Entropy as a Measure of Uncertainty in Software Reliability

Sunil Kamavaram and Katerina Goseva-Popstojanova

Lane Department of Computer Science and Electrical Engineering
West Virginia University, Morgantown, WV 26506-6109

1. Introduction

The predictive quality of software reliability models is affected by the ability to estimate the correct operational profile. However, building an operational profile is not an easy task, especially for a new product. Therefore, uncertainty analysis of the operational profile and software reliability are of essential importance. In this paper, we present the uncertainty analysis of architecture-based software reliability models [2] using entropy, a well-known concept from information theory [1]. Source entropy that measures the amount of uncertainty inherent in a Markov source is given by [1]

$$H = -\sum_i \pi_i \sum_j p_{ij} \log p_{ij} \qquad (1)$$

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. The entropy value is maximum when all the transitions that are exit arcs from each state are equiprobable.

Source entropy was used in [3] to quantify the uncertainty present in a Markov model of software specifications. In this paper, we use the concept of source entropy to quantify the uncertainty of the operational profile and architecture-based software reliability models. In addition, conditional entropy is used to determine the uncertainty of components.

2. Uncertainty of operational profile

We illustrate our approach of uncertainty analysis on the European Space Agency application which consists of almost 10,000 lines of C code [2]. Discrete time Markov chain (DTMC) that represents software architecture is shown in Figure 1; components 1, 2 and 3 correspond to the Parse, Computational and Formatting subsystems respectively, while state E represents the completion of execution. Transition probability p_{ij} represents the probability that the control will transfer from component i to component j. In order to estimate the source entropy we consider multiple software executions, i.e., add a transition (with probability 1) from state E to state 1.

In the experiment presented in [2], two faulty versions (A and B) of the program were constructed. Estimated values of transition probabilities p_{ij} and component reliabilities R_i for both faulty versions are given in Table 1.

Using equation (1) we plot in Figure 2 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 $p_{12} = p_{23} = 0.5$. The uncertainty of the operational profiles A and B are 0.4707 and 0.4604, respectively. Thus, operational profile A is more uncertain than operational profile B, although the difference is not significant.

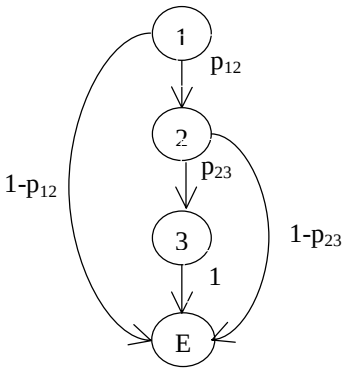


Figure 1. Software architecture for the case study

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. Estimated parameters for versions A and B

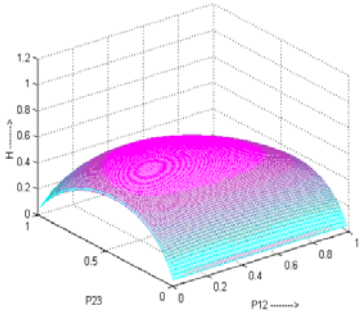


Figure 2. Uncertainty of the operational profile

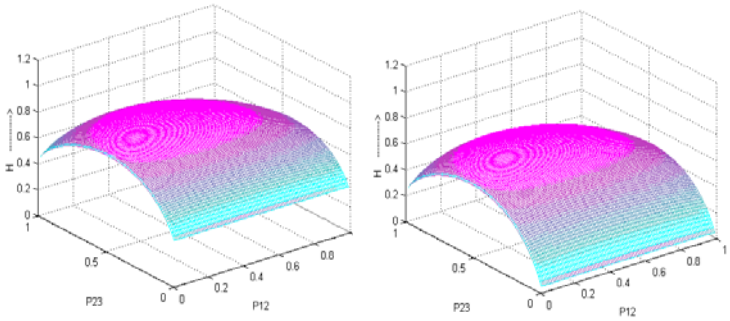
3. Uncertainty of software reliability

Next, we consider the uncertainty of software reliability. In the model, the failure state F is added and the failure of component i is considered by creating a directed arc to F with transition probability $(1-R_i)$. The

Student paper ISSRE Copyright 2002

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.

Again, we consider multiple software executions by adding transitions from both states E and F to the starting state 1. In Figure 3 we illustrate how the uncertainty H for versions A and B vary as functions of p_{12} and p_{23}. 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 2). Note that version B, which is more reliable, is less uncertain than version A.



Version A Version B

Figure 3. Uncertainty of software reliability

4. Uncertainty of components

In this section we focus on the uncertainty of components using the conditional entropy [1]

$$H_i = -\sum_j p_{ij} \log p_{ij} . \quad (2)$$

In general, uncertainty of component i will be higher if it transfers the control to more components and the transition probabilities equiprobable. As it can be seen from Table 2, component 1 in operational profile A has the highest uncertainty since transition probability p_{12} is close to 0.5. The uncertainty of component 3 is zero because it is certain that the control will be transferred to E. We also compute steady state probabilities $\pi = [\pi_i]$ by solving $\pi = \pi P$ (see Table 2). 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. Thus, components that have higher expected execution rate and higher uncertainty will require more testing effort.

Uncertainties and expected execution rates of components for software reliability model are shown in Table 3. Comparing the results in Table 2 and Table 3, we see that the uncertainty of component 1 in version B remains the same because $R_1 = 1$. For all other components (1 and 2 in version A and 2 in version B) the component uncertainty increases due to $R_i < 1$. In summary, components that have higher expected execution rate, higher component uncertainty and moderate reliability should be allocated more testing effort.

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

Table 2. Uncertainties and expected execution rates of components for operational profiles

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

Table 3. Uncertainties and expected execution rates of components for software reliability models

5. Conclusion

In this paper we have presented an approach of uncertainty analysis in software reliability. We use the concept of source entropy to quantify the uncertainty in operational profile and architecture-based software reliability models. It is shown that software systems that have uniform operational profile and moderate components reliabilities are more uncertain, and thus would require more testing effort. We further estimate the execution rate and uncertainty of each component using the theory of Markov chains and conditional entropy respectively. These measures can guide the process of identifying critical components and allocating testing time and resources.

6. Acknowledgements

This work is funded in part by grant from NASA OSMA managed through the NASA Independent Verification and Validation Facility, Fairmont, WV.

References

[1] R. Ash. *Information Theory*. Wiley, 1965.

[2] K. Goseva – Popstojanova, A. P. Mathur, and K. S. Trivedi. Comparison of Architecture – Based Software Reliability Models. *Proc. 12th Int'l Symp. Software Reliability Engineering*, 2001, pp. 22-31.

[3] J. A. Whittaker and J. H. Poore. Markov Analysis of Software Specifications. *ACM Trans. Software Engineering and Methodology*, 2 (1), 1993, pp. 93-106.