In *Proc. of Advanced Computing (ADCOMP) '97*, Chennai, India

Structure-Based Software Reliability Prediction*

Swapna S. Gokhale and Kishor S. Trivedi
Center for Advanced Computing and Communication
Dept. of Electrical and Computer Engineering
Box 90291, Duke University, Durham, NC 27708-0291

Abstract

Prevalent approaches to software reliability modeling are black-box based, i.e., the the software system is considered as a whole and only its interactions with the outside world are modeled without looking into its internal structure. However, with the advancement and widespread use of object oriented systems design and development, the use of component-based software development is on the rise. Software systems are developed in a heterogeneous (multiple teams in different environments) fashion, and hence it may be inappropriate to model the overall failure process of such systems using only one of the several software reliability growth models. In this paper we outline the constituents of the structural models. We then present a exhaustive analyses of the classes of methods where the architecture of the application is modeled either as a discrete time Markov chain (DTMC) or a continuous time Markov chain (CTMC), and illustrate these methods using examples.

1 Introduction

The impact of software structure on its reliability and correctness has been highlighted as early as 1975 by Parnas [4]. Prevalent approaches to software reliability modeling however are black-box based, i.e., the software system is considered as a whole and only its interactions with the outside world are modeled, without looking into its internal structure. With the advancement and widespread use of object oriented systems design and development, the use of component-based software development is on the rise. The software components can be commercially available off the shelf (COTS), developed in-house, or developed contractually. Thus, the whole application is developed in a heterogeneous (multiple teams in different environments) fashion, and hence it may be inappropriate to model the overall failure process of such applications using only one of the several software reliability growth models (black-box approach). Predicting reliability of such heterogeneous application taking into account the information about its architecture, testing and reliabilities of its components, is thus absolutely essential.

In this paper, we outline the constituents of the structural models for the prediction of software reliability. We characterize the structural approaches to reliability prediction, based on the models used to define the architecture of the software, failure behavior of its components and that of the interfaces between them, and the method of analyses they employ. We then present an exhaustive analyses of the classes of methods where the architecture of the application is modeled either as a discrete time Markov chain (DTMC) or a continuous time Markov chain (CTMC), and illustrate these methods using examples.

The layout of the paper is as follows: Section 2 discusses the constituents of the structural models, Section 3 gives a brief overview of discrete and continuous time Markov chains, Section 4 describes various DTMC and CTMC techniques for reliability prediction, Section 5 illustrates the techniques presented in Section 4 with various examples, and Section 6 presents conclusions and directions for future research.

2 Constituents of Structural Models

The constituents of structural models include:

Architecture of the software: This is the manner in which the different modules of the software interact. The architecture may also include information about the execution time (mean,variance, distribution) of each module. The architecture of the application can be modeled either as a DTMC (Discrete Time Markov Chain), CTMC (Continuous Time Markov Chain), SMP (Semi-Markov Process) or a DAG (Di-

*Supported by a contract from Charles Stark Draper Laboratory and in part by Bellcore as a core project in the Center for Advanced Computing and Communication

rected Acyclic Graph). DTMC, CTMC, and SMP can be further classified into irreducible and absorbing categories, where the former represents an infinitely running application, and the latter a terminating one. DAG models concurrent software having no loops and can be used to predict time to completion of a concurrent program [5]. Transitions among modules could either be instantaneous or there could be an overhead in terms of time.

Failure behavior: The failure behavior of the modules and that of the interfaces between the modules, can be specified in terms of the probability of failure (or reliability), or failure rate (constant/time-dependent). The time-dependent failure rate could be the failure rate associated with one of the reliability growth models [2].

The architecture of the software can be combined with the failure behavior of the modules and the interfaces into a composite model which can then be analyzed to predict reliability of the software. The composite model also enables the computation of various performance measures such as the time to completion of the application. We will refer to this method of reliability prediction as the "Composite Method". The other possibility is to solve the architectural model and superimpose the failure behavior of the modules and that of the interfaces on to the solution, to predict reliability. We refer to this method as "Hierarchical Method".

We characterize every approach by a tuple $(Method, Model, Modules/Interfaces, Trans.)$, where $Method$ indicates the method of analysis, $Model$ indicates the architectural model, $Modules/Interfaces$ indicates the failure behavior of the modules/interfaces, $Trans.$ indicates the nature of transitions (timed/instantaneous).

3 Overview of Markov Models

In this section we provide an overview of discrete time Markov chains (DTMCs) and continuous time Markov chains. The interested reader is referred to [5, 6] for a detailed study of these topics.

Let $X(t)$ denote a discrete state stochastic process. Without loss of generality, we assume the discrete state space $I \in \{0, 1, 2, \ldots\}$. Let $P(X(t) = j)$ denote the probability that the process is in state j at time t. $X(t)$ is a Markov chain iff, for any ordered times $t_0 < t_1 < t_2 \ldots < t_n < t$, the conditional probability mass function of $X(t)$ for given values of $X(t_0), X(t_1), \ldots, X(t_n)$, depends only on $X(t_n)$.

Time-dependent state probabilities of the Markov chain at time t are defined by the vector:

$$\pi(t) = [\pi_0(t), \pi_1(t), \ldots] \quad (1)$$

where $\pi_j(t) = P(X(t) = j)$. The objective of the analysis of a Markov chain is to obtain these state probabilities at specific values of t, and in the steady state $(t = \infty)$.

For our purposes, Markov chains can be classified into the following two categories:

- Irreducible: A Markov chain is said to be irreducible if every state can be reached from every other state.
- Absorbing: A Markov chain is said to be absorbing, if there is at least one state i, from which there is no outgoing transition. A Markov chain upon reaching an absorbing state is destined to remain there forever.

Depending upon the points in time when the transitions can take place, Markov chains can be classified as discrete time or continuous time as discussed in the subsequent subsections.

3.1 Discrete Time Markov Chains

In case of a discrete time Markov chain (DTMC), the points at which the state changes occur are discrete, and without loss of generality we let the parameter space $T \in \{0, 1, 2, \ldots\}$.Let $\mathbf{P} = [p_{ij}]$ be the one-step transition probability matrix. $\mathbf{P}$ is a stochastic matrix since all the elements in a row of $\mathbf{P}$ sum to one, and each element lies in the range $[0, 1]$.

The state probability vector at time step n denoted by $\pi(n)$ in terms of the initial probabilities $\pi(0)$, is given by [6]:

$$\pi(n) = \pi(0)\mathbf{P}(n) \quad (2)$$

The matrix $\mathbf{P}(n)$ in Equation (2) is called the n-step transition probability matrix of the DTMC. Let $p_{ij}(n)$, denote the (i, j)-th entry of the matrix $\mathbf{P}(n)$. $p_{ij}(n)$ denotes the probability of reaching state j at time step n, starting from state i. For steady state analysis of a finite, irreducible and aperiodic DTMC, we take limits on both sides of Equation (2). This gives us the following system of equations for computing the probability vector in the steady state:

$$\pi = \pi.\mathbf{P}, \qquad \pi.\mathbf{e} = 1 \quad (3)$$

where $\mathbf{e} = (1, 1, \ldots, 1)^T$ and the subscript T denotes the transpose.

In the case of a DTMC with one or more absorbing states, the expected number of times the application visits state j, denoted by V_j, can be computed by solving the following system of linear equations:

$$V_j = \sum V_i p_{ij} + \pi_j(0) \quad (4)$$

where $\pi(0)$ denotes the initial state probability vector.

3.2 Continuous Time Markov Chains

The analysis of the continuous time or continuous parameter Markov chains is similar to the discrete parameter case, except that the transitions can occur at any instant of time.

Let the infinitesimal generator matrix $\mathbf{Q} = [q_{ij}]$ of the CTMC be defined so that q_{ij} is the rate of transition from state i to state j, and:

$$q_{ii} = -\sum_{i \neq j} q_{ij} \tag{5}$$

The state probability vector $\pi(\mathbf{t})$ for a CTMC obeys the Kolmogorov differential equation:

$$\frac{d\pi(t)}{dt} = \pi(t)\mathbf{Q} \tag{6}$$

For steady state analysis, taking limits of both sides of Equation (6), we have the following system of equations:

$$\pi.\mathbf{Q} = 0 \qquad \pi.\mathbf{e} = 1 \tag{7}$$

The (i,j)-th entry of the matrix $\mathbf{Q}$ denoted by q_{ij} is the rate at which the system goes from state i to state j, and $-q_{ii}$ is the rate at which the system departs from state i.

The average amount of time spent by the CTMC in state i during the interval $(0,t)$, denoted by $L_i(t)$ is given by:

$$L_i(t) = \int_0^t \pi_i(x)dx \tag{8}$$

Integrating both sides of Equation (6), we obtain a differential equation for the vector $\mathbf{L}(t)$:

$$\frac{d\mathbf{L}(t)}{dt} = \mathbf{L}(t)\mathbf{Q} + \pi(0) \tag{9}$$

For a CTMC with one or more absorbing states, Equation (9) becomes:

$$\tau\mathbf{Q} = \pi(0) \tag{10}$$

where

$$\lim_{t\to\infty} \mathbf{L}(t) = \tau \tag{11}$$

where τ_i represents the expected time spent in module i, until absorption.

4 Reliability Prediction

In this section, we discuss various approaches to reliability prediction when the architecture of the application is modeled either by a DTMC or a CTMC. Some of these approaches have been presented in the literature, and are discussed here for the sake of completeness, and to highlight their merits and demerits in the light of the exhaustive analysis presented here. Each method will be identified by the tuple in Section 2. We assume a single entry and exit state for a terminating application.

4.1 DTMC Based Approaches

DTMC_1:$(Composite, A-DTMC, R/1, \infty)$
This method has been discussed by Cheung [1]. The architecture of the model consists of the transition probability matrix $W = w_{ij}$, which is the probability that the transition (i,j) is taken when the control is at node i. The reliability of node i is assumed to be R_i. Let the entry node be denoted by 1, which is the initial state of the DTMC. The terminal states C and F are added, which represent the state of correct output and failure respectively. For every node i, a directed branch (i,F) is created with transition probability $(1-R_i)$, representing the occurrence of an error in the execution of module i. The original transition probability between i and j is modified to $R_i w_{ij}$, which represents the transfer of control to j from i, conditional to the successful execution of i. Thus $p_{ij} = R_i w_{ij}$. A directed branch is created from the exit node n to C, with transition probability R_n, which represents the correct termination at the exit node. Note that $w_{ij} = 0$ if the branch (i,j) does not exist. The reliability of the application after k steps, is the probability of reaching the terminal state C, starting from the initial state 1 and is given by $p_{1C}(k)$. The reliability of the application (without qualification) is given by $\sum_{k=0}^{\infty} p_{1C}(k)$.

DTMC_2: $(Composite, I-DTMC, R/1, \infty)$
The composite model of an irreducible DTMC representing an infinitely running application along with the reliabilities of the components, is constructed by adding a terminal state F which corresponds to the failure of the application. We can compute the mean time to failure of the application in this case.

DTMC_3: $(Hierarchical, A-DTMC, R/1, \infty)$
The reliability of the application when the architecture is represented by an absorbing DTMC and the reliabilities of the components are known, using hierarchical method of analyses is given by:

$$R = \prod_{i=1}^{n-1} {R_i}^{V_i} R_n \tag{12}$$

where R_i and V_i denote the reliability and the expected number of visits to module i respectively. If t_i denotes the expected time spent in module i per visit, then the expected execution time $\bar{t}$ of the application is given by [6]:

$$\bar{t} = \sum_{i=1}^{n-1} V_i t_i + t_n \tag{13}$$

DTMC_4: $(Hierarchical, A-DTMC, CFR/1, \infty)$
The reliability of the application when the architecture

is modeled by an absorbing DTMC and a constant failure rate is associated with each of the components is:

$$R = \prod_{i=1}^{n} e^{-\lambda_i t_i V_i} \tag{14}$$

where V_i denotes the expected number of visits to component i, and t_i is the expected time spent in component i per visit.

DTMC_5: $(Hierarchical, A-DTMC, TDR/1, \infty)$
When a time dependent failure rate is associated with each module, and the architecture of the application is modeled by an absorbing DTMC, its reliability is given by:

$$R = \prod_{i=1}^{n} e^{-\int_0^{V_i t_i} \lambda_i(\tau) d\tau} \tag{15}$$

where V_i and t_i are as defined before. $V_i t_i$ is thus the expected total time spent in module i per execution of the application.

DTMC_6: $(Hierarchical, I-DTMC, R/1, \infty)$
The reliability of an application when the architecture is defined by an irreducible DTMC and the reliabilities of the components are known, is given by:

$$R = \sum_{i=1}^{n} \pi_i R_i \tag{16}$$

where π_i is the probability that the application is executing module i, in the steady state.

DTMC_7: $(Hierarchical, I-DTMC, CFR/1, \infty)$
The overall failure rate and the reliability of the application denoted by λ and $R(t)$ respectively, when the architecture of the application is modeled by an irreducible DTMC, and a constant failure rate is associated with its components is given by:

$$\lambda = \sum_{i=1}^{n} \pi_i \lambda_i, \qquad R(t) = e^{-\lambda t} \tag{17}$$

4.2 CTMC Based Approaches

CTMC_1: $(Composite, A-CTMC, CFR/1, \infty)$
The architecture of the software is described by a CTMC, by specifying a distribution for the execution time for each module, and intermodular transition probabilities. The terminal states C and F which represent successful and abnormal termination of the application respectively, are added, and the infinitesimal generator matrix $\mathbf{Q}_c$ corresponding to this composite model is constructed in the following manner: For every node i, a directed branch (i, j) is created with rate $\mu_i w_{ij}$, where the distribution of the execution time of module i, is given by $exp(\mu_i)$, and w_{ij} is the probability that the control is transferred to module j after the successful execution of module i. Thus $q_{ij} = \mu_i w_{ij}$. An additional directed branch is created from every node i to node F, with a transition rate λ_i, corresponding to the failure of node i. A branch is added from node n to C with rate μ_n corresponding to the successful execution of the application. The composite model can then be solved to compute the probability of being in state C, corresponding to the successful execution of the application. The distribution of the time to completion of the application can also be obtained in this case. $\pi_F(t)$ denotes the probability of failure at time t, $\pi_F(\infty)$ denotes the eventual probability of failure, while $\pi_C(t)$ denotes the probability of successful execution of the application by time t.

CTMC_2: $(Composite, I-CTMC, CFR/1, \infty)$
The infinitesimal generator matrix in this case, is also constructed by adding a state F corresponding to the failure of the application. The addition of this state transforms the irreducible CTMC into an absorbing one, with a single absorbing state. Given infinite time, the application will reach this absorbing state with probability 1. The mean time to failure (MTTF) of the application is given by:

$$MTTF = \sum_{i=1}^{n} \tau_i \tag{18}$$

where τ_i are as defined in Equation (11).

CTMC_3: $(Hierarchical, A-CTMC, R/1, \infty)$
When the architecture of the application is represented by an absorbing CTMC, and the reliabilities of the components are known, the reliability of the application using hierarchical analyses is given by Equation (12). The visit counts are computed by the analysis of the embedded DTMC.

CTMC_4: $(Hierarchical, A-CTMC, CFR/1, \infty)$
The reliability of the application with an architecture represented by an absorbing CTMC, and the components fail as per a constant failure rate is:

$$R(t) = \prod_{i=1}^{n} e^{-\lambda_i L_i(t)} \tag{19}$$

where λ_i is the failure rate of module i, and $L_i(t)$ is the expected time the application spends executing module i during a single run as per Equation (9).

CTMC_5: $(Hierarchical, A-CTMC, TDR/1, \infty)$
If the failure behavior of the components is given by a time-dependent failure rate, then the reliability is given

by:

$$R(t) = \prod_{i=1}^{n} e^{-\int_0^{L_i(t)} \lambda_i(\tau) d\tau} \quad (20)$$

where $\lambda_i(\tau)$ denotes the failure intensity of module i, and $L_i(t)$ denotes the expected time spent in module i, as given by Equation (9).

CTMC_6:$(Hierarchical, I-CTMC, R/1, \infty)$
The analyses in this case is similar to that of its DTMC counterpart, and the reliability of the application is given by Equation (16). This approach has been discussed by Laprie and Kanoun [3].

CTMC_7:$(Hierarchical, I-CTMC, CFR/1, \infty)$
The analysis here is also similar to the corresponding DTMC case, and has been discussed by Laprie and Kanoun [3]. We assume that the failure rates are small as compared to the rates governing the transitions among the components, equivalently stated a large number of transitions will take place among the components, before the occurrence of a failure. The overall failure rate and the reliability of the application are given by Equation (17).

Thus we see that composite methods of analyses are convenient, if the type of the model used to represent the architecture of the application is retained, after combining it with the failure behavior of the components. For instance, composite analysis is possible if the architecture is modeled by a DTMC, and the reliabilities of the components are given, since the combination of this information results in a DTMC model. Similarly, in case of a CTMC-based architecture, composite analysis is possible if the failure of the components is characterized by constant failure rates. Hierarchical analyses is possible in case of all the applications when the architecture is described by absorbing DTMC/CTMC, irrespective of how the failure behavior of the components is characterized. In the irreducible case, hierarchical analyses can be used to predict reliability, when the components are characterized either by their reliabilities or constant failure rates. Extensions to this basic scheme are under investigation.

5 Illustrations

In this section, we illustrate the various methods discussed in the previous section with examples. The basic application used for this purpose has been described in [1], which is an absorbing DTMC, and the reliabilities of the modules are given. This basic example is subsequently modified/updated to describe various architectures and failure behaviors. The application under consideration has 10 modules where 1 represents the entry state and 10 represents the exit state. The reliabilities of the modules are given in Table 2, and the intermodular transition probabilities are given in Table 1.

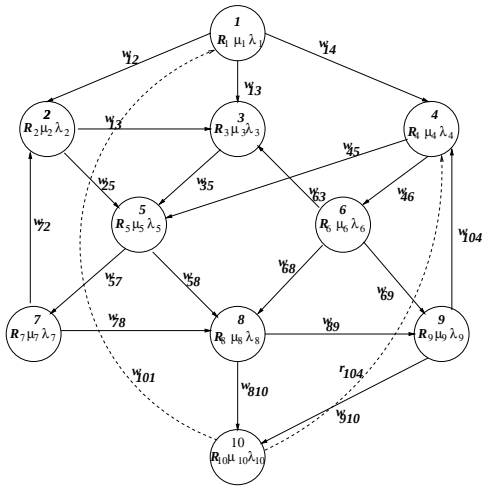

Figure 1. Example of a Program Control Flow Graph

Table 1. Intermodular Transition Probabilities

$p(1,2) = 0.60$	$p(1,3) = 0.20$	$p(1,4) = 0.20$	
$p(2,3) = 0.70$	$p(2,5) = 0.30$		
$p(3,5) = 1.0$			
$p(4,5)0.40$	$p(4,6) = 0.60$		
$p(5,7) = 0.40$	$p(5,8) = 0.60$		
$p(6,3) = 0.30$	$p(6,7) = 0.30$	$p(6,8) = 0.10$	$p(6,9) = 0.30$
$p(7,2) = 0.50$	$p(7,9) = 0.50$		
$p(8,4) = 0.25$	$p(8,10) = 0.75$		
$p(9,8) = 0.10$	$p(9,10) = 0.90$		

Table 2. $R_i/\lambda_i/\mu_i$ of the components

Module #	Reliability (R_i)	CFR (λ_i)	TTC (μ_i)
1	0.99	0.001	0.1
2	0.980	0.002	0.2
3	0.990	0.0015	0.15
4	0.970	0.0005	0.05
5	0.950	0.00075	0.075
6	0.995	0.0025	0.25
7	0.995	0.00175	0.175
8	0.950	0.00025	0.125
9	0.975	0.00225	0.225
10	0.985	0.00025	0.025

The control flow graph of the application is shown in Figure 1, where the solid transitions correspond to the basic application described in [1]. The significance of the dashed transitions is explained shortly. The reliability of the basic application as per $DTMC\_1$

is 0.8351, while using the hierarchical method as per $DTMC\_3$ is 0.8188. Assuming that the average time spent in a module per visit in a single execution of the application is 0.03, the reliability of the application as per $DTMC\_4$ is 0.8177. Without loss of generality, we assume that the time-dependent failure rate associated with each component is that of the Goel-Okumoto model [2], and is $0.34 * 0.0057 * e^{-0.0057*t}$. The reliability of the application in this case is given by 0.99. The first modification we consider is a transformation of the architecture from an absorbing DTMC to an irreducible DTMC, which we achieve by adding two directed branches $w_{101} = 0.8$, and $w_{104} = 0.2$. These two transitions are shown by dashed lines in the control flow graph in Figure 1. The reliability of the application as per $DTMC\_6$ is 0.97519. The composite model has a single absorbing state corresponding to the failure of the application, and the steady state probability of reaching that state is 1. However, we can compute mean time to failure (MTTF), according to $DTMC\_2$, which is 55.551. We next assume that the failure behavior of the modules is given by a constant failure rate as in Table 2. The overall failure rate of the application is given by 0.0011, the reliability as per $DTMC\_5$, is $R(t) = e^{-0.0011t}$, and hence the $MTTF = \frac{1}{0.0011} = 909.09$. The absorbing DTMC is then transformed to an absorbing CTMC, by associating a exponentially distributed execution time for each module, as given in Table 2. The reliability of the application as per $CTMC\_3$ is 0.8148, as per $CTMC\_1$ is 0.92664, as per $CTMC\_4$ is 0.9407, and as per $CTMC\_5$ is 0.8858. The absorbing CTMC is then transformed to an irreducible CTMC by adding the same two transitions as before. The reliability of the application as per $CTMC\_6$ is 0.97578. The overall failure rate of the application as per $CTMC\_7$ is 0.00086320, and the reliability is given by $R(t) = e^{-0.00086320}$, and hence the $MTTF$ is $\frac{1}{0.00086320} = 1158$. The mean time to failure of the application as per ($CTMC\_2$) and is 1150. Whenever both hierarchical and composite methods of analyses are feasible, the difference in the reliability predictions obtained using these two methods can be attributed to the fact that the hierarchical method provides an approximation to the reliability estimates obtained using the composite method.

6 Conclusions and Future Research

In this paper we have discussed the specification and solution techniques for the structural models to quantify software reliability. In addition, we have also proposed a classification of the techniques based on the models used to define the architecture of the application, failure behavior of the components, and the method of analyses. We have presented an exhaustive discussion of the discrete time and continuous time Markov chain based methods. We have illustrated the techniques with examples. Empirical validation of the methods discussed here is being carried out.

References

[1] R. C. Cheung. "A User-Oriented Software Reliability Model". *IEEE Trans. on Software Engineering*, SE-6(2):118–125, March 1980.

[2] W. Farr. *Handbook of Software Reliability Engineering, M. R. Lyu, Editor*, chapter Software Reliability Modeling Survey, pages 71–117. McGraw-Hill, New York, NY, 1996.

[3] J. C. Laprie and K. Kanoun. *Handbook of Software Reliability Engineering, M. R. Lyu, Editor*, chapter Software Reliability and System Reliability, pages 27–69. McGraw-Hill, New York, NY, 1996.

[4] D. L. Parnas. "The Influence of Software Structure on Reliability". In *Proc. 1975 Int'l Conf. Reliable Software*, pages 358–362, Los Angeles, CA, April 1975.

[5] R. A. Sahner, K. S. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package.* Kluwer Academic Publishers, Boston, 1996.

[6] K. S. Trivedi. *"Probability and Statistics with Reliability, Queuing and Computer Science Applications".* Prentice-Hall, Englewood Cliffs, New Jersey, 1982.