Ann. Inst. Statist. Math.
Vol. 50, No. 4, 789–799 (1998)

A STATE SPACE MODEL FOR SOFTWARE RELIABILITY

YEN-CHANG CHANG AND LII-YUH LEU

Graduate Institute of Statistics, National Central University, Chung-Li, Taiwan, Republic of China

(Received July 1, 1996; revised October 13, 1997)

Abstract. In this paper, we propose a non-Gaussion state space model to apply in software reliability. This model assumes an exponential distribution for the failure time in every test-debugging stage, conditionally on the state parameter—the number of faults in the program. It is a generalized JM model which can be applied to the imperfect debugging situation as well as in evolving programs. By examining a set of data on evolving program failures, the effect of evolving program model is amply proved.

Key words and phrases: JM model, linear exponential loss function, Poisson distribution, exponential distribution.

1. Introduction

The reliability of computer software is a crucial measure about the quality of computer system. During the past thirty years, many software reliability models had been proposed, discussed, modified and generalized. The first well known model for software reliability was JM model which was made by Jelinski and Moranda (1972). By adopting Bayesian method and unifying the earlier model, Langberg and Singpurwalla (1985) generalized JM model. According to their viewpoint, the models made by Littlewood and Verrall (1973, 1974), Geol and Okumoto (1979) can be regarded as gencalized JM models. The detail for the history of software reliability modeling can refer to Musa *et al.* (1987), and Singpurwalla and Wilson (1994).

Singpurwalla and Wilson (1994) reviewed the literature in software reliability and classified these models into three types: models based on concatenated failure rate, models based on non-homogeneous Poisson processes, and models based on a specified relationship between the software's inter-failure time after each cycle of testing and modification. The third type of model, making use of time series, had been proposed. Because of their ability to track failure data, the models give superior predictive abilities to many other models. Examples of the third type models are the model due to Singpurwalla and Soyer (1992), the model due to Chen and Singpurwalla (1994), and the model due to Becker and Camarinopoulos (1990).

Singpurwalla and Soyer (1992) proposed a non-homogeneous autoregressive process, as well known as a Kalman filter, to describe reliability growth. Chen and Singpurwalla (1994) suggested a non-Gaussian Kalman filter to apply to software reliability. The model has a Gamma-Gamma type invariant conditional distribution. Becker and Camarinopoulos (1990) proposed a model that assumes an exponential distribution for the observations, conditionally on the unknown failure rate. A specified class of conjugate prior for the failure rate had been defined to describe the growth of reliability.

Chen and Singpurwalla (1997) unified software reliability models by self-exciting processes. All the models based on concatenated failure rate or non-homogeneous Poisson processes are subsumed by the family of self-exciting processes. The models that we disscussed above are members of the family of self-exciting processes, too. Note that not all models are members of the family of self-exciting processes, for examples, the models by Crow and Singpurwalla (1984) and Sahinoglu (1992).

In this paper we develop a state space model for software reliability, which is generalized from JM model. It assumes an exponential distribution for the failure time in every test-debugging stage, conditionally on the number of faults in program. We can apply this model to an imperfect debugging situation as well as to evolving programs. The reliability of evolving programs had been throughly described in Chapter 15 of Musa *et al.* (1987). In Section 2, we will discuss the reliability function and construct the model, a predictor of failure time can be found by a linear exponential (LINEX) loss function (Zellner (1986)). In Section 3, the basic model and the evolving program model will be examined. By using some techniques such as the prequential likelihood ratio and u-plot (Dawid (1984), Littlewood (1987)), a comparison between these two models is provided.

2. The model

Following are the assumptions of JM model:

(i) the total number of initial faults N_0 is an unknown constant;

(ii) a detected fault is removed immediately and no new fault is introduced;

(iii) the times between failures of the program are independent with an exponential distribution;

(iv) each of the remaining faults contributes an equal amount to the failure rate.

Let T_i, $i \geq 0$ be the times between i-th and $(i+1)$-th failures. Then the density of T_i is

$$g_{T_i}(t_i \mid \phi, N_0) = \phi(N_0 - i)e^{-\phi(N_0-i)t_i},$$

where ϕ is a positive constant as the amount of each fault contributes to the failure rate, namely, the failure rate of every fault.

In our situation, the numbers of residual faults in every test-debugging stage will be regarded as a sequence of random variables with a Poisson distribution. Let N_i be the number of residual faults at i-th test-debugging stage, and ϕ_i be the failure rate of every fault. The model assumptions are shown below:

1. The initial assumption: N_0 has prior distribution $Poisson(\lambda_0)$;

2. The system equation: For fixed n, $n \geq 1$, $T^{n-1} = \{(T_0, T_1, \ldots, T_{n-1}) \mid 0 < T_i < \infty, 0 \leq i \leq n-1\}$, we have

$$(N_n - (N_{n-1} - 1) \mid N_{n-1}, T^{n-1}) \sim Poisson(a_n), \tag{2.1}$$

where a_n is a nonnegative constant, which may depend on T^{n-1} though not on N_{n-1};

3. The observation equation: $(T_n \mid N_n) \sim \text{Exp}(\phi_n N_n)$.

Based on the discussion of Musa *et al.* (1987), a failure caused only one fault. Therefore, under assumption (ii) in JM model, when the scale of the program is stable, a_n in (2.1) is zero, i.e., $N_n = N_{n-1} - 1$ with probability 1 for all $n \geq 1$. The model is the first model in Langberg and Singpurwalla (1985). When the initial faults number N_0 degenerates to a constant, the model will return to JM model. If debugging the program is imperfect, we may introduce a new fault in every debugging stage. Therefore, a reasonable value for a_n is among 0 and 2. The case that more than one fault is introduced can be seen as a rare event. To describe the reliability growth, a reasonable region of a_n is (0,1). Note that, when the scale of the program is unstable (we only consider an evolving program with pure growth), even the assumption (ii) holds, a_n may be not zero.

Results. From assumption 1 ~ 3, we have

(1) $(N_{n-1} - 1 \mid T^{n-1}) \sim Poisson(\lambda_{n-1} e^{-\phi_{n-1} T_{n-1}}), n = 1, 2, \ldots$, where $\lambda_n = a_n + \lambda_{n-1} e^{-\phi_{n-1} T_{n-1}}$;

(2) Conditionally on T^{n-1}, we have $(N_n \mid T^{n-1}) \sim Poisson(\lambda_n)$.

PROOF. For $n = 1$,

$$\begin{aligned} p_{N_0}(n_0) \cdot p_{T_0|N_0}(t_0 \mid n_0) &= \frac{e^{-\lambda_0} \lambda_0^{n_0}}{n_0!} \cdot \phi_0 n_0 e^{-\phi_0 n_0 t_0} \\ &= \frac{\phi_0 e^{-\lambda_0} (\lambda_0 e^{-\phi_0 t_0})^{n_0}}{(n_0 - 1)!} \\ &\propto p_{N_0|T^0}(n_0 \mid t^0). \end{aligned}$$

Then, $(N_0 - 1 \mid T^0)$ has a Poisson distribution with parameter $\lambda_0 e^{-\phi_0 t_0}$. Next, we will check the result (2) when $n - 1$:

$$\begin{aligned} &p_{N_1|N_0,T^0}(n_1 \mid n_0, t^0) \cdot p_{N_0|T^0}(n_0 \mid t^0) \\ &= \frac{e^{-a_1} a_1^{n_1 - (n_0 - 1)}}{(n_1 - (n_0 - 1))!} \cdot \frac{e^{-\lambda_0 \exp(-\phi_0 t_0)} (\lambda_0 e^{-\phi_0 t_0})^{n_0 - 1}}{(n_0 - 1)!} \\ &= \frac{e^{-(a_1 + \lambda_0 \exp(-\phi_0 t_0))} a_1^{n_1 - (n_0 - 1)} (\lambda_0 e^{-\phi_0 t_0})^{n_0 - 1}}{(n_1 - (n_0 - 1))!(n_0 - 1)!} \\ &= \frac{e^{-(a_1 + \lambda_0 \exp(-\phi_0 t_0))} (a_1 + \lambda_0 e^{-\phi_0 t_0})^{n_1}}{n_1!} \cdot \frac{n_1!}{(n_1 - (n_0 - 1))!(n_0 - 1)!} \\ &\quad \cdot \left(\frac{a_1}{a_1 + \lambda_0 \exp(-\phi_0 t_0)} \right)^{n_1 - (n_0 - 1)} \left(\frac{\lambda_0 \exp(-\phi_0 t_0)}{a_1 + \lambda_0 \exp(-\phi_0 t_0)} \right)^{(n_0 - 1)}, \end{aligned}$$

then, take sum for n_0 from 1 to $n_1 + 1$, we get the density of N_1, conditionally on T^0:

$$p_{N_1|T^0}(n_1 \mid t^0) = \frac{e^{-(a_1+\lambda_0 \exp(-\phi_0 t_0))}(a_1 + \lambda_0 e^{-\phi_0 t_0})^{n_1}}{n_1!}.$$

Thus, $(N_1 \mid T^0)$ has a Poisson distribution with parameter $a_1 + \lambda_0 \exp(-\phi_0 t_0) = \lambda_1$, result (2) holds when $n = 1$. Now, by induction, the results (1) and (2) hold for every $n \geq 1$. □

From Bayesian viewpoints, the result (1) is the $(n-1)$-th stage posterior, and the result (2) is the n-th stage prior (see Chen and Singpurwalla (1994)). We can find the density of T_n conditionally on T^{n-1} is

$$f_{T_n}(t_n \mid t^{n-1}) = \phi_n \lambda_n e^{-\lambda_n(1-e^{-\phi_n t_n})-\phi_n t_n}, \quad 0 < t_n < \infty; \tag{2.2}$$

and $P(T_n = \infty \mid t^{n-1}) = e^{-\lambda_n}$ is positive. It is not difficult to get the reliability function in the next stage. Let $Y_n = 1 - e^{-\phi_n T_n}$, from (2.2), Y_n has a density function

$$f_{Y_n}(y_n \mid t^{n-1}) = \lambda_n e^{-\lambda_n y_n}, \quad 0 < y_n < 1,$$

and $P(Y_n = 1 \mid t^{n-1}) = e^{-\lambda_n}$. Note that, $Y_n = 1$ iff $T_n = \infty$. Thus, the reliablity of T_n condionally on T^{n-1} and $T_n < \infty$ is given by

$$\begin{aligned} R_{T_n}(t \mid t^{n-1}, T_n < \infty) &= P(T_n > t \mid t^{n-1}, T_n < \infty) \\ &= P(Y_n > 1 - e^{-\phi_n t} \mid t^{n-1}, Y_n < 1) \\ &= \frac{e^{-\lambda_n(1-e^{-\phi_n t})} - e^{-\lambda_n}}{1 - e^{-\lambda_n}}. \end{aligned} \tag{2.3}$$

Let Y_n^* be Y_n truncated at 1. Thus, Y_n^* has density

$$f_{Y_n^*}(y_n^* \mid t^{n-1}) = \lambda_n e^{-\lambda_n y_n^*}/(1 - e^{-\lambda_n}), \quad 0 < y_n^* < 1.$$

We predict next failure time by a LINEX loss function. If ϕ_n is known, the predictor $\hat{t}_n$ for T_n is the rule of the LINEX loss function

$$l(\hat{t}, t) = e^{\phi_n(\hat{t}-t)} - \phi_n(\hat{t} - t) \quad 1.$$

We have

$$\hat{t}_n = -\log(E(e^{-\phi_n T_n} \mid T^{n-1}, T_n < \infty))/\phi_n = -\log(E(1 - Y_n^* \mid T^{n-1}))/\phi_n.$$

This rule is more conservative than predicted mean $E(T_n \mid t^{n-1}, T_n < \infty)$. When ϕ_n is small, the linear exponential loss is approximate to squared error loss. Then the rule is near to the predicted mean.

Note that the moment generating function of Y_n^* conditionally on T^{n-1} is

$$M_{Y_n^*|t^{n-1}}(s) = \frac{\lambda_n}{1 - e^{-\lambda_n}} \frac{1 - e^{-(\lambda_n - s)}}{\lambda_n - s},$$

for $s < \lambda_n$. Then the expected value of Y_n^* conditionally on T^{n-1} can be found by

$$\frac{d}{ds} \log M_{Y_n^* | t^{n-1}}(s) \mid_{s=0-} \frac{1}{\lambda_n} - \frac{e^{-\lambda_n}}{1 - e^{-\lambda_n}}.$$

Thus, we have

$$\hat{t}_n = -\log\left(1 - \frac{1}{\lambda_n} + \frac{e^{-\lambda_n}}{1 - e^{-\lambda_n}}\right) / \phi_n. \tag{2.4}$$

In practice, ϕ_n is always unknown, ϕ_n can be estimated firstly and then substituting the estimator to (2.4) to find $\hat{t}_n$.

As above, let T_n^* be T_n truncated at ∞. Then T_n^* has density

$$f_{T_n^*}(t_n^* \mid t^{n-1}) - \phi_n \lambda_n e^{-\lambda_n(1-e^{-\phi_n t_n^*}) - \phi_n t_n^*} / (1 \quad e^{-\lambda_n}), \qquad 0 < t_n^* < \infty. \tag{2.5}$$

Chen and Singpurwalla (1997) pointed out that practically all the software reliability models proposed in the literature can be regarded as special cases of self exciting point processes. From (2.2), the coditional probability that T_n is infinity is positive. Therefore, the predictive distributions of the times to next failure is not absolutely continuous. Our model is not a self-exciting point process. In fact, we predict the next failure time by the truncated distribution (2.5). The density function satisfies the uniform boundness condition of Theorem 4.2 in Chen and Singpurwalla (1997). This means that a self-exciting point process can be used to describe the behavior of the truncated variables $\{T_n^*\}$.

3. Application and discussion

3.1 *The basic model*

To avoid complexity, we assume that debugging of program is perfect, and $\phi_n \equiv \phi$, $\forall n$, ϕ is a positive constant. It is the first model in Langberg and Singpurwalla (1985). Suppose λ_0 is known, given $N^n = (N_0, N_1, \ldots, N_n)$, $N_0 T_0$, $N_1 T_1, \ldots, N_n T_n$ have identically independently exponential distributions with failure rate ϕ. In our situation, true values of N's cannot be observed. We may use the posterior mean $E(N_i \mid T^i)$ as an predictor of N_i, for all $i > 0$. Therefore, we will take $\sum_{i=0}^{n} E(N_i \mid T^i) T_i/(n+1)$ as an estimator for ϕ^{-1}. In this model, we have $\lambda_n = \lambda_0 e^{-\phi \sum_{j=0}^{n-1} T_j}$. From result (1) in Section 2,

$$\sum_{i=0}^{n} E(N_i \mid T^i) T_i/(n+1) = \sum_{i=0}^{n} T_i/(n+1) + \sum_{i=0}^{n} \lambda_0 e^{-\phi \sum_{j=0}^{i} T_j} T_i/(n+1).$$

For the function contains ϕ, it is naturally to find the estimate of ϕ by solving the equation:

$$\phi = \left\{ \sum_{i=0}^{n} T_i/(n+1) + \sum_{i=0}^{n} \lambda_0 e^{-\phi \sum_{j=0}^{i} T_j} T_i/(n+1) \right\}^{-1}. \tag{3.1}$$

The right hand side of (3.1) is an increasing concave function of ϕ. If $\phi = 0$, it is positive; if $\phi = \infty$, it is a finite number. So, there is exactly one root in (3.1).

If λ_0 is unknown, but has a Gamma prior distribution with parameters (A, B), we have

$$(\lambda_i \mid T^i) \sim Gamma(i + 1 + A, \beta_i + 1 - e^{-\phi T_i})$$

as the posterior of λ_0 $(= \lambda_i e^{\phi \sum_{j=0}^{i-1} T_j})$ in i-th stage; and

$$(\lambda_i \mid T^{i-1}) \sim Gamma(i \mid A, \beta_i)$$

as the prior of λ_0 in i-th stage, where $\beta_i = (1 - e^{-\phi T_{i-1}} + \beta_{i-1})(e^{\phi T_{i-1}}) = \cdots = (1 + B)e^{\phi \sum_{j=0}^{i-1} T_j} - 1$, $\beta_0 = B$. Then, there is a similar method for estimating ϕ. For every $0 \le i \le n$,

$$\begin{aligned} E(N_i \mid T^i)T_i &= T_i + E(\lambda_0 e^{-\phi \sum_{j=0}^{i} T_j} \mid T^i)T_i \\ &= T_i + E(\lambda_{i+1} \mid T^i)T_i \\ &= T_i + \frac{(i + 1 + A)T_i}{(1 + B)e^{\phi \sum_{j=0}^{i} T_j} - 1}. \end{aligned}$$

As above, we can estimate it by solving the equation:

$$\phi = \frac{1}{\sum_{i=0}^{n} T_i/(n+1) + \sum_{i=0}^{n} \dfrac{(i + 1 + A)T_i}{(n+1)((1 + B)e^{\phi \sum_{i=0}^{n} T_i} - 1)}}, \tag{3.2}$$

it does not difficult to find that there is exactly one root in (3.2).

3.2 *The model for evolving programs*

We need specify the values of $\{a_n\}$ to construct an evolving program model. Three conditions based on the discussion of Musa *et al.* ((1987), Chapter 6) are considered:

(1) The program evolves sequentially, i.e., at any time, there is only one path of evolution of the program.

(2) There is pure growth. The system changes by adding, not removing.

(3) The number of faults introduced by program change is proportional to the number of developed instructions.

The developed instructions are the instructions that were designed and newly written or modified for the program. The condition (3) is from an important assumption that the number of inherent faults is linearly related to program size. The validity of this assumption is supported by Basili and Hutchens (1983), and Takahashi and Kamayachi (1985). To simplify our inference, we assume that the condition (ii) in Section 2 holds. Let I_n be the total number of deliverable executable object instructions at n-th test-debugging stage, i.e., the time interval of n-th and $(n+1)$-th failures; and ΔI_n be the total number of deliverable developed executable source instructions at n-th stage. Note that $\{I_n\}$ and $\{\Delta I_n\}$ are both sequences of cumulated values. As discussed in Musa *et al.* (1987), we will ignore

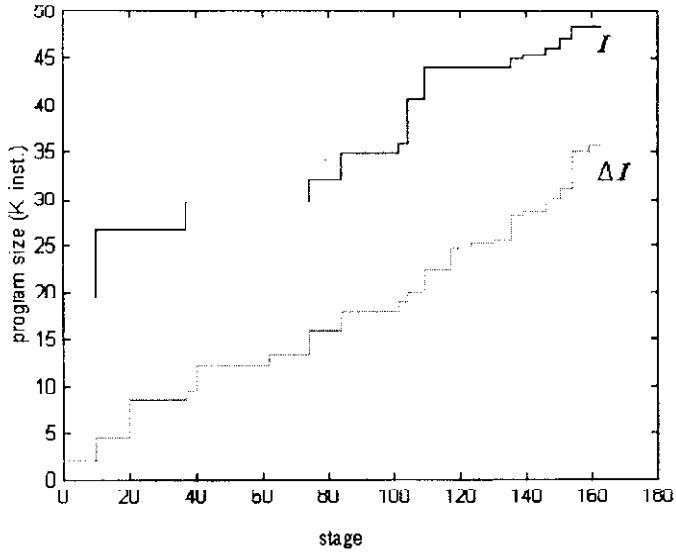

Fig. 1. History of program growth on software project.

minor changes in I's and ΔI's and adjust parameters only for the major changes. Note that, the failure intensity contributed by each fault, ϕ_n, can be seen as a decreasing function of I_n. From condition (3), the total number of faults in n-th stage is proportional to the number of deliverable executable source instructions in n-th stage. Thus, as an approximation, if the program has no branches or loops, we can suppose that ϕ_n is a decreasing linear function of I_n, i.e., let $\phi_n = \phi_0 I_0 / I_n$.

Now, suppose λ_0 is known. The first major change is at the beginning of (M_1)-th stage. This means that we observed M_1 faults before the first major changed stage. Thus, the expected number of unobserved faults can be estimated by the posterior mean $E(N_{M_1-1} - 1 \mid T^{M_1-1}) = \lambda_0 e^{-\sum_{j=0}^{M_1-1} \phi_j T_j}$, and then the expected total number of faults before (M_1)-th stage can be estimated by $M_1 + \lambda_0 e^{-\sum_{j=0}^{M_1-1} \phi_j T_j}$. The number of developed instructions at the beginning of (M_1)-th stage is $\Delta I_{M_1} - \Delta I_{M_1-1}$. By the assumption that the number of faults is uniform distributed in the program, the expected number of faults in the developed code will be estimated by $(M_1 + \lambda_0 e^{-\sum_{j=0}^{M_1-1} \phi_j T_j})(\Delta I_{M_1} - \Delta I_{M_1-1})/\Delta I_{M_1-1}$. Then, the expected number of faults at the beginning of stage M_1 is:

$$(M_1 + \lambda_0 e^{-\sum_{j=0}^{M_1-1} \phi_j T_j})(\Delta I_{M_1} - \Delta I_{M_1-1})/\Delta I_{M_1-1} + \lambda_0 e^{-\sum_{j=0}^{M_1-1} \phi_j T_j}.$$

Similarly, the second major change is at the beginning of (M_2)-th stage, we found M_2 faults, and there are ΔI_{M_2-1} $(= \Delta I_{M_1})$ deliverable developed executable source instructions before (M_2)-th stage. The expected number of inherent faults can be estimated by $M_2 + \lambda_{M_1} e^{-\sum_{j=M_1}^{M_2-1} \phi_j T_j}$. Then, by the assumption that the number of inherent faults is uniform distributed in the program again, the expected

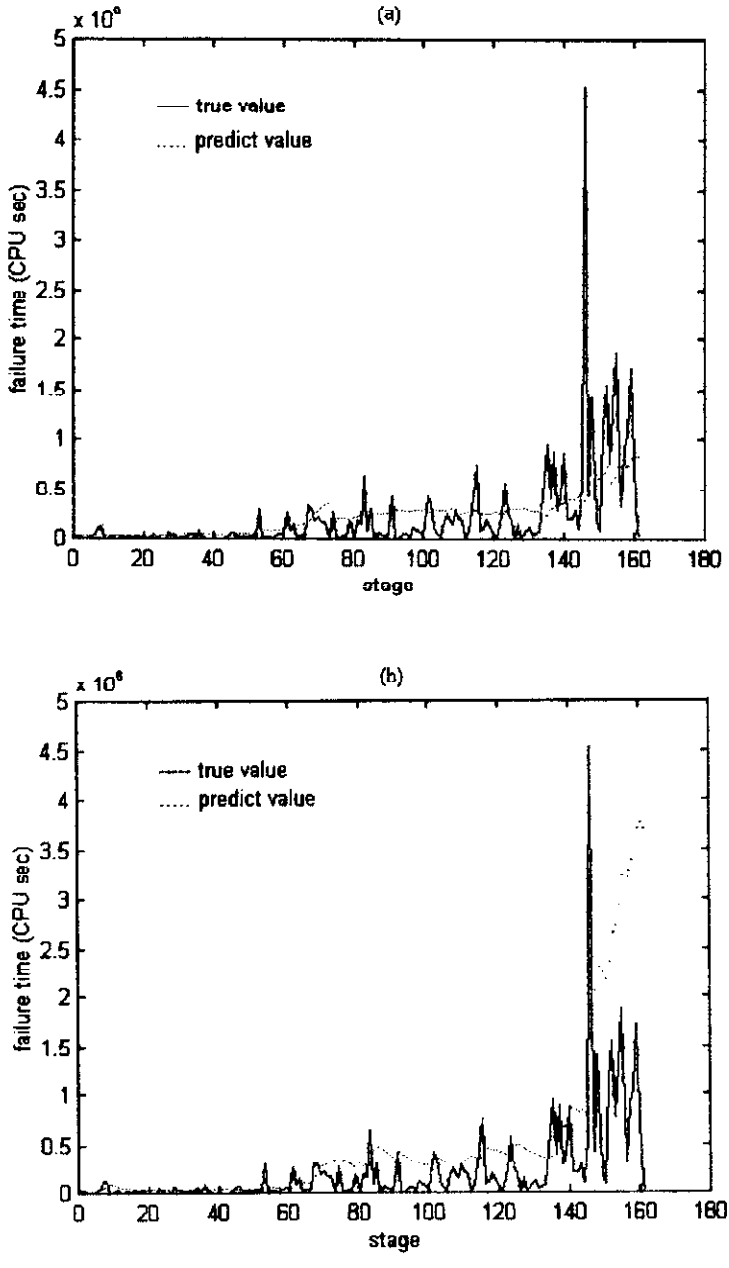

Fig. 2. (a) Predicted value of evolving program model and true failure time; (b) predicted value of basic model and true failure time.

number of faults at the beginning of stage M_2 is:

$$(M_2 + \lambda_{M_1} e^{-\sum_{j=M_1}^{M_2-1} \phi_j T_j})(\Delta I_{M_2} - \Delta I_{M_2-1})/\Delta I_{M_2-1} + \lambda_{M_1} e^{-\sum_{j=M_1}^{M_2-1} \phi_j T_j}.$$

We will make inference about the other stages by this way.

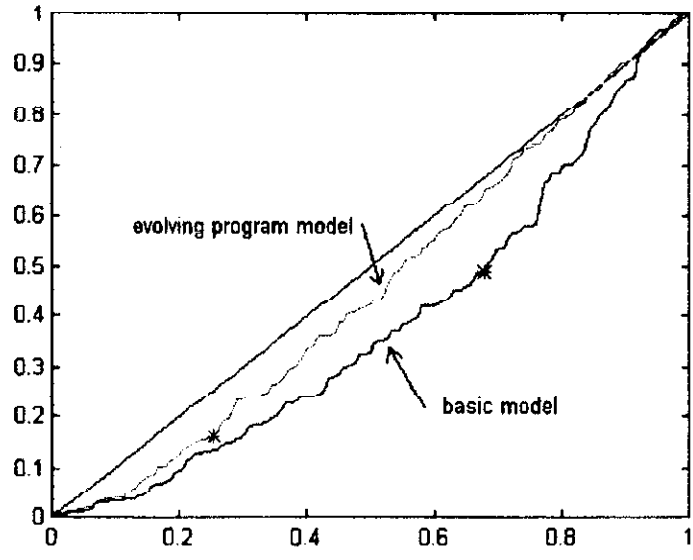

Fig. 3. u-plot. '$*$' shows the position of the maximum vertical deviation.

Based on the discussion above, we have

$$a_{M_i} = (M_i + \lambda_{M_{i-1}} e^{-\sum_{j=M_{i-1}}^{M_i-1} \phi_0 T_j}) \cdot (\Delta I_{M_i} - \Delta I_{M_i-1})/\Delta I_{M_i-1},$$

where M_i is the failure number before i-th major changes, $M_0 = 0$, $i = 1, 2, \ldots$; and $a_j = 0$ otherwise.

If ϕ_0 is unknown, we can estimate ϕ_0 as above, to solve the equation:

$$\phi_0 = \left\{ \sum_{j=0}^{n} (I_0/I_j) T_j/(n+1) + \sum_{j=0}^{n} (I_0/I_j) \lambda_j e^{-\phi_0 (I_0/I_j) T_j}/(n+1) \right\}^{-1}, \tag{3.3}$$

where $\lambda_j = \Delta_j (j + \lambda_{j-1} e^{-\phi_0 (I_0/I_{j-1}) T_{j-1}}) + \lambda_{j-1} e^{-\phi_0 (I_0/I_{j-1}) T_{j-1}}$, $\Delta_j = (\Delta I_j - \Delta I_{j-1})/\Delta I_{j-1}$. Similarly, the right hand side of (3.3) is an increasing concave function of ϕ_0, it has one and only one root.

Example. The models in Subsections 3.1 and 3.2 have been applied to a military software system. The failure data of the system is shown in Musa *et al.* ((1987), 454–457). In Fig. 1, we show the history of program growth during the system test period (also shown in Musa and Iannino (1981)). Note that, this project experienced 20 changes in its testing phase and 163 test failures were experienced in 9994 CPU hr of testing. For basic model assumes that the scale of program is stable, we estimate parameters ϕ and λ_0 by m.l.e. in each stage. Musa *et al.* ((1987), Chapter 5) estimated that mean inherent fault density remaining at the beginning of system test is 6.01 per thousand source lines. We may adopt this value as our prior information. From the beginning of the test, the number

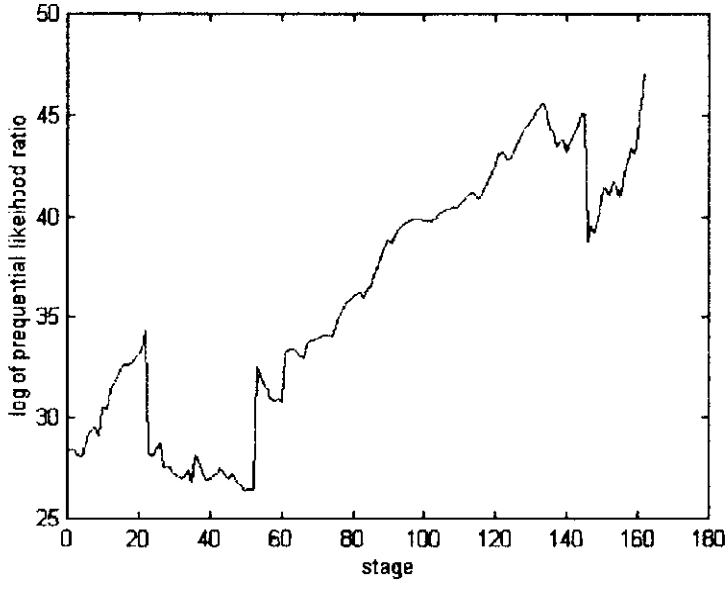

Fig. 4. Plot of the logarithm of the prequential likelihood ratios of evolving program model vs. basic model.

of developed executable source instructions is near two thousand, the initial value $\lambda_0 = 12$ is given for the evolving program model.

The predicted values of both models and the true failure time are shown in Fig. 2. The sum of the absolute error of evolving program model is 29,574,713, and the basic model is 58,620,297. Thus, evolving program model seems more stable. In this example, $\hat{\phi}$ and $\hat{\phi}_0$ are small. So, the predicted values $\hat{t}$'s are near to predicted means based on the estimates. We compare two models by u-plot and prequential likelihood ratio. Figure 3 shows the u-plot, the maximum vertical deviation are drawn on the figure. These are 0.196 (basic model) and 0.086 (evolving program model). Figure 4 shows the prequential likelihood ratio for evolving program model vs. basic model. Though there is a valley between 21st and 53rd stages, an increasing trend is in evidence. The improvement of prediction can be seen as the effect of the evolving program model.

4 Conclusions

We have presented a state space model to predict software failure time, which is a generalized JM model. Two important indexes for quality of software have been reached: the reliability function and the probability that the program is perfect. In practice, the system equation can be modified to satisfy the different situation. Even though the main assumptions and prior information are based on the discussion and observation of Musa *et al.* (1987), this paper has achieved the following goals: a comparison between basic and evolving program model is made in Section 3; the effect of evolving model is in evidence; and most importantly, an

evolving program model is well constructed.

REFERENCES

Basili, V. R. and Hutchens, D. H. (1983). An empirical study of a syntactic complex family, *IEEE Trans. Software Engrg.*, SE-**9**(6), 664–672.

Becker, G. and Camarinopoulos, L. (1990). A Bayesian estimation method for the failure rate of a possibility correct program, *IEEE Trans. Software Engrg.*, SE-**16**, 1307–1310.

Chen, Y. and Singpurwalla, N. D. (1994). A non-Gaussian Kalman filter model for tracking software reliability, *Statistica Sinica*, **4**, 535–548.

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

Crow, L. H. and Singpurwalla, N. D. (1984). An empirically developed Fourier series model for describing software failures, *IEEE Trans. Reliability*, **33**, 176–183.

Dawid, A. P. (1984). Statistical theory: the prequantial approach, *J. Roy. Statist. Soc. Ser. A*, **147**, 278–292.

Goal, A. L. and Okumoto, K. (1979). Time-dependent error-detection rate model for software reliability and other performance measures, *IEEE Trans. Reliability*, R-**28**(3), 206–211.

Jelinski, Z. and Moranda, P. B. (1972). *Software Reiability Reseach. Statistical Computer Performance Evaluation* (ed. W. Freiberger), 465–497, Academic Press, New York.

Langberg, N. and Singpurwalla, N. D. (1985). A unification of some software reliability models, *SIAM J. Sci. Statist. Comput.*, **6**, 781–790.

Littlewood, B. (1987). How good are software reliability predictions?, *Software Reliability—Achievement and Assessment* (ed. B. Littlewood), 154–166, Blackwell Scientific Publications, Oxford.

Littlewood, B. and Verrall, J. L. (1973). A Bayesian reliability growth model for computer software, *J. Roy. Statist. Soc. Ser. C*, **22**(3), 332–346.

Littlewood, B. and Verrall, J. L. (1974). A Bayesian reliability model with a stochastically monotone failure rate, *IEEE Trans. Reliability*, R-**23**(2), 108–114.

Musa, J. D. and Iannino, A. (1981). Software reliability modeling—accounting for program size variation due to integration or design changes, *ACM SIGMETRICS Performance Evaluation Review*, **10**(2), 16–25.

Musa, J. D., Iannino, A. and Okumoto, K. (1987). *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, New York.

Sahinoglu, M. (1992). Compound-Poisson software reliability model, *IEEE Trans. Software Engrg.*, **18**, 624–630.

Singpurwalla, N. D. and Soyer, R. (1992). Non-homogeneous autoregressive processes for tracking (software) reliability growth, and their Bayesian analysis, *J. Roy. Statist. Soc. Ser. B*, **54**, 145–156.

Singpurwalla, N. D. and Wilson, S. P. (1994). Software reliability modeling, *Internat. Statist. Rev.*, **62**, 289–317.

Takahashi, N. and Kamayachi, Y. (1985). An empirical study of a model for program error prediction, *Proceedings Eighth International Conference on Software Engineering*, London, 330–336.

Zellner, A. (1986). Bayesian estimation and prediction using asymmetric loss functions, *J. Amer. Statist. Assoc.*, **81**, 446–451.