Sensitivity of Software Usage to Changes in the Operational Profile

Sunil Kamavaram and Katerina Goševa-Popstojanova
Lane Department of Computer Science and Electrical Engineering
West Virginia University, Morgantown, WV 26506-6109
{sunil, katerina}@csee.wvu.edu

Abstract

In this paper we present a methodology for uncertainty analysis of the software operational profile suitable for large complex component–based applications and applicable throughout the software life cycle. Within this methodology we develop a method for studying the sensitivity of software usage to changes in the operational profile based on perturbation theory. This method is then illustrated on three case studies: software developed for the European Space Agency, an e–commerce application, and real–time control software. Results show that components with small execution rates are the most sensitive to the changes in the operational profile. This observation is very important due to the fact that rarely executed components usually handle critical functionalities such as exception handling or recovery.

1 Introduction

Verification and validation of software products, as well as the predictive quality of software reliability estimates are affected by the ability to estimate the correct operational profile. However, 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 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 different users. In addition, 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. For these reasons, uncertainty analysis of the operational profile and software reliability estimates are of essential importance.

Several papers that use discrete Markov chains to describe software usage are relevant to our work. In [26] Markov analysis of software specifications was presented and entropy was used as a measure of uncertainty. In [25] the impact of uncertainties in the operational profile on the usage coverage was analyzed.

The most common method for uncertainty analysis in software reliability is conducting sensitivity studies. In [3], [20], [21] sensitivity of software reliability estimates to errors in the operational profile has been investigated in the context of black–box reliability growth models. In [4], [23] architecture–based models have been used to study the sensitivity of software reliability estimates. In these studies the operational profile was assumed to be fixed (i.e., fixed known values for transition probabilities) and the sensitivity of software reliability was studied with respect to the reliability of each component. Sensitivity studies of software reliability with respect to the transition probabilities (i.e., operational profile) and component reliabilities were presented in [7], [9], [27].

In addition to sensitivity studies, there have been several attempts to quantify the variability of software reliability. In [19] authors used black–box approach and 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 recognized that there is uncertainty in the estimations of the reliability for each partition, as well as uncertainty in the probability of using each partition. In [24] the mean and the variance of software failure probability for the architecture–based model were estimated using Bayesian approach and assuming Beta prior distributions for component failure probabilities. In another related work [16] three optimization models for software reliability allocation under an uncertain operational profile were formulated and solved.

From the above it is obvious that uncertainty analysis

was not used systematically and extensively in the area of software reliability. Recently, we have developed a methodology for uncertainty analysis of the operational profile and software reliability [10]. Our methodology is suitable for large complex component–based applications and applicable through the software life cycle. Within this methodology we have considered several methods for uncertainty analysis. In [15] we have used entropy for uncertainty analysis. Entropy is a well know concept from information theory that allows us to quantify the uncertainty of the operational profile, uncertainty of the overall system reliability, and component uncertainties. Although the obtained results are useful for verification and validation of component–based systems, the method based on entropy does not provide an estimate of software reliability. In [10] we have used method of moments and Monte Carlo simulation to quantify the uncertainty of software reliability estimates. We have argued that 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.

In this paper, we present a new method for uncertainty analysis of the software operational profile based on perturbation theory. Perturbation theory, as well as entropy, does not provide estimates of the software reliability measures. Thus, we can say the perturbation theory and entropy as methods for studying uncertainty are complementary with the method of moments and Monte Carlo simulation.

The rest of the paper is organized as follows. In section 2 we briefly describe our methodology for uncertainty analysis. Section 3 presents an overview of the perturbation theory and its use for sensitivity analysis of software usage. In section 4 the perturbation theory as a method for uncertainty analysis is applied and validated on three case studies: the software developed for the European Space Agency, an e–commerce application, and a real–time control software. Finally, the concluding remarks are given in section 5.

2 Methodology for uncertainty analysis of the operational profile

Our methodology for uncertainty analysis of the software operational profile is presented in Figure 1. We use the architecture–based approach to describe the software operational profile. Software architecture shows the behavior of the software with respect to the manner in which different components interact. Dynamic information in software architecture represented by the probabilities of control transfer clearly depends on the software usage, that is, the operational profile. We use state–based approach to build the model of the software operational profile [8], [9]; the states represent active components and arcs represent the transfer of control. Here, we assume that the transfer of control between the components has a Markov property. Therefore, the operational profile is modelled with a discrete time Markov chain (DTMC) with a transition probability matrix $P = [p_{ij}]$, where $p_{ij} = \Pr\{$*program transfers the control from component i to component j*$\}$.

We construct the Markov chain in two phases. 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 **dynamical statistical phase** involves the estimation of the relative frequencies of component interactions, that is, transition probabilities which are clearly dependent on the operational profile. During the early phases of the 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 the software operational profile.

- **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 and usage 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 [2]. Use case diagrams provide graphical description of how external objects interact with the system, while sequence diagrams show components and messages that are exchanged between them.

- **Informed Approach** is used during the late phases of the software development when testing or field data become available. Thus, component traces obtained using profilers [28] and test coverage tools [29] can be used to obtain a set of execution paths and establish the frequency count of the transition arcs.

The discrete time Markov chain proves to be a good model for software architectures and operational profiles for several reasons. From the software engineering point of view, the model can be build both in early and late phases of the software life cycle. Once the model has been built, any number of statistically typical test cases can be generated from the model [26]. From the analytical point of view, this is a tractable stochastic process with well developed theory, analytical results, and computational algorithms. Furthermore, the model provides basis for building the several different architecture-based software reliability models [8].

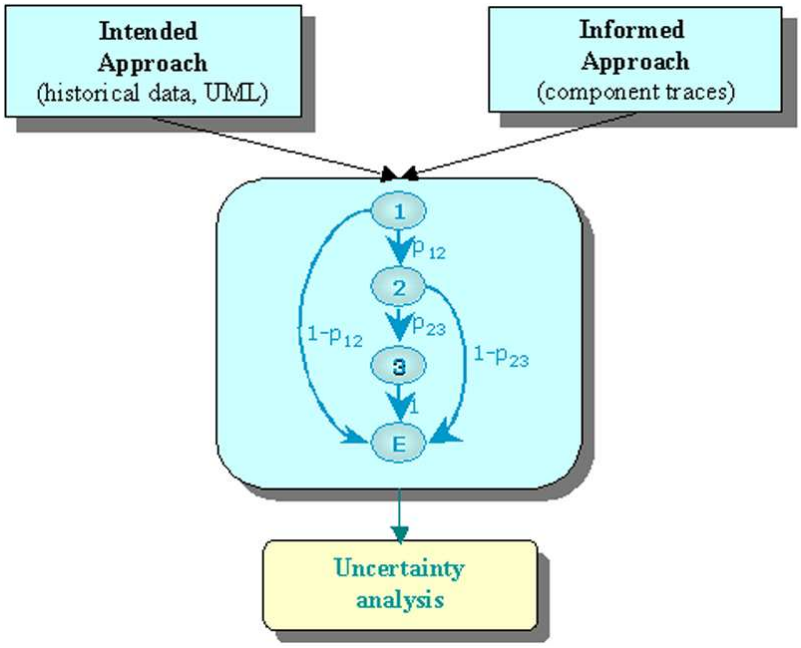

Figure 1. Methodology for uncertainty analysis of the operational profile

3 Overview of perturbation theory

Perturbation theory provides mathematical means to study how the stationary distribution of an irreducible discrete time Markov chain changes due to the variation of transition probabilities. As described in section 2 we describe the operational profile of a software as a discrete time Markov chain with transition probability matrix P. The stationary distribution vector of P is a positive vector $\pi^T = (\pi_1, \pi_2, \ldots, \pi_n)$ that is obtained by solving the equation

$$\pi^T P = \pi^T, \quad \sum_{j=1}^{n} \pi_j = 1. \tag{1}$$

Since π_i can be interpreted as the expected execution rate of a 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 by a matrix E. Then, the resultant matrix $\bar{P} = P - E$ is also a transition probability matrix of an irreducible discrete time Markov chain. Sensitivity results concerning the absolute perturbations have been phrased in terms of the perturb matrix $E = P - \bar{P}$ as

$$\|\pi^T - \bar{\pi}^T\| \leq k\|E\|_\infty \tag{2}$$

where $\|E\|_\infty$ is the norm of the perturb matrix and k is a condition number used as measure of sensitivity. In [5], [6], [14] 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 $Z \equiv (A + e\pi^T)^{-1}$ [18] of the underlying Markov chain or the group inverse of $A \equiv I - P$ [22]. 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 shortcomings and are not satisfying because of the following two reasons. First, irreducible chains exist for which the bounds are not tight, that is, 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 the 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 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. Therefore, in this paper we use the condition number k_8 expressed in terms of mean passage time to estimate the bounds on stationary probabilities.

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 π^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. A standard procedure for computing the mean first passage times of a finite irreducible discrete time Markov chains was developed in [11], [12], [13]. The matrix M (i.e., mean first passage time matrix) is computed using Meyer's group inverse matrix [18] given by

$$A^{\#} \equiv Z - \Pi \tag{3}$$

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

The diagonal elements M_{jj} of the matrix M specify the mean return time of the states and non-diagonal elements M_{ij} specify the mean first passage time from state i to state j. The matrix M is estimated using the equation [12]

$$M = [M_{ij}] = [I - A^{\#} + LA_d^{\#}]D \tag{4}$$

where I is the identity matrix, $A^{\#}$ is Meyer's group inverse matrix given by equation (3), $L = ee' = [1]$ (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 $D = (\Pi_d)^{-1}$.

Using the condition number expressed in terms of the mean first passage times, the absolute change in the j-th stationary probability [13] is estimated as

$$|\pi_j - \bar{\pi}_j| \leq \frac{\max_{i \neq j} M_{ij} - 1}{2 \cdot M_{jj}} ||E||_\infty \tag{5}$$

which leads the to relative change in π_j given by

$$\frac{|\pi_j - \bar{\pi}_j|}{\pi_j} \leq \frac{\max_{i \neq j} M_{ij} - 1}{2} ||E||_\infty. \tag{6}$$

4 Application of perturbation analysis on case studies

In this section we illustrate the use of perturbation theory as a method for uncertainty analysis of the operational profile on three different case studies.

4.1 European Space Agency software

The application from the European Space Agency (ESA) [9] provides language–oriented user interface, which allows the user to explain the configuration of an array of antennas. The application purpose is to prepare a data file in accordance with a predefined format and characteristics from the 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 has been extensively used after the last fault removal without failures. This gold version was used as an oracle in the experiment. A set of test cases was generated randomly accordingly to the known operational profile determined by interviewing the users of the program. Component traces obtained during the testing were used for building the software operational profile and estimating transition probabilities. It follows that for the ESA case study we use informed approach.

DTMC that represents the software operational profile is shown in Figure 2. Components 1, 2 and 3 correspond to the Parser, the Computational and the Formatting subsystems respectively; state E represents the completion of execution. 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. Transition probabilities were estimated using $p_{ij} = \frac{n_{ij}}{n_i}$, where n_{ij} is the number of times control was transferred form component i to component j, and $n_i = \sum_j n_{ij}$.

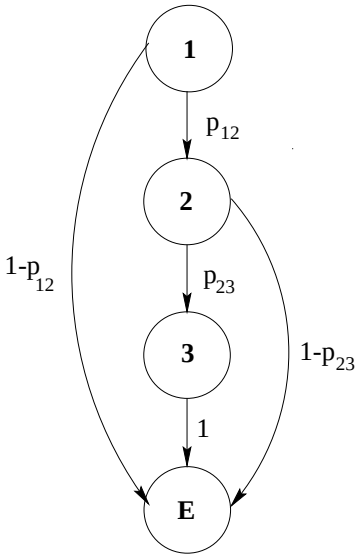

Figure 2. Operational profile of the ESA case study

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 perturb matrix E which leads to the transition probability matrix $P_B = P_A - E$ of the operational profile B.

$$P_A = \begin{bmatrix} 0 & 0.5933 & 0 & 0.4067 \\ 0 & 0 & 0.7704 & 0.2296 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

$$E = \begin{bmatrix} 0 & 0.1431 & 0 & -0.1431 \\ 0 & 0 & -0.1018 & 0.1018 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

Using the equations (5) and (6) we estimate the absolute and relative changes in components execution rates. As it can be seen from Table 1, 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 a hypothetical example based on the European Space Agency application which has a loop back from component 2 to 1 [9]. The operational profile of the hypothetical example is shown in Figure 3. Note that for the high values of the transition probability p_{21} the components 1 and 2 will be executed within a loop many times.

States	1	2	3	E
Execution rate	0.3278	0.1945	0.1498	0.3278
Absolute change	0.0962	0.1153	0.1217	0.0962
Relative change	0.2934	0.5926	0.8119	0.2934

Table 1. Perturbation analysis of the ESA case study

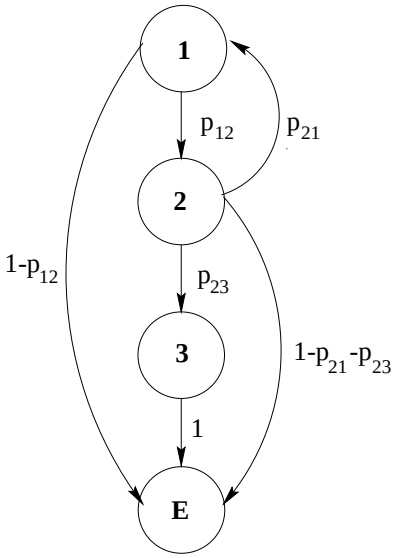

Figure 3. Operational profile of the hypothetical example

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 $P_D = P_C - E$.

$$P_C = \begin{bmatrix} 0 & 0.8 & 0 & 0.2 \\ 0.25 & 0 & 0.25 & 0.5 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad (9)$$

$$E = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & -0.5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (10)$$

From the results given in Table 2 it can be seen that the operational profile C is not stable in relative sense to the perturbations. In particular, the component 3 which has the smallest execution rate is the most sensitive in relative sense to the changes made to the transition probability matrix P_C.

4.2 E–commerce application

In this section we analyze the sensitivity of a software usage in a typical e–commerce application adopted from [17]. In the e–commerce applications 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 [17], the user's navigation pattern within a session is captured by so called Customer Behavior Model Graph (CBMG). The Customer Behavior Model Graph describes how the users navigate through the site, which functions they use, and the frequency of transitions from one function to another function. DTMC that describes the typical e–commerce application is shown in Figure 4.

States	1	2	3	E
Execution rate	0.3571	0.2857	0.0714	0.2857
Absolute change	0.3214	0.3571	0.4643	0.3571
Relative change	0.9	1.25	6.5	1.25

Table 2. Perturbation analysis of the hypothetical example

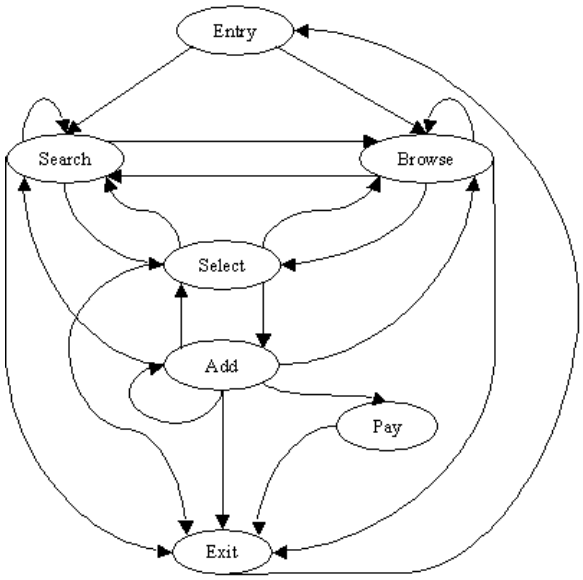

Figure 4. Operational profile of the e–commerce case study

As discussed in section 2, building the DTMC includes first construction of the structure of the application, and then assigning transition probabilities. For Web applications, there is usually a close resemblance between navigation patterns and the underling Web design and code because Web sites are designed to support directly such navigations. Consequently, the basic structure can be easily identified from product specifications, 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 operational profile from the HTML code or from the Web access logs.

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 [17] two operational profiles were 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 perturb matrix E that leads to operational profile typical for the heavy buyer.

$$P = \begin{bmatrix} 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0.4 & 0.35 & 0 & 0.2 & 0 & 0.05 \\ 0 & 0.35 & 0.4 & 0 & 0.2 & 0 & 0.05 \\ 0 & 0.2 & 0.2 & 0.05 & 0.2 & 0.3 & 0.05 \\ 0 & 0.425 & 0.425 & 0.05 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (11)$$

$$E = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.05 & -0.005 & 0 & 0.2 & 0 & 0.1 \\ 0 & -0.05 & -0.005 & 0 & 0.2 & 0 & 0.1 \\ 0 & 0.025 & 0.025 & 0.05 & -0.1 & 0 & 0 \\ 0 & -0.1 & -0.1 & 0.25 & 0 & 0 & -0.05 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (12)$$

As shown earlier, components with lower execution rates are more sensitive relatively to the changes in the operational profile. Thus, it can be observed from the results shown in Table 3 that the components 'Add' and 'Pay' which are visited rarely in the operational profile of an occasional buyer exhibit excessive relative change in the 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 than 'Select' and two orders of magnitude higher than 'Browse' and 'Search'. Further, the relative change of the expected execution rate of component 'Pay' is one order of magnitude higher than the relative change of the expected execution rate of 'Add' component.

4.3 Real–time control software

The real-time control software presented in this section is a real world application whose operational profile is shown in Figure 5. Due to confidentiality issues, details about this application are not provided. The operational profile for this case study was built using the intended approach. For this application we had available the UML use cases and sequence diagrams. We built the DTMC using UML sequence diagram that presents software components used for the given scenario and the messages that 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 scenario.

The sensitivity of the operational profile of the real–time control software given in Figure 5 is analyzed by perturbing the transition probability matrix P with perturb matrix E.

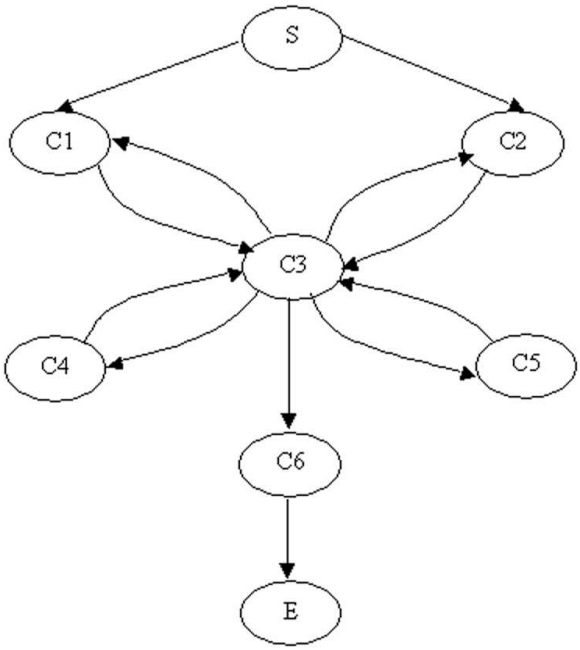

Figure 5. Operational profile of the real–time control software

The resulting matrix $\bar{P}$ represents a different usage of the real–time control application.

$$P = \begin{array}{c} \\ S \\ C1 \\ C2 \\ C3 \\ C4 \\ C5 \\ C6 \\ E \end{array} \begin{bmatrix} 0 & 0.5 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1/9 & 1/9 & 0 & 1/9 & 2/9 & 4/9 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

$$E = \begin{array}{c} \\ S \\ C1 \\ C2 \\ C3 \\ C4 \\ C5 \\ C6 \\ E \end{array} \begin{bmatrix} 0 & -0.3 & 0.3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/9 & 1/9 & 0 & 1/9 & 0 & -1/3 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

Table 4 shows the results of the perturbation analysis of the real–time control software obtained using equations (5) and (6). 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. Further, components C4 and C5 which are visited rarely are the most sensitive to changes in the relative sense.

States	Execution rate	Absolute change	Relative change
Entry	0.0542	0.2365	4.3615
Browse	0.3666	0.3149	0.8589
Search	0.3666	0.3149	0.8589
Add	0.0078	0.2652	34
Select	0.1428	0.2779	1.875
Pay	0.0023	0.2494	106.58
Exit	0.0542	0.2365	4.3615

Table 3. Perturbation analysis of the e–commerce case study

States	Execution rate	Absolute change	Relative change
S	0.1333	0.2889	2.1667
C1	0.1	0.3	3
C2	0.1	0.3	3
C3	0.3	0.3	1
C4	0.0333	0.3556	10.667
C5	0.0667	0.3778	5.6667
C6	0.1333	0.2889	2.1667
E	0.1333	0.2889	2.1667

Table 4. Perturbation analysis of the real–time control software

5 Conclusion

In this paper we have presented a method for uncertainty analysis based on perturbation theory that can be used to study how the change in the operational profile affects the expected execution rates of software components. We have applied this method on three different case studies: European Space Agency software, an e–commerce application, and a real–time control software. The 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 relative insensitive, then the operational profile is stable in both absolute and relative sense. Further, components with small execution rates (i.e., small stationary probabilities) are the most sensitive to changes in the operational profile. This observation is very important for software verification and validation due to the fact that rarely executed components usually handle critical functionalities such as for example exception handling or recovery.

Acknowledgements

This work is funded in part by grant from 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, Fairmont, West Virginia and by grant from the NASA West Virginia Space Grant Consortium, Research Initiation Grant Program.

References

[1] T. Adams, “Total Variance Approach to Software Reliability Estimation”, *IEEE Trans. Software Engineering*, Vol. 22, No.9, 1996, pp. 687-688.

[2] G. Booch, J. Runbaugh and I. Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, 1998.

[3] 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 Sysmposium on Software Reliability Engineering*, 1994, pp. 276-281.

[4] R. C. Cheung, “A User-Oriented Software Reliability Model”, *IEEE Trans. Software Engineering*, Vol. 16, No. 2, 1980, pp. 118-125.

[5] G. E. Cho and C. D. Meyer, “Markov Chain Sensitivity Measured by Mean First Passage Times”, *Linear Algebra Appl.*, 316, 2000, pp. 21-28.

[6] G. E. Cho and C. D. Meyer, “Comparison of Perturbation Bounds for the Stationary Distribution of a Markov Chain”, *Linear Algebra Appl.*, 335, 2001, pp.137-150.

[7] S. S. Gokhale and K. S. Trivedi, “Reliability Prediction and Sensitivity Analysis Based on Software Architecture”, *Proc. 13th International Symposium Software Reliability Engineering*, 2002, pp. 64-75.

[8] K. Goševa–Popstojanova and K. S. Trivedi, "Architecture-Based Approach to Reliability Assessment of Software System", *Performance Evaluation*, Vol. 45, N0. 2-3, 2001, pp. 179-204.

[9] K. Goševa–Popstojanova, A. P. Mathur, and K. S. Trivedi, "Comparison of Architecture-Based Software Reliability Models", *Proc. 12th International Symposium on Software Reliability Engineering (ISSRE 2001)*, 2001, Hong Kong, pp.22-31.

[10] K. Goševa–Popstojanova and S. Kamavaram, "Assessing Uncertainty in Reliability of Component-Based Software System", *Proc. 14th IEEE International Symposium on Software Reliability (ISSRE 2003)*, Denver, CO, Nov. 2003.

[11] J. J. Hunter, "A Survey of Generalized Inverses and their Use in Stochastic Modeling", *Res. Lett. Inf. Math. Sci.*, 1, 2000, pp. 25-36.

[12] J. J. Hunter, "Generalized Inverses, Stationary Distributions and Mean First Passage Times with Applications to Perturbed Markov Chains", *Res. Lett. Inf. Math. Sci.*, 3, 2002, pp. 99-116.

[13] J. J. Hunter, "Stationary Distributions and Mean First Passage Times of Perturbed Markov Chains", *Research Letters in Information and Mathematical Sciences, Institute of Information and Mathematical Sciences*, Massey University, Auckland, New Zealand, 3, 2002 pp. 85-98.

[14] I. C. F. Ipsen and C. D. Meyer, "Uniform Stability of Markov Chains", *SIAM J. Matrix Anal. Appl.*, Vol. 15, 1994, pp. 1061-1074.

[15] S. Kamavaram and K. Goševa-Popstojanova, "Entropy as a Measure of Uncertainty in Software Reliability", *13th Int'l Symp. Software Reliability Engineering*, 2002, Student paper, pp. 209-210.

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

[17] D. A. Menasce, "TCP-W A Banchmark for E-Commerce", *IEEE Internet Computing*, Vol.6, No.3, May/June 2002, pp.83-87.

[18] C. D. Meyer, "ensitivity of the Stationary Distribution of a Markov Chain", *SIAM J. Matrix Anal. Appl.*, Vol. 15, No. 3, July 1994, pp. 715-728.

[19] 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.

[20] J.D.Musa,"Sensitivity of Field Failure Intensity to Operational Profile Errors", *Proc. 5th International Sysmposium on Software Reliability Engineering*, 1994, pp.334-337.

[21] 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.

[22] Paul J. Schweitzer, "Perturbation Theory and Finite Markov Chains", *J. Appl. Prob.* 5, 1968, pp. 401-413.

[23] K. Siegrist, "Reliability of System with Markov Transfer of Control", *IEEE Trans. Reliability*, Vol. 14, No. 7, 1988, pp. 1409-1053.

[24] H. Singh, V. Cortellessa, B. Cukic, E. Guntel, and V. Bharadwaj, "A Bayesian Approach to Reliability Prediction and Assessment of Component Based Systems", *12th Int'l Symp. Software Reliability Engineering*, 2001, pp. 12-21.

[25] A. Wesslen, P. Runeson and B. Regnell, "Assessing the Sensitivity to Usage Profile Changes in Test Planning", *Proc. 11th International Symposium on Software Reliability Engineering*, 2000, pp. 317-326.

[26] 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.

[27] S. M. Yacoub, B. Cukic and H. H. Ammar, "Scenario-Based Reliability Analysis of Component-Based Software", *Proc. 10th Internaional Symposium on Software Reliability Engineering*, 1999, pp. 22-31.

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

[29] http://xsuds.argreenhouse.com