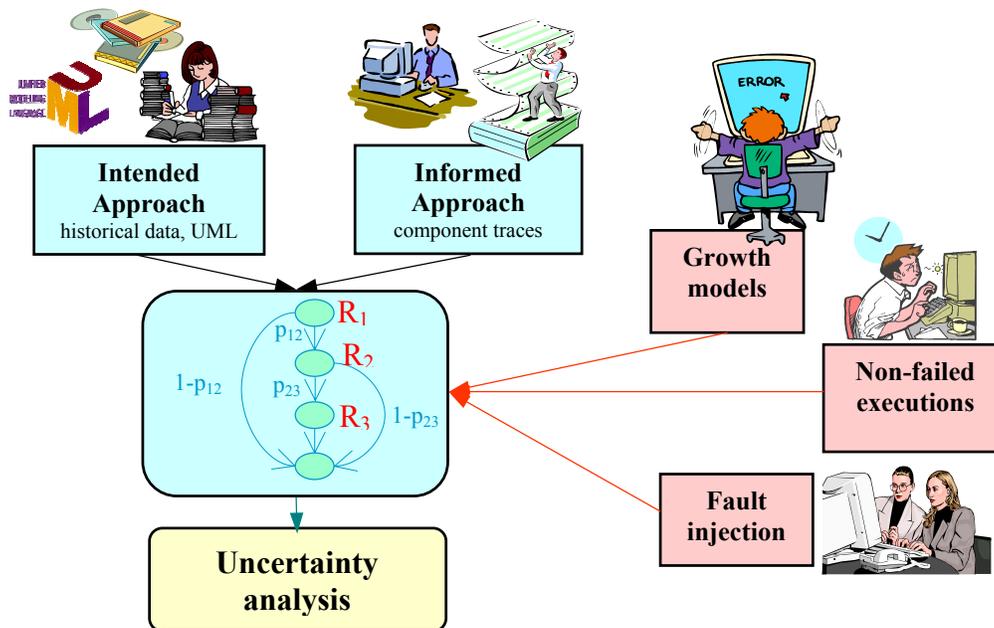


Figure 1. Methodology for uncertainty analysis of software reliability

3.1. Software architecture

Software behavior with respect to the manner in which different components interact is defined through the software architecture. We use state - based approach to build the architecture - based software reliability model [Goseva01a], [Goseva01b]. This approach uses the control flow graph to represent software architecture. The states represent active components and the arcs represent the transfer of control. Based on the assumption that the transfer of control between components has a Markov property, the architecture is modeled with a discrete time Markov chain (DTMC) with a transition probability matrix $P = [p_{ij}]$, where

$p_{ij} = \Pr\{\text{program transfers the control from component } i \text{ to component } j\}$. 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 build 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 components interactions, that is, transition probabilities which are clearly dependent on the operational profile. During the early phases of software development, dynamic software behavior can be captured using 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 software development. We base our estimates on historical data from similar products or on high level information about software architecture

obtained from specification and design documents. Since, UML is rapidly becoming a standard for software development, in intended approach we are looking into the UML annotations such as use cases and sequence diagrams [UML]. Use case diagrams provide graphic description of how external entities interact with the system. Sequence diagrams depict how group of components interact in a use case. Each sequence diagram shows a number of components and the how many times the messages are exchanged between them.

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

Dynamic information in software architecture clearly depends on the software usage, that is, the operational profile. Operational profiles have been developed successfully for the applications such as real-time telecommunication systems where the use of the software is predictable because it is related to identifiable events due to human activity [Musa93]. In general, the estimation of a trustworthy operational profile is difficult because it requires anticipating the field usage of the software and a priori knowledge about the application and system environments. A typical example would be a flight control system of a spacecraft in which very 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 other cases, a single operational profile is not sufficient to describe the use of the product by different users. Because the effort required to derive 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 above difficulties, problems could arise due to the changes of the operational profile during the development and field usage of the software. Thus, software systems evolve because functions are added or modified. As a consequence, 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. Therefore, it is important to conduct uncertainty analysis due to uncertainty in the operational profile estimation.

3.2. Components failure behavior

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

Several techniques for estimating component's reliability have been proposed. **Software reliability growth models** can be applied to each software component exploiting component's failure data obtained during testing [Farr96]. However, due to the scarcity of failure data it is not always possible to use software reliability growth models. Another possibility is to estimate component's reliability from explicit consideration of **non-failed executions**, possibly together with failures [Miller92], [Nelson73]. In this context, testing is not an activity for discovering faults, but an independent validation activity. The problem that arises with these models is the large number of executions

necessary to establish a reasonable statistical confidence in the reliability estimate. Finally, one can use **fault injection technique** to estimate component's reliability [Goseva01b]. 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.

3.3. Combining software architecture with failure behavior

The presented methodology for uncertainty analysis can be applied to any architecture - based software reliability model that has a close form solution for the system reliability. In this report we use the model first presented in [Cheung80] which uses composite method to combine software architecture with failure behavior. Two absorbing states C and F are added to the DTMC, representing the correct output and failure respectively. The transition probability matrix P is modified to \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 directed 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 directed edge to failure state F with transition probability $(1 - R_i)$. The reliability of the program is the probability of reaching the absorbing state C of the DTMC. Let Q be the matrix obtained from \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 $R = S(1, n)R_n$.

3.4. Uncertainty analysis

Using the model described in Section 3.3 we obtain the expression for system reliability R as 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 in components failure data exists (as it usually does) then a significant uncertainty in calculated system reliability exists. Therefore, the traditional approach of computing the point estimate of the system reliability by plugging point estimates of component reliabilities into the model is not appropriate. In order to answer the question how parameters uncertainties propagate into overall system reliability, uncertainty analysis is necessary. To conduct uncertainty analysis, we can treat unknown parameters as random variables and quantify the uncertainty of system reliability. In this case, the system reliability is also a random variable.

Different methods can be applied for synthesizing uncertainty in system reliability from uncertainties in component reliabilities and transition probabilities (see Figure 2). The choice of the method will depend on criteria such as data requirements, reliability measures derived, accuracy of the solutions, and scalability with respect to the number of components. In this report we use entropy, method of moments, and Monte Carlo simulation for uncertainty analysis of software reliability.

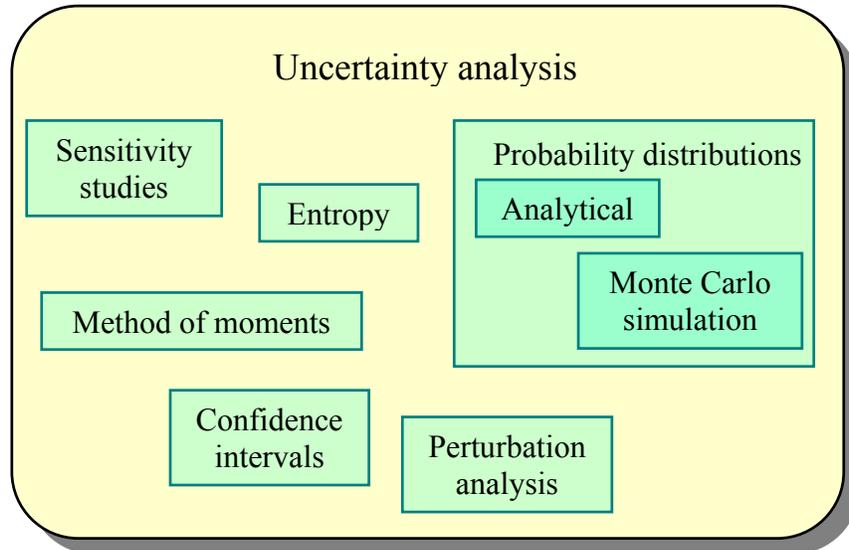


Figure 2. Methods for uncertainty analysis in software reliability

4. Description of the case studies

We apply and validate our methodology on case studies from European Space Agency and NASA. In this section we briefly describe these case studies.

4.1. Case study from the European Space Agency (ESA)

The application from the European Space Agency [Goseva01b] 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:

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

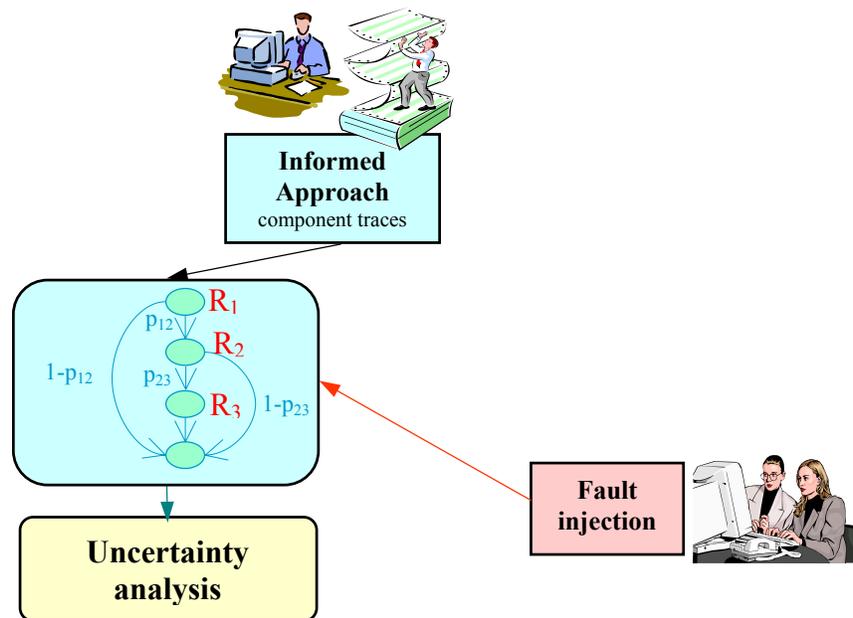


Figure 3. Special case of the methodology used for the ESA case study

DTMC that represents software architecture is shown in Figure 4. Components 1, 2, and 3 correspond to the Parser, Computational, and Formatting subsystems respectively. State E represents the completion of execution. The choice for the decomposition was made in order to reach a tradeoff between number of components, their size, and the ability to collect data needed for use in the model.

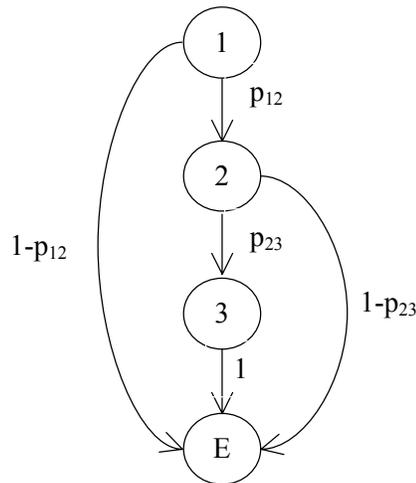


Figure 4. Software architecture for the ESA case study

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 from component i to component j , and $n_i = \sum_j n_{ij}$.

When the outputs of the faulty version and the oracle disagreed it was necessary to determine the component that has failed. Identification of the fault responsible for the failure 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 $R_i = 1 - \lim_{n_i \rightarrow \infty} \frac{f_i}{n_i}$, 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 p_{ij} and component reliabilities R_i for both faulty versions are given in Table 1.

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

Table 1. Transition probabilities and component reliabilities for versions A and B

DTMC presented in Figure 5 is a composite state based model of this application. The expression for system reliability obtained using the model is given by

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

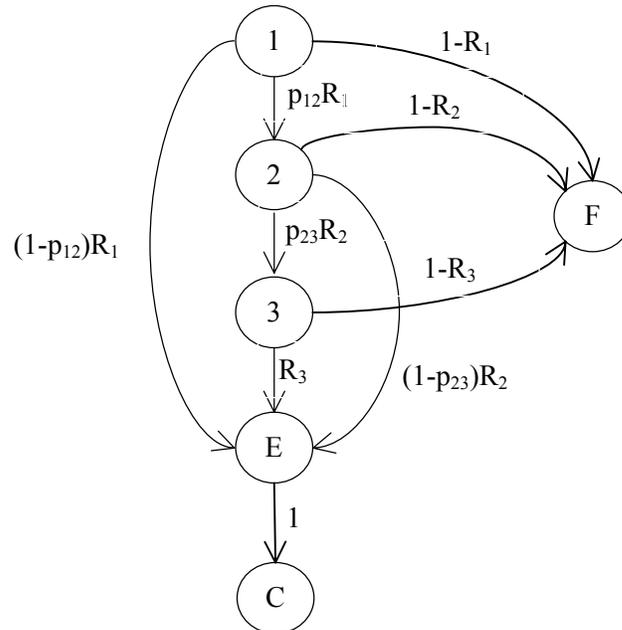


Figure 5. Architecture-based software reliability model for the ESA case study

As it can be seen from Table 2, the architecture - based software reliability model gives accurate estimations compared to the actual reliability for each of the faulty versions which validates the appropriateness of this model for software reliability estimation.

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

Table 2. Comparison of the results

We also consider a hypothetical example of software architecture given in Figure 6 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.

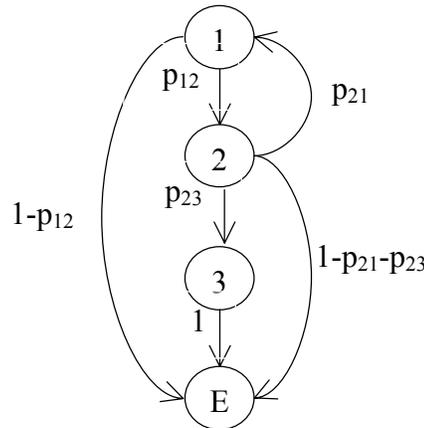


Figure 6. Software architecture for the hypothetical example

For the example in Figure 6 the system reliability obtained using the model described in Section 3.3 is given by

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

4.2. Case study from NASA: Hub Control Software

Our case study from NASA is the Hub Control Software (HCS). This is Computer Software Configuration Item (CSCI) resident in the Hub Control Zone Multiplexers/Demultiplexers (HCZ MDMs) which are installed in the Node 3 Module of the ISS (International Space Station). For this case study we only had available the UML use case diagram and sequence diagrams for each use case. Therefore, we are using the intended approach to build software architecture. We build DTMCs using UML sequence diagrams that present software components used for given scenario and the 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 from component i to all other components that are present in the sequence diagram.

Data for the components failure behavior were not available for the HCS case study. Also, we didn't have historical failure data for similar projects. Therefore, for the HCS case study we only apply the uncertainty analysis of the operational profile based on entropy. Figure 7 presents the special case of our methodology used for the HCS case study.

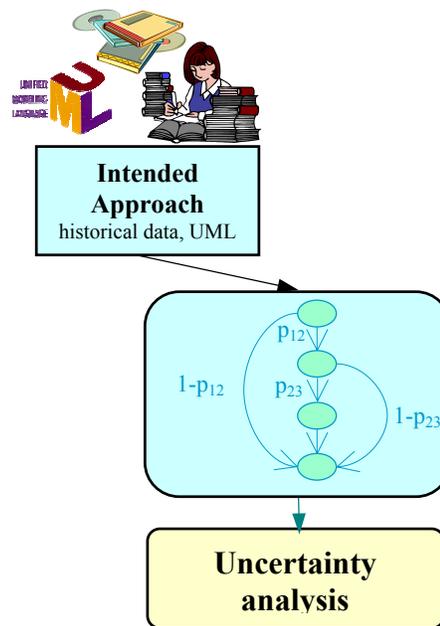


Figure 7. Special case of the methodology used for HCS case study

Figure 8 shows the main use case diagram and all the relationships among the use cases and the actors. Each use case is realized by at least one sequence diagram (or scenario).

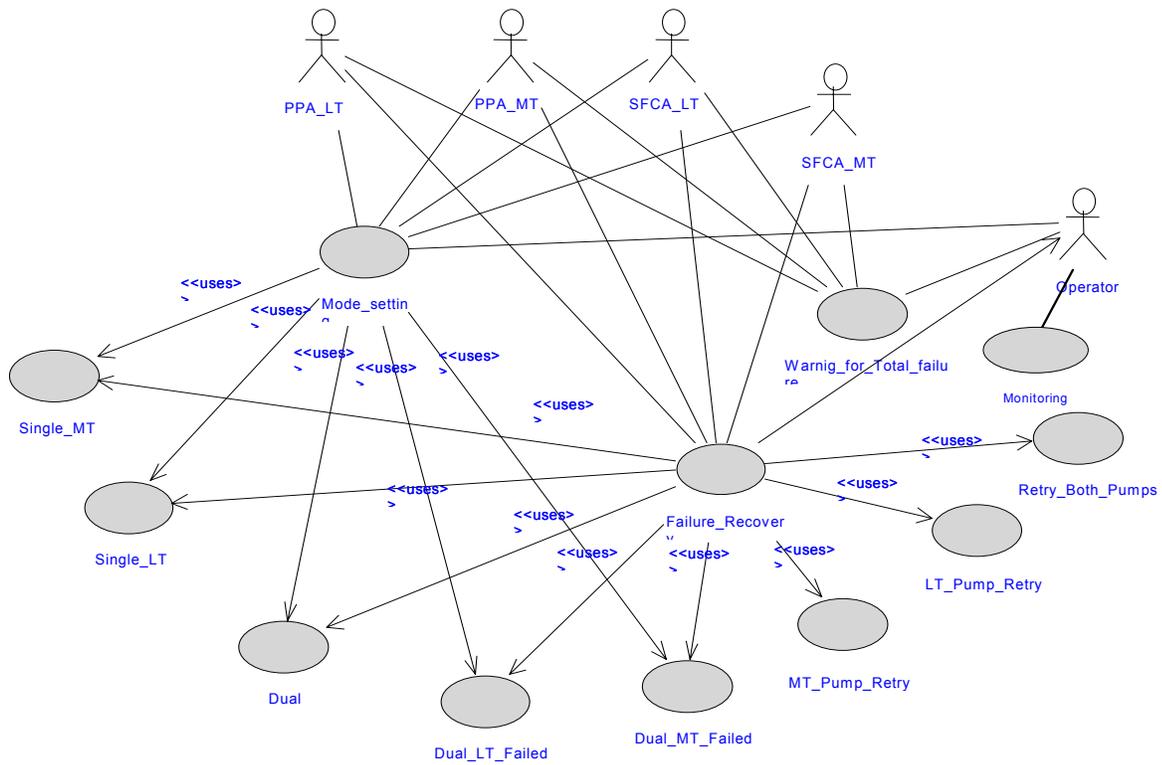


Figure 8. Use case diagram of the HCS case study

Out of the available sequence diagrams for the use cases in the HCS study we use the *Both Pumps Retry*, *Dual*, and *LT Pump Retry*. Figure 9 shows the sequence diagram of the *Both Pumps Retry* scenario.

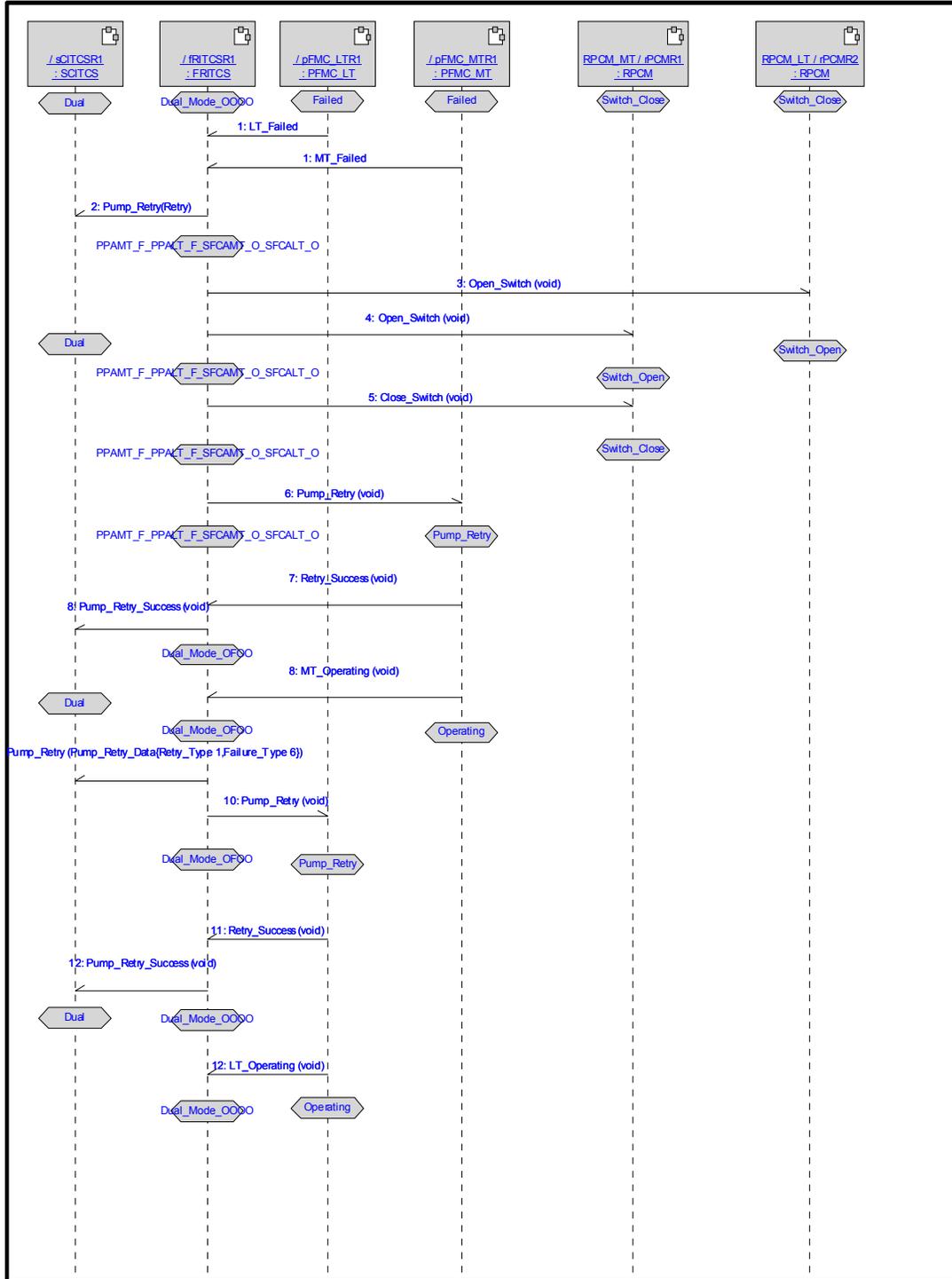


Figure 9. Sequence diagram of the Both Pumps Retry scenario

Analyzing the sequence diagram of the *Both Pumps Retry* scenario given in Figure 9, we construct the DTMC that represents the software execution behavior as shown in Figure 10.

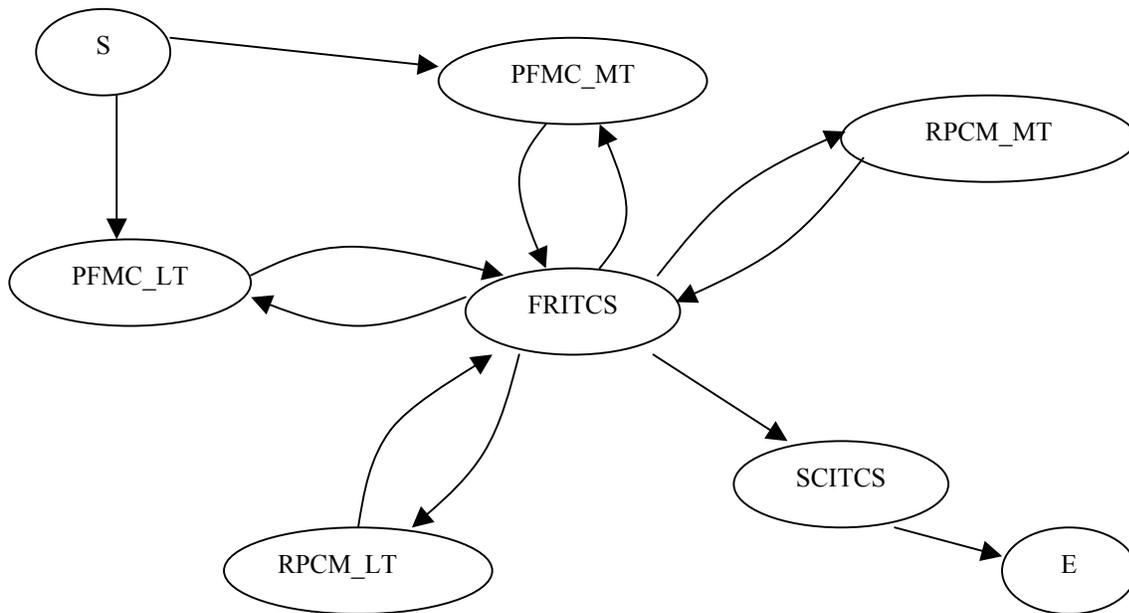


Figure 10. DTMC for the Both Pumps Retry scenario

Transition probability matrix for the *Both Pumps Retry* scenario is given by

$$P = \begin{matrix} & \begin{matrix} S & 1 & 2 & 3 & 4 & 5 & 6 & E \end{matrix} \\ \begin{matrix} S \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ E \end{matrix} & \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1/9 & 1/9 & 0 & 1/9 & 2/9 & 4/9 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{matrix}$$

where S, 1, 2, 3, 4, 5, 6, and E denote the start state, components PFMC_LT, PFMC_MT, FRITCS, RPCM_LT, RPCM_MT, SCITCS, and the end (terminating) state respectively.

DTMC that represents the software execution behavior of the *Dual* scenario is shown in Figure 11.

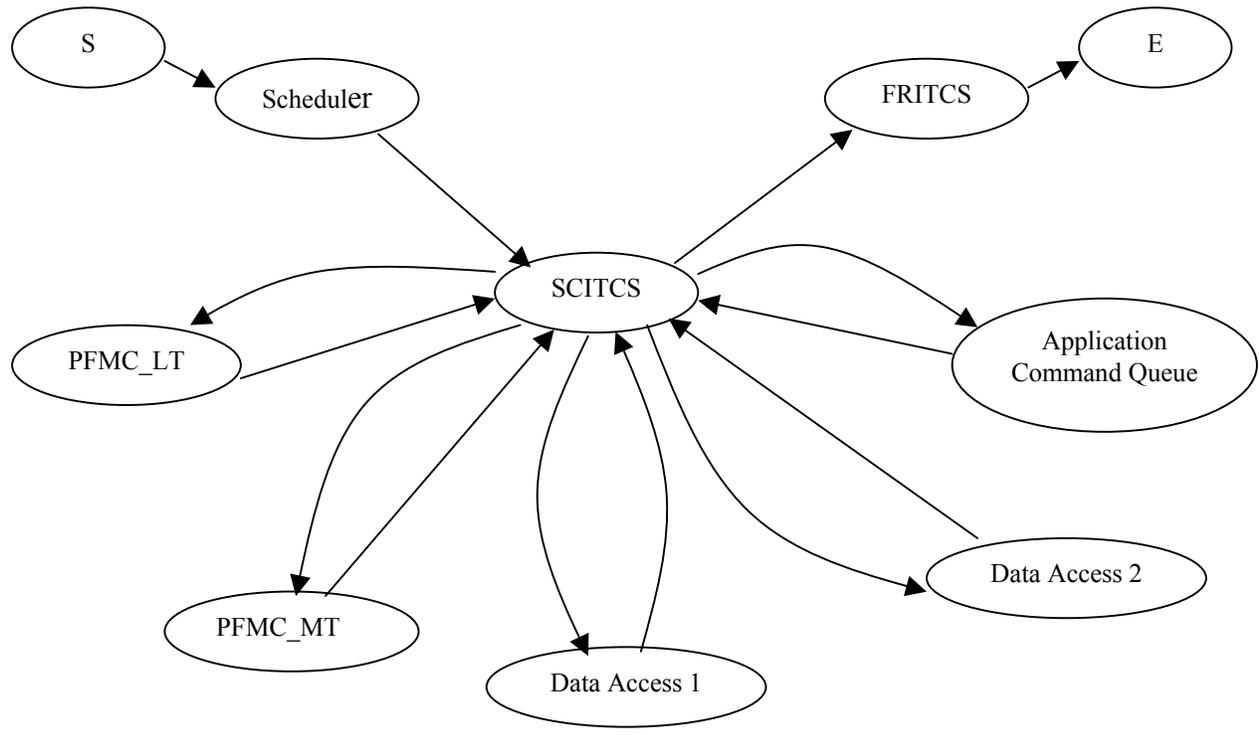


Figure 11. DTMC of the Dual scenario

Transition probability matrix for the *Dual* scenario is given by

$$P = \begin{matrix} & \begin{matrix} S & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & E \end{matrix} \\ \begin{matrix} S \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ E \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/12 & 3/12 & 3/12 & 0 & 1/12 & 2/12 & 2/12 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{matrix}$$

where S, 1, 2, 3, 4, 5, 6, 7, 8, and E represent the start state, components SCITCS, FRITCS, PFMC_LT, PFMC_MT, Scheduler, Application Command Queue, Data Access 1, Data Access 2, and the end state respectively.

DTMC that represents the software execution behavior of the *LT Pump Retry* scenario is shown in Figure 12.

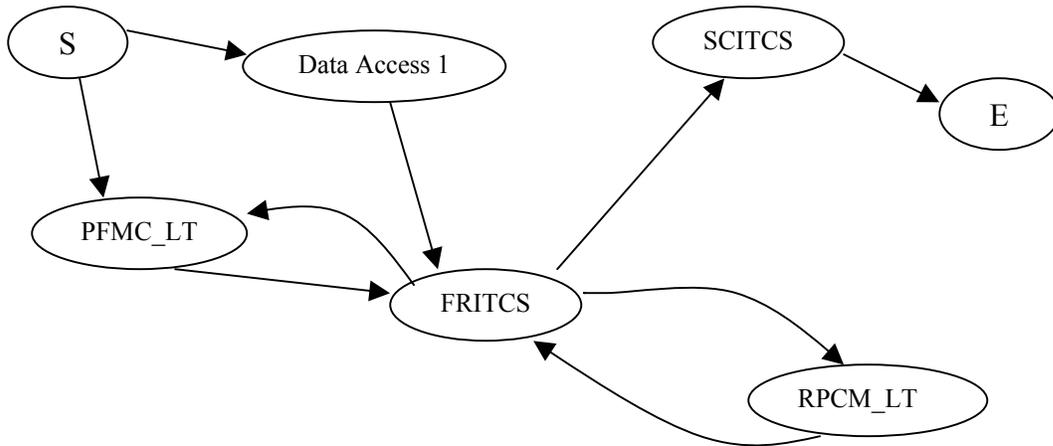


Figure 12. DTMC of the LT Pump Retry scenario

Transition probability matrix for the *LT Pump Retry* scenario is given by

$$P = \begin{matrix} & \begin{matrix} S & 1 & 2 & 3 & 4 & 5 & E \end{matrix} \\ \begin{matrix} S \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ E \end{matrix} & \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1/6 & 0 & 0 & 3/6 & 2/6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{matrix}$$

where S, 1, 2, 3, 4, 5, and E represent the start state, components PFMC_LT, Data Access 1, FRITCS, SCITCS, RPCM_LT, and the end state.

5. Uncertainty analysis based on entropy

In this section, we present the uncertainty analysis using entropy, a well-known concept from information theory [Ash65]. Source entropy that measures the amount of uncertainty inherent in a Markov source is given by [Ash65]

$$H = -\sum_i \pi_i \sum_j p_{ij} \log p_{ij}$$

where π_i is the steady state probability of state i and p_{ij} are the transition probabilities. This single value is related to the number of paths that are statistically typical of the Markov chain. Thus, higher value implies exponentially greater number of typical paths, i.e., more paths exist 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. The range of entropy for a Markov chain with n states is $0 \leq H \leq \log(n)$.

In this report, we use 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 [Ash65]. Thus, the uncertainty of component i (i.e., state i) is given by equation $H_i = -\sum_j p_{ij} \log p_{ij}$. 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 π_i 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.

5.1. Application of the entropy on the ESA case study

5.1.1. Uncertainty of ESA operational profile

Using the equation for the source of entropy H we plot in Figure 13 the variation in the uncertainty of the operational profile as a function of p_{12} and p_{23} . In general, when transition probabilities are close to 0 or 1 the number of 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 Table 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.

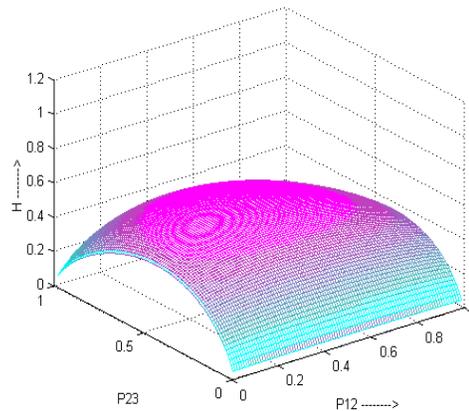


Figure 13. Uncertainty of the operational profile as a function of p_{12} and p_{23}

5.1.2. Uncertainty of ESA software reliability

Next, we consider the uncertainty of software reliability. As in the case of the operational profile, in order to estimate the source entropy of a DTMC given in Figure 5, we consider multiple software executions by adding transitions from both states E and F to the starting state 1. The addition of failure state F to DTMC and the modification of transition probability matrix affect the source entropy. In Figure 14 we illustrate how the uncertainty H and system reliability R vary as functions of p_{12} and p_{23} for versions A and B. As indicated by these figures, considering components failure behavior increases the uncertainty of both versions compared to the uncertainty due to operational profile (see Figure 13). Note that version B, which is more reliable, is less uncertain than version A.

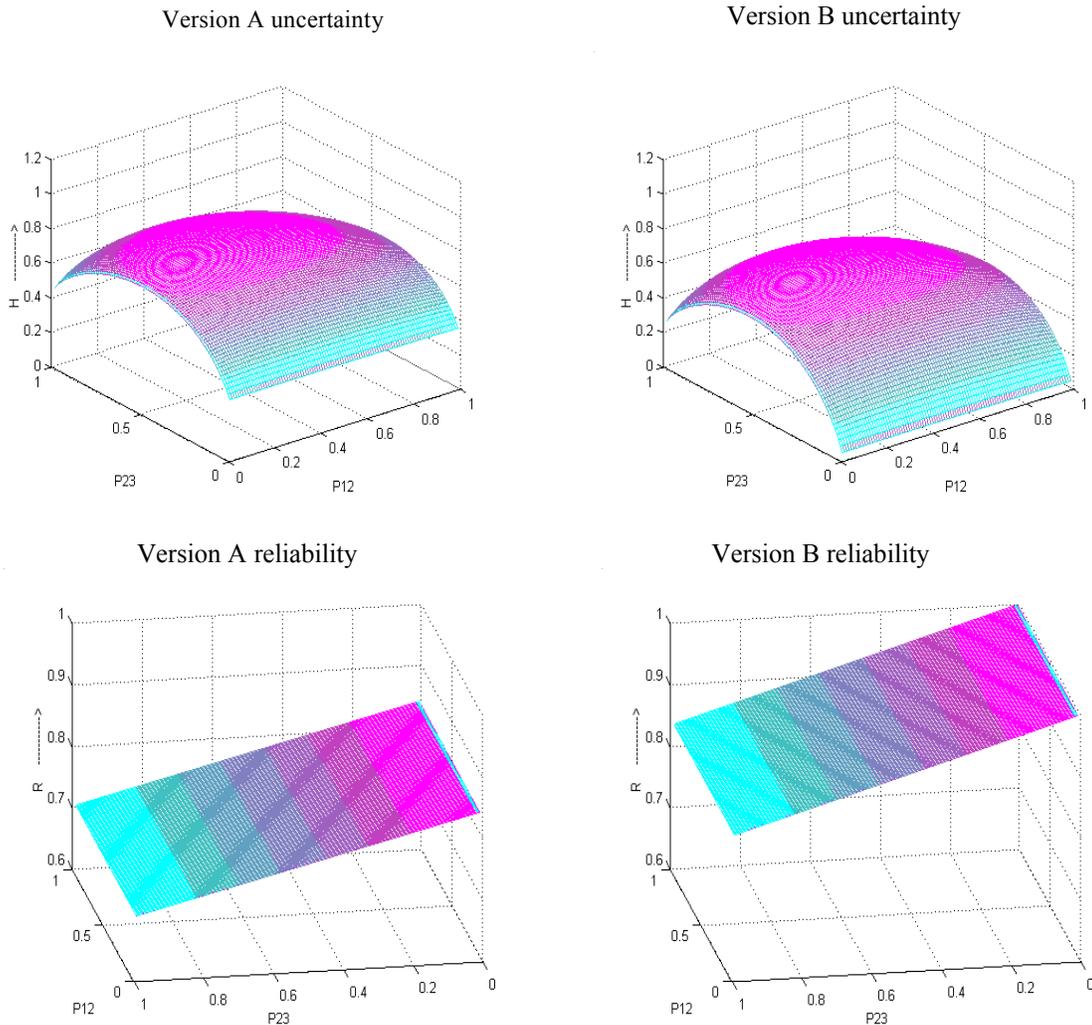


Figure 14. Uncertainty and reliability for versions A and B as functions of p_{12} and p_{23}

5.1.3. Uncertainty of ESA components

Expected execution rates π_i and uncertainties H_i for components in the operational profiles A and B are shown in Table 3 and Figure 15. 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 4. Of course, components that have higher expected execution rate and higher uncertainty will require more testing effort.

	π_i		H_i	
	Version A	Version B	Version A	Version B
State 1	0.3278	0.3085	0.9747	0.8321
State 2	0.1945	0.2271	0.7773	0.8971
State 3	0.1498	0.1560	0	0
State E	0.3278	0.3085	0	0

Table 3. Execution rates and uncertainties of components for operational profiles A and B

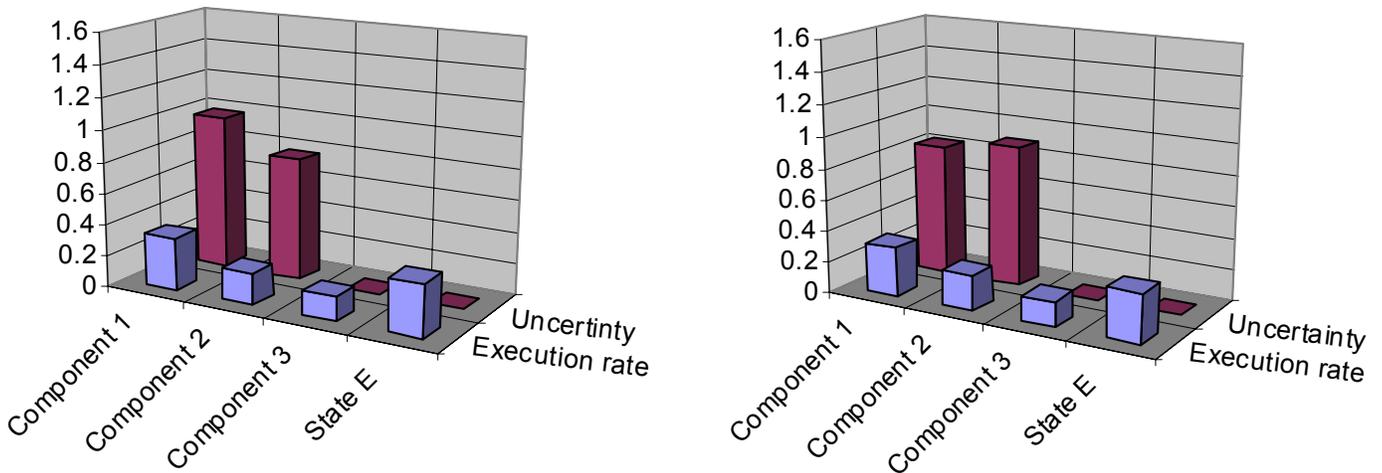


Figure 15. Execution rates and uncertainties of components for operational profiles A and B

Expected execution rates π_i and uncertainties H_i for the components in the software reliability model given in Figure 5 are shown in Table 4 and Figure 16.

	π_i		H_i	
	Version A	Version B	Version A	Version B
State 1	0.3544	0.3166	1.4491	0.8321
State 2	0.1772	0.2332	1.2958	1.3958
State 3	0.1139	0.1336	0	0
State E	0.2694	0.2781	0	0
State F	0.0851	0.0386	0	0

Table 4. Expected execution rates and uncertainties of components for the software reliability model, versions A and B

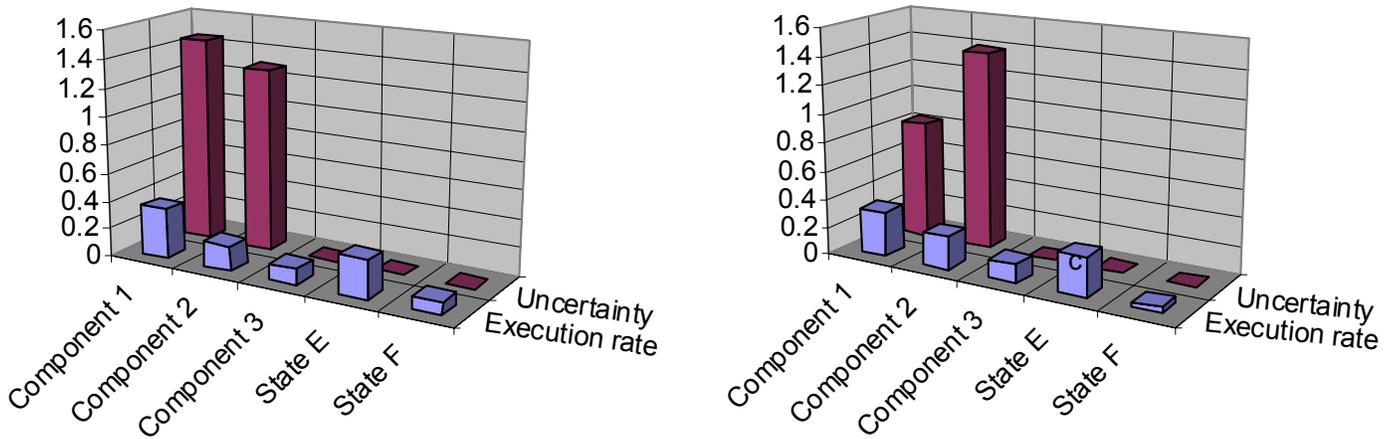


Figure 16. Expected execution rates and uncertainties of the components for the software reliability model, versions A and B

Comparing the results in Table 3 and Table 4 (i.e, Figure 15 and Figure 16), we see that the uncertainty of component 1 in version B remains the same because $R_1 = 1$ results in zero transition probability to failure state. For all other components (1 and 2 in version A and 2 in version B) the component uncertainty increases due to $R_i < 1$ which leads to additional transitions to failure state. In summary, components that have higher expected execution rate, higher component uncertainty and moderate reliability should be allocated more testing effort.

5.2. Application of the entropy on the NASA's HCS case study

In this section we apply entropy as a measure for uncertainty on the HCS case study. DTMC that describes the software architecture of *Both Pumps Retry* scenario given in Figure 10 consists of eight components, including the starting and end state of the application. The uncertainty of the operational profile defined by the transition probability matrix of this scenario is 0.7505. Note that this value of uncertainty is low compared to the maximum uncertainty ($\log_2 8 = 3$) due to the fact that the control flow graph (DTMC) of this scenario is not highly connected, that is, the transition probability matrix is

sparse with many transition probabilities equal to 0. Using the equation for the source of entropy we have plotted the variation of the uncertainty of the operational profile as a function of p_{12} and p_{47} on Figure 17.

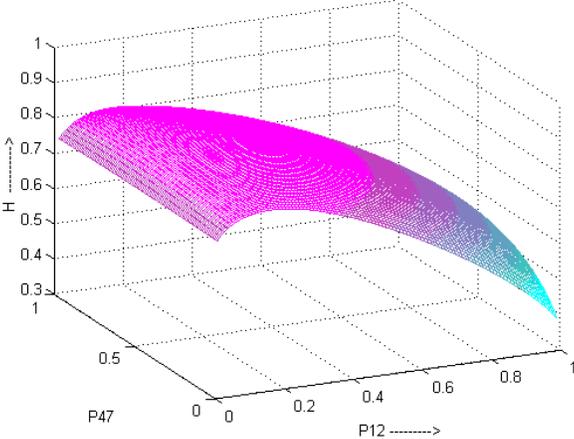


Figure 17. Uncertainty for the operational profile for the Both Pumps Retry scenario

Next we focus on the component uncertainty and expected execution rate for the *Both Pumps Retry* scenario (see use Table 5 and Figure 18).

States	S	PFMC_LT	PFMC_MT	FRITCS	RPCM_LT	RPCM_MT	SCITCS	T
π_i	0.1334	0.1	0.1	0.2999	0.0333	0.0666	0.1334	0.1334
H_i	1	0	0	2.0579	0	0	0	0

Table 5. Execution rates and uncertainties of components in the Both Pump Retry scenario

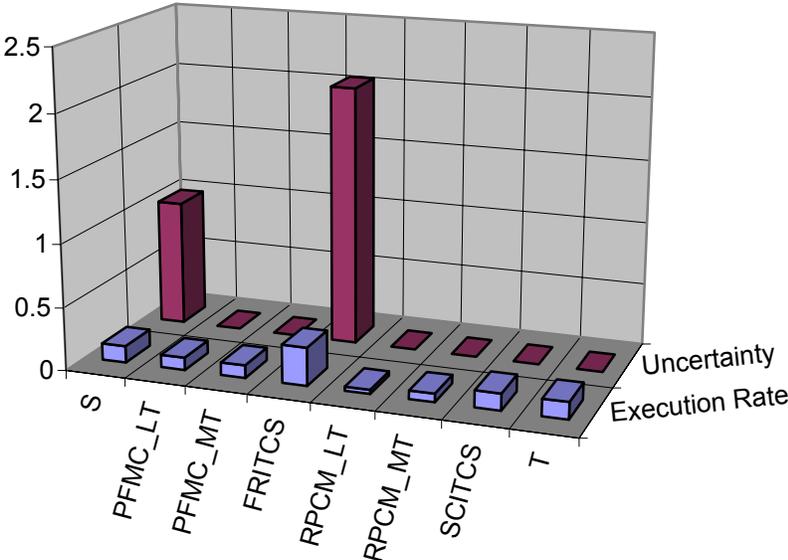


Figure 18. Execution rates and uncertainties of components in the Both Pump Retry scenario

It is obvious from Table 5 and Figure 18 that the component *FRITCS* is executed most often and its uncertainty is the highest. Clearly, *FRITCS* is the most critical components in *Both Pumps Retry* scenario and would require significantly more testing effort than other components.

Next, we consider the *Dual* scenario with a DTMC given in Figure 11. The uncertainty of this scenario estimated using the equation for the source of entropy is 1.093. The entropy is higher than in the case of *Both Pumps Retry* scenario because more components are involved in *Dual* scenario. The variation of the uncertainty of *Dual* scenario as a function of p_{34} and p_{39} is presented in Figure 19.

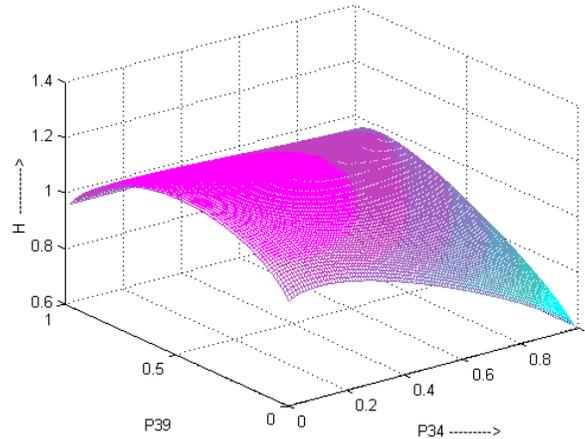


Figure 19. Uncertainty for the operational profile of the Dual scenario

The expected execution rates and uncertainties of components in the *Dual* scenario are presented in Table 6 and Figure 20. In the case of *Dual* scenario component *SCITCS* is executed most often and has the highest uncertainty. On the other side, *FRITCS* which is the most critical component in the *Both Pumps Retry* scenario is not critical for the *Dual* scenario. First, it is executed with the expected rate 0.037 significantly lower than 0.2999 in the *Both Pumps Retry* scenario. Also, the uncertainty of *FRITCS* in *Dual* scenario is 0 (it transfers the control only to the end state).

States	S	Scheduler	SCITCS	PFMC_LT	PFMC_MT	Data Access 1	Data Access 2	App. Comm. Queue	FRITCS	T
π_i	0.037	0.037	0.4445	0.111	0.1111	0.0741	0.0741	0.037	0.037	0.037
H_i	0	0	2.4591	0	0	0	0	0	0	0

Table 6. Execution rates and uncertainties of components in the Dual scenario

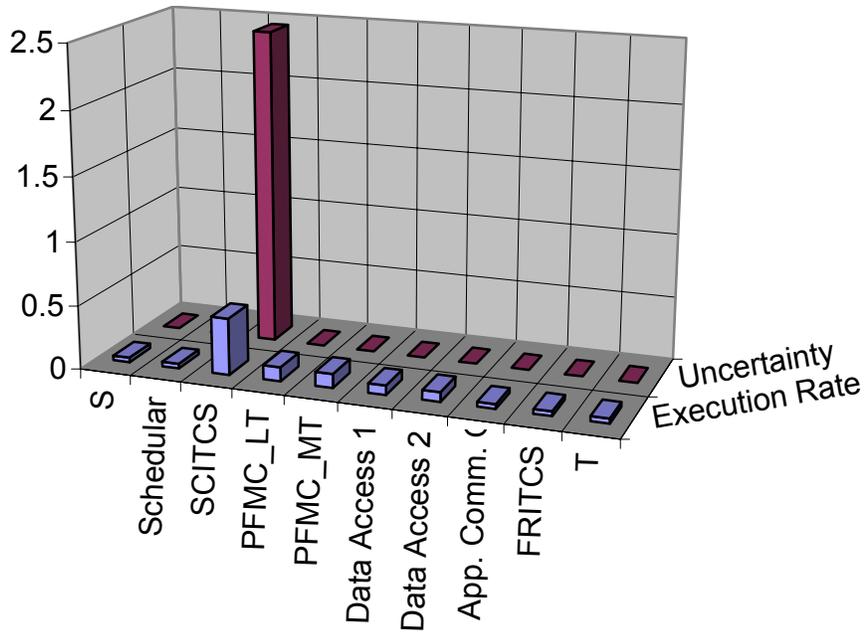


Figure 20. Execution rates and uncertainties of components in the Dual scenario

Let us now consider the *LT Pump Retry* scenario of the HCS case study. The DTMC for the *LT Pump Retry* scenario given in Figure 12 has 7 states (including the starting state S and end state E). Consequently, it has less uncertain (0.5599) than *Both Pumps Retry* scenario. The variation of the uncertainty of *Dual* scenario as a function of p_{12} and p_{45} is presented in Figure 21.

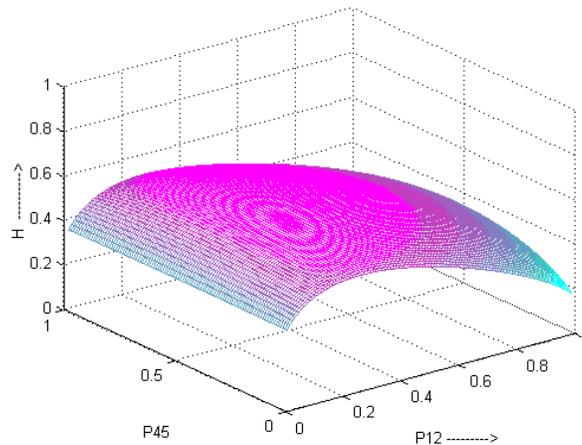


Figure 21. Uncertainty for the operational profile of the LT Pump Retry scenario

The expected execution rates and uncertainties of components in the *Dual* scenario are presented in Table 7 and Figure 22.

States	S	PFMC_LT	Data Access 1	FRITCS	SCITCS	RPCM_LT	T
π_i	0.1429	0.1191	0.0714	0.2857	0.1429	0.0951	0.1429
H_i	1	0	0	1.4595	0	0	0

Table 7. Execution rates and uncertainties of components in the LT Pump Retry scenario

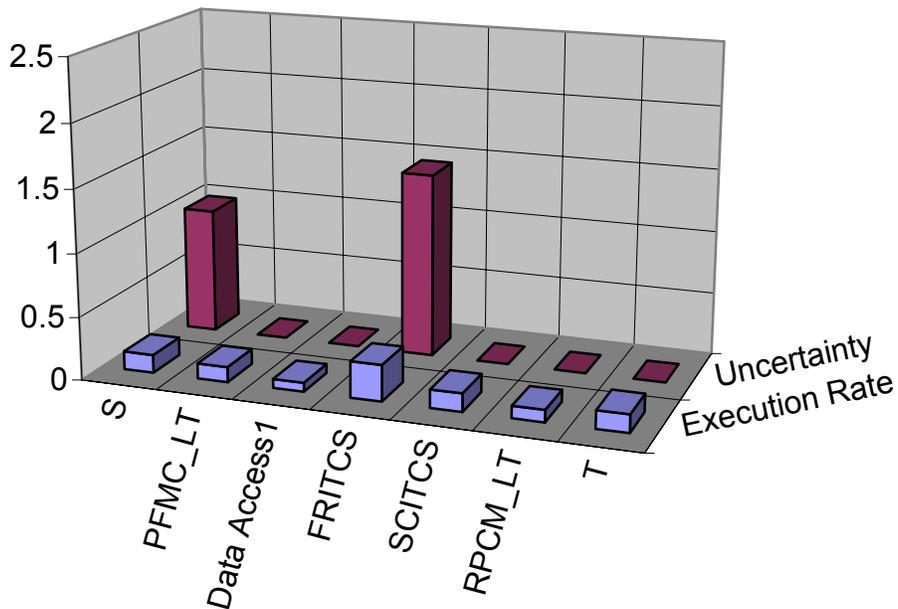


Figure 22. Execution rates and uncertainties of components in the LT Pump Retry scenario

It is obvious from Table 7 and Figure 22 that the *FRITCS* component is the most critical component for *LT Pump Retry* scenario as well as for the *Both Pumps Retry* scenario. The expected execution rates are close (0.2857 and 0.2999 for the *LT Pump Retry* and *Both Pumps Retry* respectively). However, the component uncertainty for the *LT Pump Retry* scenario (1.4595) is lower than for the *Both Pumps Retry* scenario (2.0579). This is due to the fact that *FRITCS* component passes the control to the smaller number of components in the *LT Pump Retry* than in the *Both Pumps Retry* scenario.

6. Uncertainty analysis based on method of moments

In this section we use the method of moments for conducting uncertainty analysis. The method of moments is an approximate method that allows us to generate the moments of system reliability from the moments of component reliabilities. We again use the model first presented in [Cheung80] to obtain the relationship between system reliability R and the component reliabilities R_1, R_2, \dots, R_n given by the function $R = f(R_1, R_2, \dots, R_n)$. The actual relationship between system reliability and components reliabilities depends on the specific software architecture. If we treat each component reliability on the right - hand side of this expression as a random variable, then the system reliability is

also a random variable. Note that the uncertainty analysis in 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 close 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). The method of moments allows us to obtain the estimates of the expected value $E[R]$ and k th central moments $\mu_k[R]$ for system reliability based on (1) knowledge of the system structure $R = f(R_1, R_2, \dots, R_n)$ and (2) data on the components reliabilities from which estimates of $E[R_i]$ and $\mu_k[R_i]$ for $i = 1, 2, \dots, n$ can be obtained.

System reliability moments are generated by expanding the system function $R = f(R_1, R_2, \dots, R_n)$ in a multivariable Taylor series expansion about the statistically expected values of each of the component reliabilities $E[R_i]$. We have used *Mathematica* to derive the system reliability expression $R = f(R_1, R_2, \dots, R_n)$ and its partial derivatives for the Taylor series expansion.

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

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

where

$$a_0 = f(E[R_1], E[R_2], \dots, E[R_n])$$

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

Then, the mean and the variance of system reliability are given by $E[R] \approx a_0$ and

$$Var[R] = \mu_2[R] \approx \sum_{i=1}^n a_i^2 Var[R_i].$$

The accuracy of the $E[R]$ and $Var[R]$ can be improved by including higher order terms in the Taylor series expansion. We have also derived the second order Taylor series expansion and the expressions for the mean and the variance of system reliability.

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

where

$$a_0 = f(E[R_1], E[R_2], \dots, E[R_n])$$

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

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

$$a_{ij} = \frac{\partial^2 R}{\partial R_i \partial R_j} \Big|_{R_i = E[R_i] \text{ for } i=1,2,\dots,n}$$

Then, the mean and the variance of system reliability for the second order Taylor approximation are given by

$$E[R] \approx a_0 + \frac{1}{2} \sum_{i=1}^n a_i \text{Var}[R_i]$$

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

Note that generating the mean and the variance of system reliability from the second order Taylor series expansion requires the knowledge of the first four central moments of component reliabilities. Even more, we can generate the first four central moments of the system reliability using the first eight central moments of component reliabilities. Then, the estimates of the first four moments may be used to select an empirical distribution from which the percentiles of the system reliability distribution may be obtained.

6.1. Application of the method of moments on the ESA case study

Next, we illustrate the method of moments on the European Space Agency case study. Table 8 compares the values obtained for the mean, variance and coefficient of variation $C_R = \sqrt{\text{Var}[R]} / E[R]$ (a relative measure of the spread of the distribution) of the system reliability for versions A and B using first and second order Taylor series expansion. As expected, version B has higher mean reliability than version A. In addition, the variance is smaller and the distribution of the system reliability is less spread. Further, for this example the second order approximation does not improve the accuracy.

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

Table 8. Mean and variance of the system reliability for versions A and B

In general, higher order Taylor series expansion will increase accuracy, as it can be seen from Table 9 which presents the results obtained for the hypothetical example given in Figure 6 (referred here as version C).

		First order Taylor series	Second order Taylor series
Version C	Mean	0.6261	0.6314
	Variance	0.0106	0.0101
	C_R	0.1640	0.1589

Table 9. Mean and variance of the system reliability for the hypothetical example

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

The method of moments has several advantages. First, it requires only the knowledge of the moments of components reliabilities, that is, no distribution function must be specified. Second, generation of random numbers is not required, therefore there is no sampling error. Finally, it could be applied to dependent as well as independent parameters, although the expressions for dependent variables would be more difficult to derive due to their complexity.

However, the method is approximate and a finite error is associated with the use of only up to first (second) order terms in the Taylor series expansion. Further, the accuracy of this method is not readily quantifiable. Therefore, if precise accuracy calculations for system reliability are required to support the uncertainty analysis, the method of moments might not be a good choice.

7. Uncertainty analysis based on Monte Carlo simulation

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

1. Derive the expression for system reliability.
2. Assign probability distributions to transition probabilities and components reliabilities.
3. Estimate the parameters of these probability distributions from 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.

In this report we derive the reliability expression (step 1) using the architecture - based software reliability model described in Section 3.3. Note that the uncertainty analysis in general and the use of the Monte Carlo method in particular are not limited to this model. They can be applied to any architecture - based model that provides close form solution.

In step 2 we assign probability distribution functions to transition probabilities and component reliabilities. These distribution functions can be based on theoretical assumption or on observed data.

We assume that component reliabilities are random variables with Beta distribution with pdf given by

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

where $0 \leq R_i \leq 1$.

We further assume that the rows in the transition probability matrix are independent and distributed accordingly to Dirichlet distribution. 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 [Martin67]. Thus, the joint density for the i th row in transition probability matrix has the form

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

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

For the simulation of Dirichlet distribution we use the transformation approach [Johnson87] based on the following property [Johnson69]. The standard Dirichlet distribution is defined as the distribution of (Y_1, Y_2, \dots, Y_n) where $Y_k = Z_k / \sum_{j=1}^n Z_j$ and $Z_j, j = 1, 2, \dots, n$ are independent, standard Gamma distributed random variables with shape parameter α_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, then the posterior distribution is also Dirichlet. Even in cases where the use of the Dirichlet distribution is not implied by theory, due to its variety of 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 ± 0.05) and they are uniformly distributed in the interval.

The basic characteristics of uncertainty analysis based on Monte Carlo simulation with respect to different criteria are the following:

- High data requirements in form of probability distribution functions of modeling parameters.
- Many characteristic of system reliability can be derived, including moments, percentiles, and distribution functions.
- The accuracy of the method may be increased simply by increasing the number of simulations. Although with the hardware available today it is not critical, it is worth mentioning that the computational cost increases with the sample size.

- Sampling errors may be involved in case of long tail distribution.
- Monte Carlo method scales very well, that is, it is not very sensitive to the number of components in the system.

7.1. Application of the Monte Carlo method on the ESA case study

Numerical results presented in this section were obtained using two commercial tools. First, we use *Mathematica* to derive the system reliability expressions $R = f(R_i, p_{ij})$ 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.

In Figure 23 we consider how the uncertainty of the operational profile A (i.e., transition probabilities) affects system reliability. For this purpose we keep the values of component reliabilities fixed to the point estimates given in Table 1.

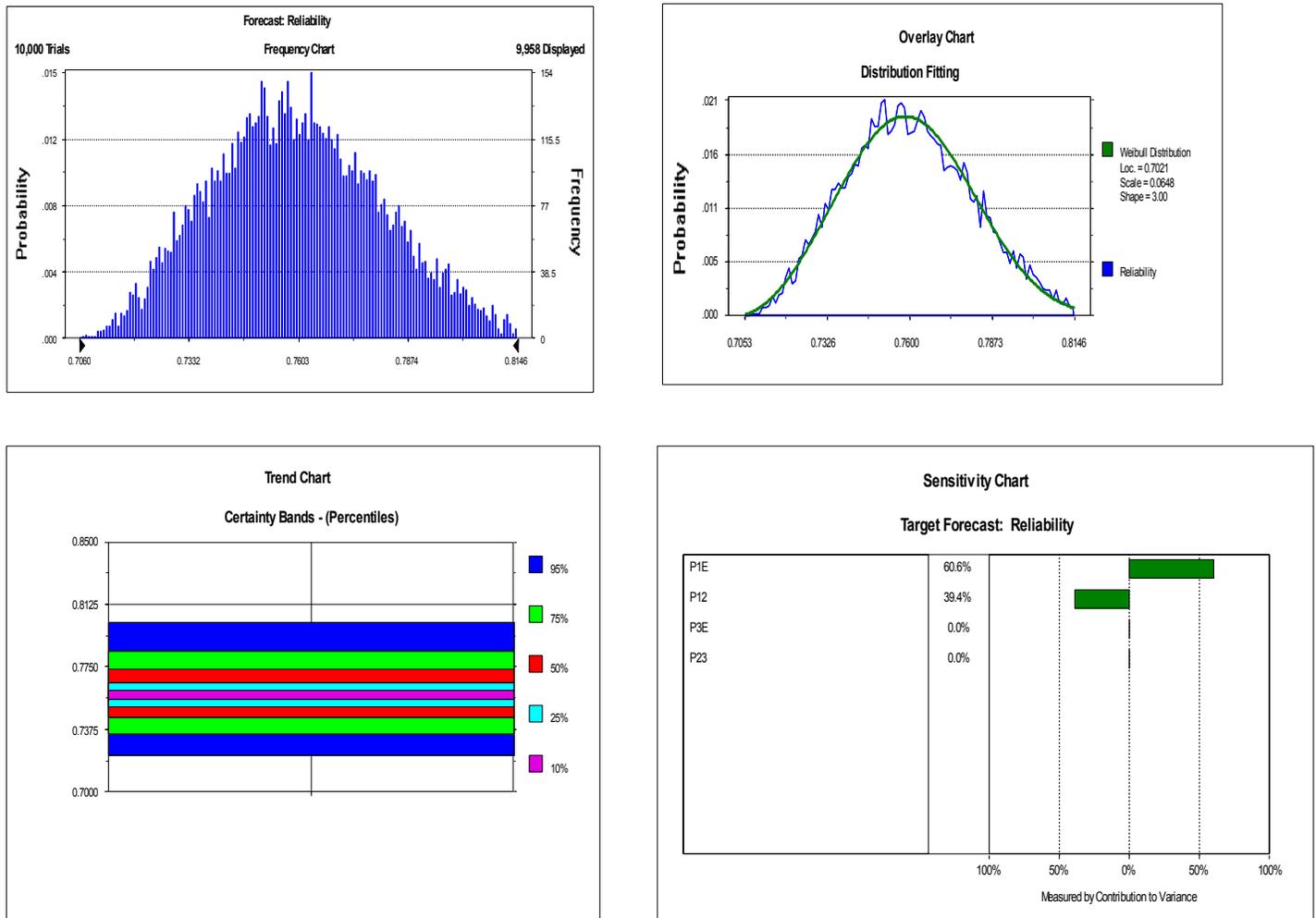


Figure 23. Uncertainty analysis of operational profile A

The mean of the system reliability obtained from the simulations 0.7600 is very close to the point estimate 0.7601 given in Table 2. The estimation of the mean reliability converges in approximately 3000 iterations (see Figure 24).

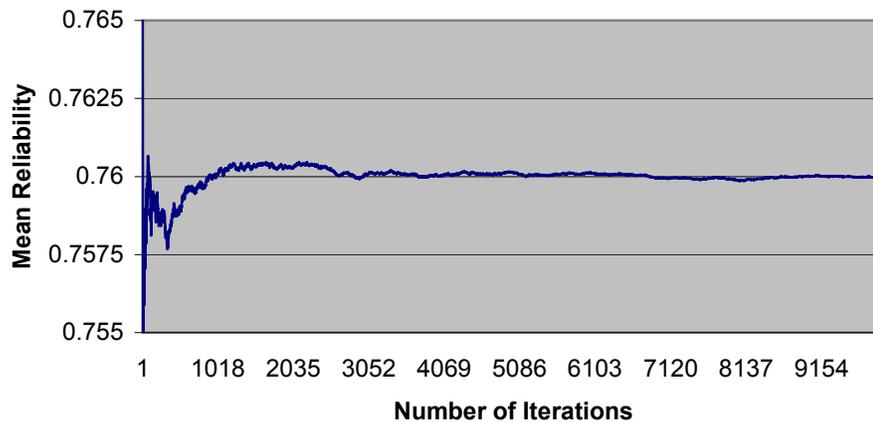


Figure 24. Convergence of the mean for the operational profile A

In addition to the mean reliability, we have estimated several other characteristics of the system reliability distribution [Hahn94]:

- Coefficient of variability which related to the spread of a distribution (0.0276).
- Skewness which relates to the lean of a distribution (0.2072).
- Kurtosis which related to the peakedness of a distribution (2.6047).

Note that these measures are relative which allows us to compare different distributions.

The frequency chart presented in Figure 23 gives the probability (frequency) of occurrence for different values of system reliability. In the case of operational profile A the range of the system reliability is 0.7048 to 0.8270. The estimated value of the variance 0.0004 is small compared to the variance of the transition probabilities. Also, the distribution is slightly skewed to the right (i.e. has a right tail). We have also done a distribution fitting for system reliability using the frequency data. In this case Weibull distribution with parameters given in the Figure 23 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. In case of operational profile A 95% band ranges from 0.7205 to 0.7975, which implies that 95% of the values obtained for the reliability fall in this range. Another interesting observation is with respect to the sensitivity of system reliability to different parameters. The parameters in the Figure 23 are ordered accordingly to their contribution to the variance of system reliability. Thus, the system reliability is the most sensitive to $p_{1E} = 1 - p_{12}$ and the variance is positive.

Next, we illustrate how the variation of transition probabilities and component reliabilities together affect the system reliability. The frequency chart, certainty bands, and sensitivity chart for version A are given in Figure 25. The range of the reliability [0.3759,0.9818] is significantly larger than the one

in Figure 23. The distribution is skewed to the left, that is, has the left tail. In this example, even though the variation of component reliabilities is small, they play critical role in the variation of system reliability. As it can be seen from the sensitivity chart 93.3% of the reliability variation is due to reliabilities R_1 and R_2 .

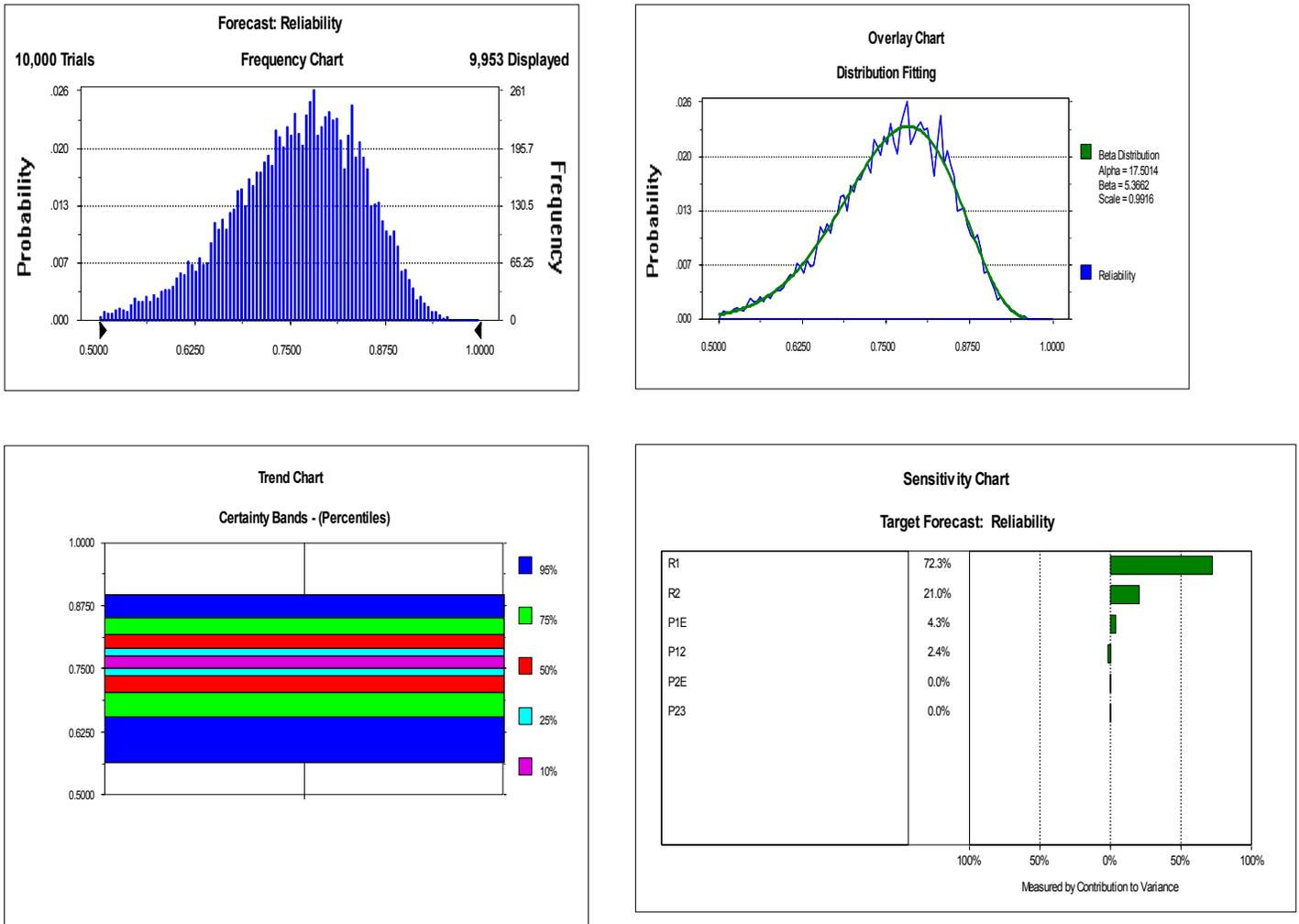


Figure 25. Uncertainty analysis of version A

In Figure 26 we present the results for version B obtained by varying transitions probabilities and component reliabilities. The reliability range in this case is [0.4571, 0.9952] and the reliability distribution is also skewed to the left. As it can be seen from the values given in Table 10 the reliability distribution of version B has higher mean and less variance. Further, it is more skewed to the left (that is, concentrated to the right), with higher peak. Also, certainty bands for version B are narrower than for version A. The system reliability is still more sensitive to the variation of the component reliabilities, although with smaller contribution to the variance (86.2%).

As the software development progresses, we expect that the software reliability will increase (for example because of fixing faults). Also, more accurate data will become available which will decrease the uncertainty of the parameters estimations. As a result, later in the life cycle we should obtain

software reliability distributions with higher mean (close to 1), less variance, concentrated to the right (i.e., skewed to the left), and with higher peaks.

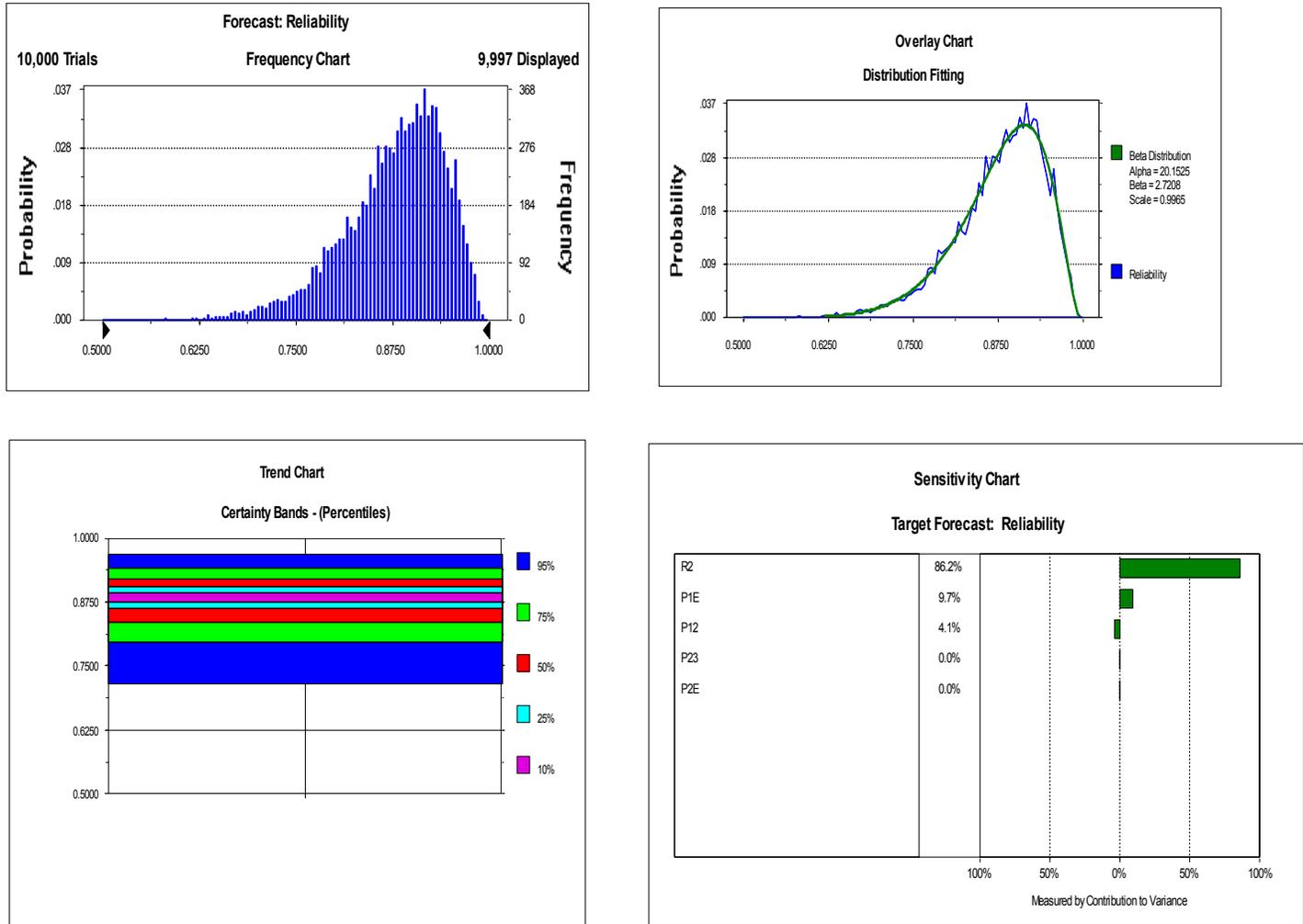


Figure 26. Uncertainty analysis of version B

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

Table 10. Characteristics of reliability distribution for versions A and B

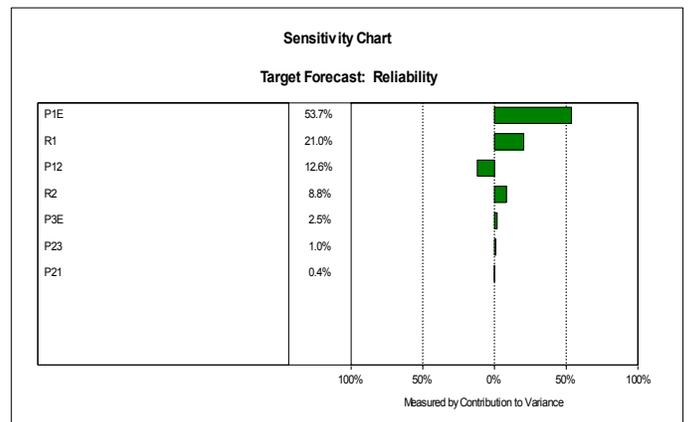
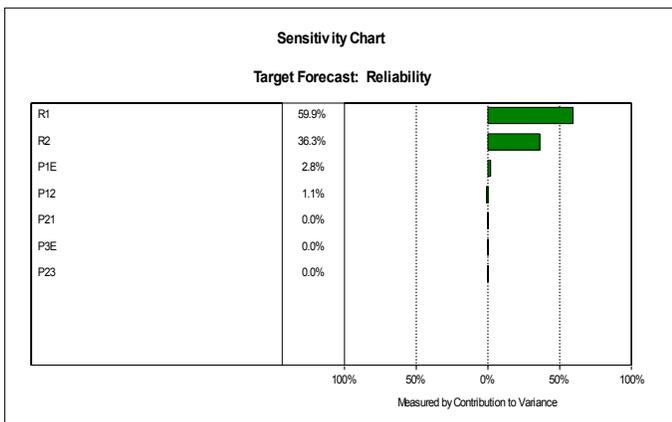
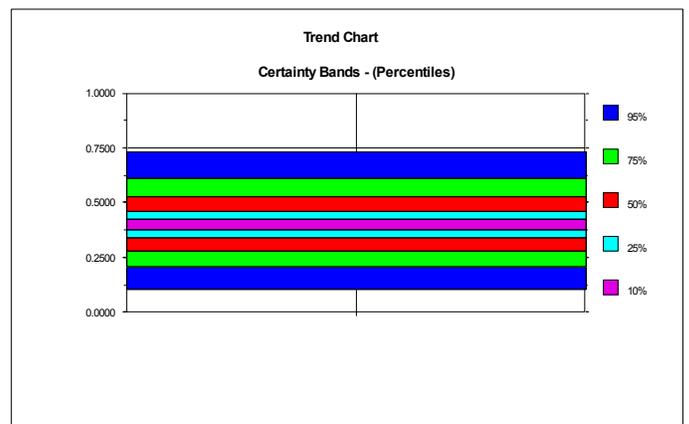
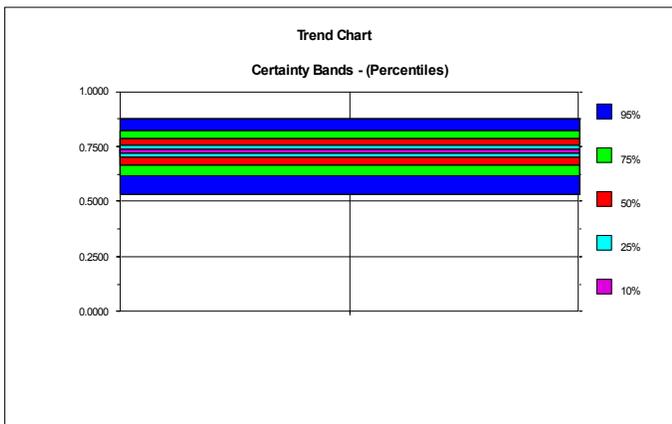
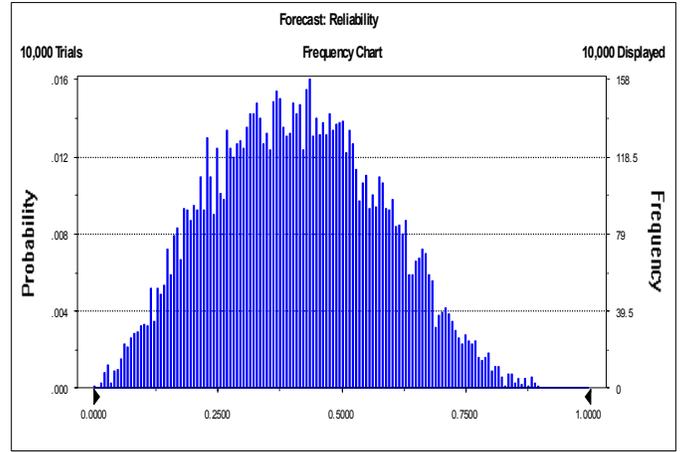
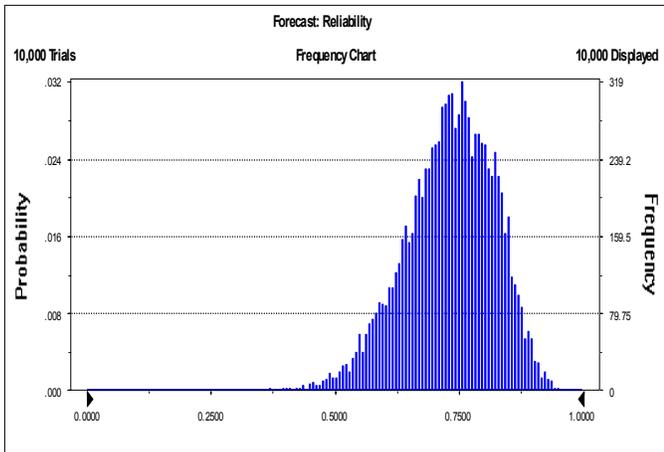
Our next numerical example illustrates the uncertainty analysis for the hypothetical example presented in Figure 6. Table 11 compares the characteristics of the system reliability distribution (mean, coefficient of variation, skewness, and kurtosis) for five different values of transition

probability p_{21} associated with the arc forming a loop in the model. In view of Table 11 the following observations are made. The mean system reliability decreases for higher values of transition probability p_{21} and is very close to the point estimate. In addition, we see that for higher values of p_{21} the coefficient of variation is increasing, distribution skewness is moving to the right, and the peak is decreasing. Due to the space limitations, we show the frequency charts, certainty bands, and sensitivity charts only for two values of p_{21} at the end of the spectrum ($p_{21} = 0$ and $p_{21} = 0.95$).

p_{21}	Mean	Coefficient of variability	Skewness	Kurtosis
0	0.7318	0.1225	-0.4458	3.0745
0.25	0.6865	0.1482	-0.3673	3.0230
0.5	0.6246	0.1886	-0.2524	2.7510
0.75	0.5363	0.2736	-0.1850	2.6411
0.95	0.4109	0.4112	0.1334	2.4294

Table 11. Characteristics of reliability distribution for the hypothetical example

It is obvious from Figure 27 that the characteristics of the system reliability distribution are very sensitive to the values of modeling parameters. We already knew from the point estimates [Goseva01b] that the system reliability for $p_{21} = 0.95$ is significantly lower than for $p_{21} = 0$. In addition, from uncertainty analysis we observe that the reliability distribution for $p_{21} = 0.95$ is widely spread and has wider certainty bands compared to $p_{21} = 0$. Also, the parameters contribution to the variance of system reliability changes significantly. Thus, in the case of $p_{21} = 0$ reliabilities R_1 and R_2 contribute 96.2% to the variance of system reliability, while in the case of $p_{21} = 0.95$ they contribute only 29.8%. Even more, when $p_{21} = 0.95$ the highest effect on the system reliability is coming from transition probability $p_{1E} = 1 - p_{12}$ which contributes 53.7% to the variance of system reliability. These results clearly illustrate the usefulness of uncertainty analysis and motivate its systematic use for software reliability prediction.



$$p_{21} = 0$$

$$p_{21} = 0.95$$

Table 27. Uncertainty analysis for the hypothetical example

8. Conclusion

In this report we have presented a methodology for uncertainty analysis of software reliability that can be applied throughout the software life cycle. Within this methodology, we have used the entropy, method of moments, and Monte Carlo simulation to analyze how the uncertainty of the parameters (transition probabilities and component reliabilities) propagates into the estimation of system reliability. We have applied the proposed methodology and different methods for uncertainty analysis on the case studies from the European Space Agency and NASA.

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

The main focus of our future work is to explore other methods for uncertainty analysis and compare them accordingly to different criteria. This comparison will 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.

References

- [Adams96] T. Adams, "Total Variance Approach to Software Reliability Estimation", *IEEE Trans. Software Engineering*, Vol. 22, No.9, 1996, pp.687-688.
- [Ash 65] R. Ash, *Information Theory*, John Wiley & Sons, 1965.
- [UML] G. Booch, J. Runbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, 1998.
- [Chen94] M. Chen, A. P. Mathur, and V. J. Rego, "A Case Study to Investigate Sensitivity of Reliability Estimates to Errors in Operational Profile", *Proc. 5th International Symposium on Software Reliability Engineering*, 1994, pp.276-281.
- [Cheung80] R. C. Cheung, "A User-Oriented Software Reliability Model", *IEEE Trans. Software Engineering*, Vol.6, No.2, 1980, pp.118-125.
- [Farr96] W. Farr, "Software Reliability Modeling Survey", in *Handbook of Software Reliability Engineering*, M. R. Lyu (Ed.), McGraw-Hill, 1996, pp.71-117.
- [Goseva01a] K. Goseva - Popstojanova and K. S. Trivedi, "Architecture-Based Approach to Reliability Assessment of Software System", *Performance Evaluation*, Vol.45, No.2-3, 2001, pp.179-204.
- [Goseva01b] K. Goseva - Popstojanova, A. P. Mathur, and K. S. Trivedi, "Comparison of Architecture-Based Software Reliability Models", *12th International Symposium on Software Reliability Engineering*, 2001, pp.22-31.
- [Goseva02a] K. Goseva-Popstojanova and S. Kamavaram, "Architecture-Based Methodology for Studying Sensitivity of Software Reliability to Operational Profile Errors", *Technical Report*, March 2002.

- [Goseva02b] K.Goseva-Popstojanova and S. Kamavaram, "Uncertainty Analysis of Software Reliability Based on Method of Moments", Fast abstract, *13th International Symposium on Software Reliability Engineering*, Nov 2002, to appear.
- [Goseva02c] K.Goseva-Popstojanova and S. Kamavaram, "Uncertainty Analysis of Architecture – Based Software Reliability", to be submitted for publication.
- [Hahn94] G. J. Hahn and S. S. Shapiro, *Statistical Models in Engineering*, John Wiley & Sons, 1994.
- [Jackson81] P. S. Jackson, R. W. Hockenbury, and M. L. Yeater, "Uncertainty Analysis of System Reliability and Availability Assessment", *Nuclear Engineering and Design*, Vol.68, 1981, pp.5-29.
- [Johnson87] M. E. Johnson, *Multivariate Statistical Simulation*, John Wiley & Sons, 1987.
- [Johnson69] N. L. Johnson, S. Kotz, *Distributions in Statistics: Continuous Multivariate Distributions*, John Wiley & Sons, 1969.
- [Kamavaram02] S. Kamavaram and K. Goseva - Popstojanova, "Entropy as a Measure of Uncertainty in Software Reliability", Student paper, *13th International Symposium on Software Reliability Engineering*, Nov 2002, to appear.
- [Leung97] Y-W Leung, "Software Reliability Allocation under an Uncertain Operational Profile", *Journal of the Operational Research Society*, Vol. 48, 1997, pp.401-411.
- [Martin67] J. J. Martin, *Bayesian Decision Problems and Markov Chains*, John Wiley & Sons, 1967.
- [Miller92] K. W. Miller, L. J. Morell, R. E. Noonan, S. K. Park, D. M. Nikol, B. W. Murrill, and J. M. Voas, "Estimating the Probability of Failure when Testing Reveals no Failures", *IEEE Trans. Software Engineering*, Vol.18, No.1, 1992, pp.33-43.
- [Musa93] J. D. Musa, "Operational Profiles in Software Reliability Engineering", *IEEE Software*, Vol.10, 1993, pp.14-32.
- [Musa94] J. D. Musa, "Sensitivity of Field Failure Intensity to Operational Profile Errors", *5th International Symposium on Software Reliability Engineering*, 1994, pp.334-337.
- [Nelson73] E. Nelson, "A Statistical Bases for Software Reliability", *TRW-SS-73-02, TRW Software series*, 1973.
- [Pasquini96] A. Pasquini, A. N. Crespo, and P. Matrella, "Sensitivity of Reliability - Growth Models to Operational Profile Errors vs. Testing Accuracy", *IEEE Trans. Reliability*, Vol.45, No.4, 1996, pp.531-540.
- [Siegrist88] K. Siegrist, "Reliability of System with Markov Transfer of Control", *IEEE Trans. Reliability*, Vol.14 No.7, 1988, pp.1409-1053.
- [Singh01] H. Singh, V. Cortellessa, B. Cukic, E. Guntel, and V. Bharadwaj, "A Bayesian Approach to Reliability Prediction and Assessment of Component Based Systems", *12th International Symposium on Software Reliability Engineering*, 2001, pp.12-21.
- [Wesslen00] A. Wesslen, P. Runeson, and B. Regnell, "Assessing the Sensitivity to Usage Profile Changes in Test Planning", *11th International Symposium on Software Reliability Engineering*, 2000, pp.317-326.
- [Whittaker93] J. A. Whittaker and J. H. Poore, "Markov Analysis of Software Specifications", *ACM Trans. Software Engineering and Methodology*, Vol.2, No.1, 1993, pp.93-106.

[Yin01] L. Yin, M. A. J. Smith, and K. S. Trivedi, "Uncertainty Analysis in Reliability Modeling", *2001 Annual Reliability and Maintainability Symposium*, 2001, pp.229-234.

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

[ATAC] <http://xsuds.argreenhouse.com>