FAILURE PROCESS SIMULATION OF A COMPONENT-BASED SOFTWARE

Florentina Suter

University of Bucharest
Department of Mathematics and Computer Science
and Centre of Mathematical Statistics of the Romanian Academy

florentina.suter@g.unibuc.ro

ABSTRACT
In order to characterize as realistically as possible the evolution of software in time, the software reliability models should take into account the structure of the software. Such models are component-based models in which the software is not a black-box, but has several interconnected components. For this kind of software reliability models, due to their complexity, mathematical tractability becomes difficult to obtain. Therefore, thanks to its flexibility, simulation is a natural choice for analyzing the failure process and for estimating software reliability. Recently some software reliability models which take into consideration the modular structure of the software were described. In this paper we take as starting point a component based software reliability model, we describe a generalization of it and we use discrete-event simulation to analyze the software failure process.

Keywords: software reliability, component-based software model, discrete event simulation

1. INTRODUCTION

The main steps in the development of a software product are: requirements identification and analysis, project development, code writing and product testing. At the testing step the software is executed and at each failure one or more faults are discovered and corrected. It is not possible to correct all the faults from a software, and an important decision to be made in the testing phase is when to stop testing and release the software on the market.

An element to support this decision can be the software reliability, that is, the probability of failure free software functioning for a certain period of time, in certain environment conditions. For this reason many software reliability models were developed. These models are based mainly on two principles: the software product is a black-box and the software reliability is estimated from the probability assumptions on the software failure process. For example several software reliability models are based on the hypothesis that the counting process $\{N(t), t\geq 0\}$, where $N(t)$ is the number of the failures experienced in the time interval $(0,t]$, is a nonhomogeneous Poisson process. In (Chen and Singpurwalla 1997) a large number of software reliability models are unified by the observation that the failure times of the software are the jump times of a self-exciting process which is a generalization of the Poisson process and of the pure birth process.

An example of software reliability model based on a self-exciting point process of memory 1 can be found in (Al-Mutairi, Chen and Singpurwalla, 1998). More precisely the counting process $\{N(t), t\geq 0\}$ has the intensity functions:

$$\eta_0(t) = \frac{1}{(t/c)+(1/b)} \tag{1}$$

$$\eta_i(t;t_i) = \frac{1}{[(t-t_i)/c]+(t_i/ib)}, \quad i = 1,2,... \tag{2}$$

with b and c two positive parameters.

Recently a new approach of software reliability models is gaining importance. This approach takes into account the modular structure of a software. Some of the advantages of this approach are that it relates system reliability to its structure and the components reliabilities and it allows the analysis of the sensitivity of system reliability to the reliabilities of its components (Gokhale and Lyu 2005). The disadvantage is that, due to their complexity, mathematical tractability for this type of models becomes difficult to obtain. Therefore, thanks to its flexibility, simulation is a natural choice for analyzing the failure process and for estimating software reliability. Software reliability models which take into consideration the modular structure of the software are described in (Gokhale and Lyu 2005; Yacoub, Cukic and Ammar 2004). We take as starting point a model introduced in (Gokhale and Lyu 2005), we describe a generalization of it and we use discrete-event simulation to analyze the software failure process.

2. A COMPONENT-BASED SOFTWARE RELIABILITY MODEL

In (Gokhale and Lyu 2005) some component-based software reliability models are introduced in order to analyze the software failure process. We will focus on the model for which some assumptions from classical software reliability models are maintained considering that at each failure corresponds only a fault and that a fault is instantaneously corrected. There are also assumptions that take into account the structure of the software, thus the black-box hypothesis being relaxed. These assumptions are:

1. The software product has *k* components.
2. The components are executed sequentially beginning with component 1 and terminating with component *k*.
3. The component *j* is executed upon the completion of component *i* with the probability p_{ij} .
4. The failures of a component are independent from the failures of others components.
5. The time spent in each component is a random variable.

Moreover, one supposes that the behaviour of each component from the failure process point of view is modelled using a well-known software reliability model based on the nonhomogeneous Poisson process, the Goel-Okumoto model.

Taking into account all these assumptions it results a model which is complex and difficult to handle from computational point of view. Therefore in (Gokhale and Lyu 2005) the discrete-event simulation is used to study the failure process of a software whose structure and behaviour are described by the above hypothesis. As events they consider the transfer of control among components and the failure of the components. It is a rate-based simulation which has as input: the length of testing time duration, the time step and the failure rate of the component. The simulation procedure returns the total number of the failures observed in the time interval (0,*t*].

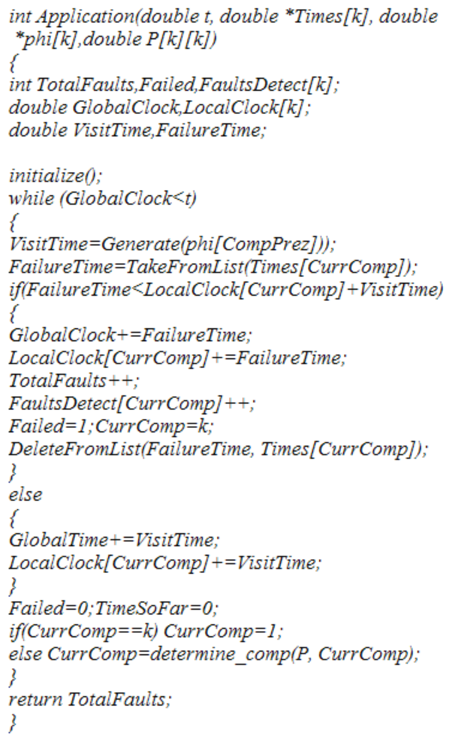
3. SIMULATION OF A SOFTWARE TESTING PROCEDURE

We consider the previous described model and we generalize it in the following way:

- instead of generating possible failures for some constant time intervals, we generate failure times using a generalization of inverse method;
- instead of considering the Goel-Okumoto model as a software reliability model for each component we consider a more general model in which the failure process is a self-exciting process.

In this way we avoid the possible problems that the choice of the length of time step could create. Moreover using the self-exciting point process we obtain a more general simulation model, such that we can associate to software components any software reliability model that is generalized by self-exciting process model. We analyze the application of our model in the case in which the failure process of the components has the characteristics described in (Al-Mutairi, Chen and Singpurwalla, 1998).

Our application is:



```
int Application(double t, double *Times[k], double
 *phi[k],double P[k][k])
{
int TotalFaults,Failed,FaultsDetect[k];
double GlobalClock,LocalClock[k];
double VisitTime,FailureTime;

initialize();
while (GlobalClock<t)
{
VisitTime=Generate(phi[CompPrez]));
FailureTime=TakeFromList(Times[CurrComp]);
if(FailureTime<LocalClock[CurrComp]+VisitTime)
{
GlobalClock+=FailureTime;
LocalClock[CurrComp]+=FailureTime;
TotalFaults++;
FaultsDetect[CurrComp]++;
Failed=1;CurrComp=k;
DeleteFromList(FailureTime, Times[CurrComp]);
}
else
{
GlobalTime+=VisitTime;
LocalClock[CurrComp]+=VisitTime;
}
Failed=0;TimeSoFar=0;
if(CurrComp==k) CurrComp=1;
else CurrComp=determine_comp(P, CurrComp);
}
return TotalFaults;
}
```

Figure 1: A software failure process simulation model

As in (Gokhale and Lyu 2005) our application simulates the execution of a software with *k* components and returns the total faults detected in the time interval [0,*t*). The input parameters for our application are

- *t* the length of the testing time interval;
- **Times[k]* an array in which each element is a list of failure times of a component
- **phi[k]* an array containing information regarding execution time spent in each component
- *P[k][k]* a matrix whose elements are intercomponent transition probabilities.

We suppose that the software execution begins with the component number 1. The testing time of each component is randomly generated. If an error occurs, i.e. there is a failure time whose value is in the testing time interval of that component, then the error is counted and the test restarts with the execution of the first component. If any error occurs, next component to be tested is chosen using the transition matrix *P[k][k]*.

The main difference between our model and the model introduced in (Gokhale and Lyu 2005) is that we generate failure times of different components. More precisely we generate random vectors whose elements represent failure times in time interval $[0,t)$ of one component of the software. For the generation of these random vectors we can use the generalized inverse method (Văduva, 1994). We can apply the generalized inverse method for generating jump times of a self-exciting point process for which the intensity function is known (Suter, 2004). For example if we apply the generalized inverse method to the point process considered in (Al-Mutairi, Chen and Singpurwalla, 1998) we obtain the following relationships for generating jump times of the point process:

$$\hat{T}_1 = \frac{c}{b}\left(U_1^{-\frac{1}{c}} - 1\right) \tag{3}$$

$$\hat{T}_i = \frac{c}{b}\hat{T}_{i-1}\left(U_i^{-\frac{1}{c}} - 1\right),\ i = 2,3,...,n \tag{4}$$

where U_1, U_2,…,U_n are independent uniform random variables.

REFERENCES

Al-Mutairi, D., Chen, Y., and Singpurwalla, N. D., 1998. An adaptive concatenated failure rate model for software reliability. *Journal of the American Statistical Association*, 93 (443), 1150–1163.

Chen, Y., and Singpurwalla, N. D., 1997. Unification of software reliability models by self-exciting point processes. *Advances in Applied Probability*, 29 (2), 337–352.

Gokhale, S., and Lyu, M., 2005. A simulation approach to structure-based software reliability analysis. *IEEE Transactions on Software Engineering*, 31 (8), 643–657.

Snyder, D. L., 1975. *Random point processes*. John Wiley & Sons.

Suter, F., 2004. *Models and Algorithms in Systems Reliability*. Thesis (PhD), University of Bucharest.

Văduva, I. (1994) *Fast algorithms for computer generation of random vectors used in reliability and applications*. Tech. Rep. 1603, Technische Hochschule Darmstadt, Zentrum für Praktische Mathematik.

Yacoub, S., Cukic, B., Ammar, H., 2004 A Scenario-Based Reliability Analysis Approach for Component-Base Software. *IEEE Transactions on Reliability*, 53 (4), 465–480.