Anticipatory Models of Software Reliability

Lorina Negreanu, Eugenia Kalisz, Irina Mocanu
University Politehnica of Bucharest, Computer Science and Engineering Dept.
Splaiul Independentei 313, sector 6
060042, Bucharest, Romania
lorina.negreanu@cs.pub.ro , http://prof.cs.pub.ro/~lorina
eugenia.kalisz@cs.pub.ro , http://turing.cs.pub.ro/~ekalisz
irina.mocanu@cs.pub.ro , http://turing.cs.pub.ro/~irinam

Abstract
A software reliability model specifies the general form of dependencies of the failure process on factors like fault introduction, fault removal and use. Because some of the foregoing factors are probabilistic in nature and operate over time, the software reliability models are formulated in terms of random processes. The paper investigates the anticipatory property of software reliability models, focusing on the geometric family of reliability models.
Keywords: Anticipatory systems, software reliability, models.

1 Introduction

Reliability is the probability that a system functions without failure (a departure of system behavior in execution from user needs) for a specified time in a specified environment (Musa, 1998). To model software reliability one must first consider the main factors that affect it: fault introduction, fault removal, and use. Fault introduction depends primarily on the characteristics of the product and the development process. Fault removal depends on time, the operational profile (a complete set of functions with their probabilities of occurrence) used in test, and the quality of the removal activity. Use is characterized by the operational profile.

A software reliability model specifies the general form of the dependence of the failure process on the mentioned factors. It is assumed, by definition, time-based. The possibilities for different mathematical forms to describe the failure process are almost limitless. We have restricted ourselves to considering well-developed models that practitioners have applied fairly broadly with real data, experiencing reasonable results.

2 General Characteristics of Software Reliability Models

A software reliability model usually has the form of a random process that describes the behavior of failures with respect to time. Specification of the model generally includes specification of a function of time, such as the mean value function (expected number of failures) or failure intensity (failures per time unit). The parameters of the function are primarily dependent on fault removal activity and properties of the software product and the development process. Properties of the product include size, complexity, and structure. The most significant product characteristic is the size of the developed code. Properties of the development process include software engineering technologies, tools used, and experience level of personnel. The time involved in the characterization of the models is a cumulative time. The origin may be arbitrarily set, frequently the start of the system test.

Software reliability models almost always assume that failures are independent of each other through assuming that failure times are independent of each other or by making the Poisson process assumption of independent increments. This condition would appear to be met for most situations. Failures are the result of two processes: the introduction of faults and their activation through selection of the input states. Because both of these processes are random, the chance that one failure is influenced by another is quite small.

Both the human error process that introduces defects into code and the run selection process that determines which code is being executed at any time are dependent on an enormous number of time-varying variables. The use of a random process model is appropriate for such a situation (Musa, 1998). There are two equivalent ways to describe the failure random process: the times of failures or the number of failures in a given period.

Let T_i and T_i' denote the random variables representing time to the ith failure and time between failures $(i-1)$ and i, respectively. The realizations (specific instances) of T_i and T_i' are denoted by t_i and t_i', respectively. Let $M(t)$ be a random process representing the number of failures experienced at time t. The realization of this random process are denoted $m(t)$. The mean value function $\mu(t)$ is defined as:

$$\mu(t) = E(M(t))$$

which represents the expected number of failures at time t. It is assumed that the function $\mu(t)$ is a nondecreasing, continuos, and differentiable function of time t. The failure intensity function of the $M(t)$ process is the instantaneous rate of change of the expected number of failures with respect to time, defined by:

$$\lambda(t) = \frac{d\mu(t)}{dt}$$

3 Anticipatory Property of Reliability Models

We will investigate the anticipatory behavior of software reliability models based on the projective validity property of the models. Projective validity is the capability of the model to project future behavior from present and past failure behavior. This capability is significant only when failure behavior is changing. Hence it is usually considered for a test phase, but it can be applied to the field when repairs are being regularly made.

Software reliability engineering in current practice estimates just current failure intensity (Musa, 1998), which is effectively a projection of zero execution time. In analyzing the anticipatory property of models, we use a more stringent criterion of nonzero projection time so that we select models with the potential to handle more demanding practice needs in the future.

We use a simple number of failures approach, because it is easy to understand and apply. Time is characterized as execution time, τ. Such a counting process is characterized by specifying the distribution of $M(\tau)$, including the mean function $\mu(\tau)$.

Assume that q failures have been observed by the end of the test time τ_q. Failure data up to time $\tau_e(\leq\tau_q)$ is used to estimate the parameters of $\mu(\tau)$. Substituting the estimations of the parameters in the mean value function yields the estimate of the number of failures $\widehat{\mu}(\tau_q)$ by τ_q. This procedure is repeated for various values of τ_q.

The projective validity can be visually checked by plotting the relative error $(\widehat{\mu}(\tau) - q)/q$ against the normalized test time τ_e/τ_q. The error will approach 0 as τ_e approaches τ_q. If the points are positive (negative), the model tends to overestimate (underestimate). Numbers closer to 0 imply more accurate projection and hence a better model.

The use of normalization enables one to overlay relative error curves obtained from different failure data sets. For an overall conclusion about the relative projective validity models, one can compare plots of the medians (taken with respect to the various data sets). A model is considered superior if it yields the curve closest to 0.

Any capability of a model for projection of software reliability in the system design and early development phases is extremely valuable because of the resultant value for system engineering and planning purposes. The projections must be taken through measurable characteristics (size, complexity, structure, etc.), the software development environment, and the operational environment.

4 Analysis of the Anticipatory Property

To illustrate the analyzing method, we considered the projective validity of the geometric family. We used the logarithmic Poisson execution time model, for the data set provided by Musa (Musa, 1979), shown in table 1:

Table 1. Data Set Sample

Reference source	Delivered object instructions	Program mers	Total test time		Size of failure sample	Nature of system
			Execution time (h)	Calendar time (days)		
Musa (1979)	21700	9	24,6	92	136	Real time command and control

The logarithmic Poisson execution time model has its mean value function given by the equation:

$$\mu(\tau) = \frac{1}{\theta}\ln(\lambda_0\theta\tau + 1)$$

and intensity function given by the equation:

$$\lambda(\tau) = \frac{\lambda_0}{\lambda_0 \theta \tau + 1}$$

The estimates of the model parameters λ_0 and θ must be obtained first. The estimates $\widehat{\lambda_0}$ and $\widehat{\theta}$ have been modeled as anticipatory values based on the failure data up to execution time values of τ_e. These values range from 20 to 100 percent, in increments of 5 percent, of the total execution time $\tau_q = 24.6$ execution hours. Table 2 summarizes the results.

Table 2. Maximum Likelihood Estimates, Predicted Number of Failures and Relative Errors Based on the Logarithmic Poisson Execution time Model

τ_e/τ_q (%)	λ_0 (failures per execution hours)	$\widehat{\theta}$	$\widehat{\mu}(\tau_q)$ (failures)	$(\widehat{\mu}(\tau_q)$ - q) / q (%)
20	40,5	0,0241	134	-1,78
25	41,3	0,0236	136	0,19
30	39,7	0,0251	129	-5,21
35	41,2	0,0232	138	1,45
40	40,4	0,0246	132	-3,25
45	40,4	0,0241	134	-1,84
50	40,8	0,0239	135	-0,86
55	41,2	0,0239	135	-0,56
60	42,9	0,0232	140	2,75
65	41,6	0,0249	132	-3,22
70	41,6	0,0241	135	-1,00
75	43,7	0,0229	142	4,05
80	43,2	0,0244	135	-0,63
85	43,2	0,0241	136	0,14
90	41,1	0,0255	129	-5,11
95	43,1	0,0229	141	3,62
100	43,1	0,0241	136	0,07

The estimates $\widehat{\lambda_0}$ and $\widehat{\theta}$ have been modeled as follows:

$$\lambda_{0\tau} = \lambda_{0\tau-1} + \Delta\lambda_{0\tau-1}$$

$$\theta_\tau = \theta_p - (\Delta\lambda_{0\tau-1}\theta_p)/(\Delta\lambda_{0\tau-1} + \lambda_{0p})$$

where:

$\Delta = random(-5;5)/100$

and

$$\lambda_{0p} = 40,5$$
$$\theta_p = 0,0241$$

are predicted values.

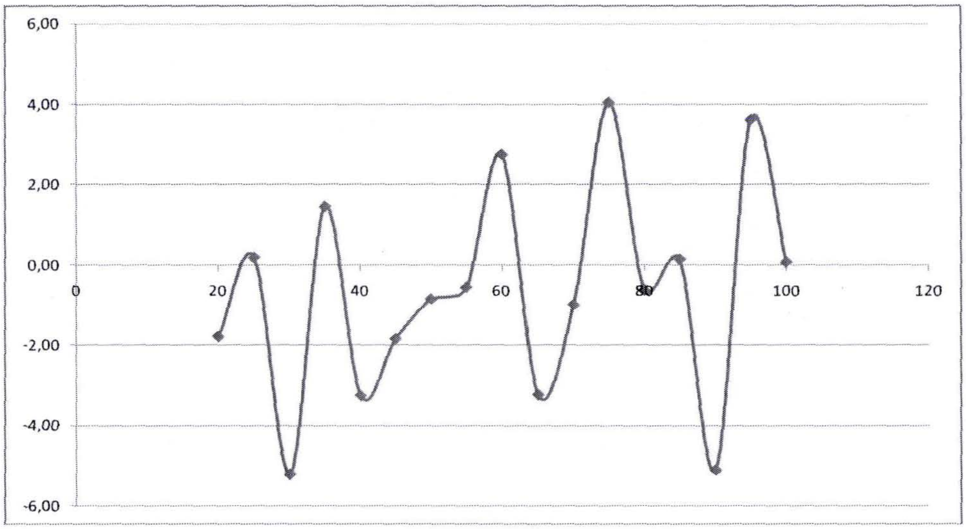
For example, for the failure data up to 60 percent of the total execution time (that is, τ_e = 14.8 execution hours), $\widehat{\lambda_0}$ = 42,9 failures per execution hour and $\hat{\theta}$ = 0.0232 per failure. The fitted mean value function can be obtained by substituting these estimates into the mean value function as:

$$\hat{\mu}(\tau) = \frac{1}{\hat{\theta}} \ln(\widehat{\lambda_0}\hat{\theta}\tau + 1) = \frac{1}{0.0232} \ln\,[(42.9)(0.0232)\tau + 1]$$

The failures experienced by the end of testing τ_q can be projected by evaluating the fitted mean value function at $\tau = \tau_q$. The value obtained is $\hat{\mu}(\tau_q)$ = 140. Note that there were 136 failures experienced at the end of the testing. Therefore the relative error in projection can be computed as:

$$\frac{\hat{\mu}(\tau_q) - 136}{136} = \frac{140 - 136}{136} = 0.0294$$

In other words, for the 60 percent of the total failure data, the logarithmic Poisson execution time model, using the maximum likelihood estimation method, overestimates by 2.9 percent. Table 2 also shows the projected values and relative errors for execution time values of τ_e that are from 20 to 100 percent of τ_q in increments of 5 percent.



Normalized execution time

Figure 1. Relative error curve for logarithmic Poisson execution time model

Figure 1 shows the relative errors plotted against the normalized execution time (τ_e/τ_q). Note that the error will approach zero as τ_e approaches τ_q. Positive values of error indicate overestimation; negative values indicate underestimation. Numbers closer to zero imply more accurate projection. Figure 1 proves that the model projects the future behavior well for this data set. The error curve is mostly within ± 5 percent.

5 Conclusions

The paper focuses on the investigation of the anticipatory property of the reliability models. We consider that the projective validity property of the reliability models might be used as a weak anticipatory property (Dubois, 2000).

We have analyzed the projective validity property of the geometric family, the logarithmic Poisson execution time model, on a sample data set. The model seems to project the future behaviour well because the error curve is, in general, within ± 5 percent when projection made after 50 percent of the total execution time. Furthermore, there is no specific pattern such as overestimation or underestimation. The results enable us to consider projective validity property as a good candidate to express the weak anticipatory property of a reliability model.

The reliability models were analyzed from the anticipation point of view using an intrinsic property. It would be very useful to use an anticipation model to relate the execution time component parameters of a reliability model to characteristics of the software product, the development process, and the execution environment, this way enabling prediction of the parameters before execution. Once the software is executed and failure data is available, the parameters can be estimated statistically from the data.

Prediction of the parameters of a reliability model is a difficult problem, and an open field of research. Although there are many proposed solutions (Bagchi, 2009; Mao, 2009; Kushwaha&Seliya, 2006; Khoshgoftaar&Misra, 2003) there is still place for improvements. We believe that using a weak anticipatory model might improve the process.

References

Bagchi, T.P. (2009), *Models for software defects and testing strategies*. ACM SIGSOFT Software Engineering Notes, Volume 34, Issue 2, ACM New York, NY, USA.

Dubois, D.M. (2000), *Review of Incursive, Hyperincursive and Anticipatory Systems - Foundation of Anticipation in Electromagnetism. Computing Anticipatory Systems*: CASYS'99. Edited by Daniel M. Dubois, Published by The American Institute of Physics, AIP Conference Proceedings 517, 2000, pp. 3-30.

Khoshgoftaar T.M., and Seliya N. (2003), *Fault Prediction Modeling for Software Quality Estimation: Comparing Commonly Used Techniques*, Empirical Software Engineering, v.8 n.3, p.255-283.

Kushwaha, D.S., and A.K. Misra (2006), *Cognitive complexity metrics and its impact on software reliability based on cognitive software development model*. ACM

SIGSOFT Software Engineering Notes, Volume 31, Issue 2, ACM New York, NY, USA.

Mao, C. (2009), *Software faults prediction based on grey system theory*. ACM SIGSOFT Software Engineering Notes, Volume 34, Issue 2, ACM New York, NY, USA.

Musa, J.D. (1979), *Software reliability data*. Data and Analysis Center for Software Report, Rome, NY: Rome Air Development Center.

Musa, J.D. (1998), *Software reliability engineering: more reliable software, faster development and testing*, ISBN 0-07-913271-5, McGraw-Hill, New York.