The Effects of Failure Correlation on Software Reliability and Performability [*]

Katerina Goševa – Popstojanova, Kishor Trivedi
Center for Advanced Computing and Communication
Department of Electrical and Computer Engineering
Duke University, Durham, NC 27708 – 0291, USA
E-mail: {katerina, kst}@ee.duke.edu

1. Introduction

The assumption of independence among successive software failures is common to the most of software reliability models (see [1] for recent surveys). Its validity can be affected by a variety of operational conditions, such as whether the input sequences exhibit independence, the extent to which internal state of a software has been affected, whether or not operations undertaken for execution resumption involve state cleaning.

Motivated by the fact that independence assumption can be easily violated in many practical applications we have recently developed a modeling framework, based on Markov renewal processes, which accounts for the correlation among successive failures. In this short paper we demonstrate its use for software operational life. The operation of software is divided into a sequence of runs by subdivision of time associated with some user – oriented tasks. A sequence of dependent software runs, when the outcome of each run depends on the outcome of the previous run, will result in a correlation of successive failures.

2. Markov renewal modeling framework

The flexibility of Markov renewal processes allows us to construct the model in two steps. First, we consider the outcomes of a sequence of possibly dependent software runs to build the model in discrete time. Next, we construct the model in continuous time considering the execution times of software runs.

We view the sequence of software runs in discrete time as a sequence of dependent Bernoulli trials in which the probability of success or failure at each trial depends on the outcome of the previous trial. It can be described by a discrete time Markov chain (DTMC) with two states. One of the states, denoted by 0, is regarded as success and the other, denoted by 1, as failure. DTMC transition probability matrix

$$P = \begin{bmatrix} p & 1-p \\ 1-q & q \end{bmatrix}, \qquad \begin{array}{c} 0 \le p, q \le 1 \\ |p+q-1| \le 1 \end{array} \qquad (1)$$

describes the probabilities of success (p and $1-q$) and failure ($1-p$ and q) conditioned on the outcome of the previous run (success and failure respectively). [1]

It can be shown that the unconditional per run failure probability is $\theta = (1-p)/(2-p-q)$ and the serial correlation coefficient is $\rho = p+q-1$.

If $Pr\{$(i+1)-st run fails $|$ i-th run has failed$\} \neq Pr\{$(i+1)-st run fails$\}$, that is, $q \neq \theta$ $(p+q \neq 1)$ then the assumption of independence is violated. The model admits as special cases the following:

1. Failures are independent $(\mathbf{p+q=1})$. Each run has probabilities p and $q = 1-p$ of being a success and failure independently of the outcome of previous run.

2. A lack of clustering $(\mathbf{p+q<1})$. Successive software runs are negatively correlated, that is, if a failure occurs in i-th run, there would be an increased chance that a success will occur in $(i+1)$-st run.

3. Failures occur in clusters $(\mathbf{p+q>1})$. Successive runs are positively correlated, that is, if a failure occurs in i-th run, there would be an increased chance that another failure will occur in the next $(i+1)$-st run.

The next step in the model construction is to obtain a process in continuous time by considering the distribution $F_{ij}(t)$ of the time spent in a transition from state i to state j of the DTMC. In our case $F_{ij}(t)$ are the distributions of the time that takes software runs to be executed. It is assumed that software execution time T_{ex} has the same distribution $F_{ex}(t) = P\{T_{ex} \le t\}$ regardless of the outcome, that is, $F_{ij}(t) = F_{ex}(t)$ for $0 \le i, j \le 1$. With the addition of $F_{ex}(t)$ to the

*Supported in part by the National Science Foundation, by Bellcore and by the Lord Foundation as a core project in the Center for Advanced Computing and Communication

[1]The boundary cases $p = q = 0$ and $p = q = 1$ are excluded from the analysis since they are somewhat trivial.

DTMC, we obtain the model in continuous time, that is, a Markov renewal process (MRP). The total number of software runs $\{N(t), t \geq 0\}$ is a superposition of two dependent renewal processes $N_S(t)$ and $N_F(t)$ which refer to the number of times states 0 (success) and 1 (failure) of the DTMC have been visited in $(0, t]$.

The following analysis is aimed at determinating the effects of correlation among successive failures on software reliability and performability measures. Note that, the presented model allows many other measures that are not considered in this paper to be derived.

The distribution of the time to next failure T, that is, the unreliability is the interval distribution of the point process $N_F(t)$. It is determinated by considering DTMC first in order to derive the *pmf* of a discrete random variable X defined as the number of runs between two successive visits to the failure state:

$$Pr\{X = k\} = \begin{cases} q & \text{if } k = 1 \\ (1-q)\, p^{k-2}\, (1-p) & \text{if } k \geq 2. \end{cases} \tag{2}$$

Then, using (2) it follows that the distribution of the time to next failure $F(t) = Pr\{T \leq t\}$ is given by:

$$F(t) = q\, F_{ex}(t) + \sum_{k=2}^{\infty} (1-q)\, p^{k-2}\, (1-p)\, F_{ex}^{k*}(t) \tag{3}$$

where F_{ex}^{k*} denotes the k-fold convolution of F_{ex}. The Laplace – Stieltjes transform (LST) of $F(t)$ is

$$\tilde{F}(s) = \frac{q\, \tilde{F}_{ex}(s) + (1-p-q)\, \tilde{F}_{ex}^2(s)}{1 - p\, \tilde{F}_{ex}(s)} \tag{4}$$

where $\tilde{F}_{ex}(s)$ is the LST of $F_{ex}(t)$. Depending on the particular distribution of the execution time $F_{ex}(t)$, the expression (4) can be inverted either symbolically or numerically. In either case, mean time to failure (MTTF) is derived by a simple differentiation of (4):

$$E[T] = \frac{2-p-q}{1-p}\; E[T_{ex}] = (2-p-q)\, E[T^{in}] \tag{5}$$

where $E[T_{ex}]$ is the run's mean execution time, while $E[T^{in}] = E[T_{ex}]/(1-p)$ if the MTTF for the independent case.

Equation (5) clearly demonstrates the effects of failure correlation on software reliability measures: if there is a luck of clustering MTTF would be greater, while if failures occur in clusters MTTF would be smaller compared to the independent case.

Performability measure $U(t)$ is defined as a number successful runs that benefit the user during a time period $(0, t]$. If a software fails at any run $N_F(t) > 0$ no run either prior or subsequent to such a failure is considered beneficial $U(t) = 0$. If $N_F(t) = 0$ then all runs during $(0, t]$ are successful and beneficial $U(t) = N(t)$. The moment generating function of $U(t)$ is

$$E[e^{sU(t)}] = E[1 - (1-q)\, p^{N(t)-1} + e^{sN(t)} (1-q)\, p^{N(t)-1}]$$

and its expectation can be expressed as

$$E[U(t)] = \left. \frac{dE[e^{sU(t)}]}{ds} \right|_{s=0} = \frac{1-q}{p}\; E[U^{in}(t)] \tag{6}$$

where $E[U^{in}(t)] = E[N(t)\, p^{N(t)}]$ is the benefit that can be expected if the successive failures were independent.

Equation (6) reveals the impact of failure correlation on performability. If failures do occur in clusters $E[U(t)]$ is reduced by factor $(1-q)/p$ compared to the independent case. On the other hand, a luck of clustering has just an opposite effect.

3. Conclusion

The presented Markov renewal approach contributes toward more realistic modeling of software as it enables to account for the correlation among successive failures and to study its effects on software reliability and performability. Our approach can be easily extended to cover the following generalizations.

- In real – time applications T_{ex} will be the time upon the end of execution or upon reaching a deadline τ, whichever occurs first, that is, its *CDF* will coincide with $F_{ex}(t)$ for $0 \leq t < \tau$, otherwise it will be equal to 1.
- Different execution time distributions for successful and failed runs can be considered by assigning distribution $F_S(t)$ to state 0 and $F_F(t)$ to state 1.
- More than one type of failures or periods of time when the software is idle can be accounted for by adding suitable states to the DTMC.
- This approach can be used for modeling fault tolerant software (FTS) systems. Per run failure probability and run's execution time distribution for a particular FTS structure can be derived using a variety of existing FTS models (see [2], [3] and references therein). Thus, in addition to the interversion failure correlation on a single run considered in related works, our approach enables to account for the correlation among successive failures.

References

[1] M.R.Lyu (Ed.), *Handbook of Software Reliability Engineering*, McGraw-Hill, 1996.

[2] A.T.Tai, J.F.Meyer, A.Avižienis, "Performability Enhancement of Fault – Tolerant Software", *IEEE Trans. on Reliability*, Vol.42, No.2, June 1993, pp. 227 – 237.

[3] K.Goševa – Popstojanova, A.Grnarov, "Hierarchical Decomposition for Estimating Reliability of Fault – Tolerant Software in Mission – Critical Systems", *Proc. IASTED Int'l Conf. Software Engineering*, Nov 1997, pp. 141 – 146.