How Different Architecture Based Software Reliability Models Are Related? *

Katerina Goševa-Popstojanova † Kishor S. Trivedi
Dept. of Electrical and Computer Engineering
Duke University, Durham, NC 27708-0294
{katerina, kst}@ee.duke.edu

Aditya P. Mathur
Computer Sciences Department
Purdue University, West Lafayette, IN 47907
apm@cs.purdue.edu

1 Introduction

In the published papers on architecture based approach to software reliability modeling a large number of variants have been proposed, mostly by ad hoc methods. These have frequently tended to obscure the unifying structural properties common to many such variants. The mathematical treatment and the relation of these models becomes evident once their common structure is exhibited. In this paper we discuss some of the existing architecture based software reliability models and their interrelationships.

2 State-based models

State-based models estimate software reliability analytically. They assume that the transfer of control between modules has a Markov property, that is, model software architecture with a discrete time Markov chain (DTMC), continuous time Markov chain (CTMC), or semi Markov process (SMP). The reliability of software application is estimated either by solving the composite model that combines software architecture with failure behavior (composite method), or by superimposing failure behavior on the solution of the architectural model (hierarchical method).

First, consider the models of applications that operate on demand for which software runs that correspond to terminating execution can be clearly identified. These models assume that the control flow graph has a single entry and a single exit node. Note that this is not a fundamental requirement; the models can easily be extended to cover multientry, multiexit graphs.

Cheung model [1] describes the architecture of the software system made up of n components by an absorbing DTMC with a transition probability matrix $P = [p_{ij}]$, where p_{ij} is the probability of transfer of control from component i to component j. Failure behavior is defined by the component's reliability R_i. The solution method is composite; two absorbing states S and F are added, representing the correct output and failure respectively and the transition probability matrix P is modified appropriately. The original transition probability p_{ij} 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. The failure of component i is modeled by creating a directed edge to failure state F with transition probability $(1 - R_i)$. The reliability of a program is expressed in terms of the fundamental matrix of the composite model. Basically, it is equivalent to the sum of the reliabilities of all paths that start at the entry node and end at the exit node S, including the possibility of infinite number of paths due to the loops that might exist between two or more components.

Kubat model [2] characterizes software architecture by an SMP defined as follows. Transitions between components follow a DTMC with transition probability matrix P and the sojourn time in state i has pdf $g_i(t)$. Assuming constant failure intensity λ_i, the reliability of component i is estimated as $R_i = \int_0^\infty e^{-\lambda_i t} g_i(t)\, dt$. Once component reliabilities are estimated the solution approach reduces to the hierarchical treatment of the Cheung model [1]. The expected number of times component i is executed during a single execution of a software, denoted by V_i, is obtained from the DTMC and the system reliability becomes $R \approx \prod_{i=1}^{n} R_i^{V_i}$.

Gokhale et al. model [3] describes the architecture by an absorbing DTMC and uses a hierarchical solution method. However, it differs in the approach taken to estimate the component reliabilities. Given time-dependent failure intensity $\lambda_i(t)$ and the cumulative expected time $V_i t_i$ spent in the component per execution of the application, the reliability of component i is estimated as $R_i = e^{-\int_0^{V_i t_i} \lambda_i(t)\, dt}$. Thus, the reliability of the overall application becomes $R = \prod_{i=1}^{n} R_i$. Note that, the special case of this model that assumes constant failure intensities λ_i is equivalent to the special case of Kubat model [2] that assumes deterministic execution times $g_i(t) = t_i$.

Now, consider the models of continuously operating software applications that describe software architecture with an irreducible CTMC [4], [5]. For the sake of comparison we discuss *Laprie model* [5] since it considers only component failures. The parameters are the transition probabilities p_{ij}, the mean execution time $1/\gamma_i$ and the constant failure intensity λ_i of each component i. Solution method is hierarchical, based on the assumption that the failure rates are much smaller than the execution rates $\lambda_i \ll \gamma_i$. This

*We acknowledge the financial support of NSF under the grant EEC-9714965 (IUCRC TIE grant between Duke CACC and Purdue SERC).

†On leave of absence from the Department of Computer Science, Faculty of Electrical Engineering, Skopje, Macedonia

leads to the asymptotic behavior relative to the execution process. As a consequence, the system failure rate tends to $\lambda_S = \sum_{i=1}^{n} \pi_i \lambda_i$, where $\pi = [\pi_i]$ is the the stationary probability vector of the execution process governed by the generator matrix B' with diagonal entries equal to $-\gamma_i$ and off-diagonal entries to $p_{ij}\gamma_i$. The probability that there will be no failure up to time t, that is, the system reliability becomes $R(t) \approx e^{-\lambda_S\, t} = e^{-\sum_{i=1}^{n} \pi_i \lambda_i\, t}$.

To compare this model with the models of terminating applications we need to partition the realization of the irreducible CTMC with generator matrix B' to single software runs that are equivalent to realizations from a fixed starting state to recurrence of starting state. The execution time of a single run is a random variable with expectation equal to $\bar{t} = \sum_{i=1}^{n} V_i/\gamma_i$[1]. It can be shown that the expression for $R(\bar{t}\,)$ is equivalent to the expression obtained using the hierarchical treatment of Cheung model.

3 Path-based models

The approach taken to combine the software architecture with the failure behavior by this class of models can be described as path-based since the system reliability is computed considering the possible execution paths of the program. Thus, *Shooman model* [6] considers m different paths that software execution can take and assumes that the frequency f_i with which path i is run and its failure probability on each run q_i are known. The system failure probability on any run is given by $q_0 = \sum_{i=1}^{m} f_i q_i$. The following two papers demonstrate two different approaches for estimating path reliabilities.

Krishnamurthy and Mathur model [7] takes an expiremental approach to obtain path reliability estimates; a sequence of components along different paths are observed using the component traces collected during testing. The component trace of a program P for a given test case t, denoted by $M(P,t)$, is the sequence of components i executed when P is executed against t which leads to the path reliability $R_t = \prod_{\forall i \in M(P,t)} R_i$. The reliability estimate of a program with respect to a test set T is obtained by averaging path reliabilities $R = \sum_{\forall t \in T} R_t / \,|T|$.

Yacoub, Cukic and Ammar model [8] takes an algorithmic approach to estimate path reliabilities; a tree-traversal algorithm expands all branches of the graph that represents software architecture. The breadth expansions of the tree are translated as the summation of reliabilities weighted by the transition probability along each path. The depth of each path represents the sequential execution of components, and is hence translated to multiplication of reliabilities. The depth expansion of a path terminates at the exit node (a natural end of an application execution) or when the summation of execution time of that path sums to the average execution time of the application. The latter condition guaranties that the loops between two or more components don't lead to a deadlock.

The difference in reliability predictions of the state-based and path-based approaches becomes evident only when the control flow graph of the application contains loops. Thus, while state-based models analytically account for the potentially infinite number of paths, path-based models restrict the number of paths to ones observed experimentally during the testing [7] or terminate the depth traversal of each path using the average execution time of the application [8].

[1] Again, it is straightforward to generalize this result and compute the expected execution time of a single run for different starting states.

4 Concluding remarks

Many architecture based software reliability models have been proposed in the past, without an attempt to establish a relationship among them. The aim of this short paper is to fill the above gap. It builds on our earlier work [9] that contains an extensive survey on the underlying assumptions, usefulness and limitations of architecture based software reliability models.

References

[1] R.C.Cheung, "A User-Oriented Software Reliability Model", *IEEE Trans. Software Eng.*, Vol.6, No.2, 1980, pp. 118-125.

[2] P.Kubat, "Assessing Reliability of Modular Software", *Operations Research Letters*, Vol.8, 1989, pp. 35-41.

[3] S.Gokhale, W.E.Wong, K.Trivedi and J.R.Horgan, "An Analytical Approach to Architecture Based Software Reliability Prediction", *Proc. 3rd Int'l. Computer Performance & Dependability Symp.*, 1998, pp. 13-22.

[4] B.Littlewood, "A Reliability Model for Systems with Markov Structure", *Applied Statistics*, Vol.24, No.2, 1975, pp. 172-177.

[5] J.C.Laprie, "Dependability Evaluation of Software Systems in Operation", *IEEE Trans. Software Eng.*, Vol.10, No.6, 1984, pp. 701-714.

[6] M.Shooman, "Structural Models for Software Reliability Prediction", *Proc. 2nd Int'l. Conf. Software Eng.*, 1976, pp. 268-280.

[7] S.Krishnamurthy and A.P.Mathur, "On the Estimation of Reliability of a Software System Using Reliabilities of its Components", *Proc. 8th Int'l. Symp. Software Reliability Eng.*, 1997, pp. 146-155.

[8] S.Yacoub, B.Cukic and H.Ammar, "Scenario-Based Reliability Analysis of Component-Based Software", *Proc. 10th Int'l. Symp. Software Reliability Eng.*, 1999, pp. 22-31.

[9] K.Goševa-Popstojanova, K.S.Trivedi, "Architecture Based Approach to Quantitative Assessment of Software Systems" *Performance Evaluation*, to appear.