M. Xie, G. Y. Hong and C. Wohlin, "Software Reliability Prediction Incorporating Information from a Similar Project", Journal of Software and Systems, Vol. 49, No. 1, pp. 43-48, 1999.

SOFTWARE RELIABILITY PREDICTION INCORPORATING INFORMATION FROM A SIMILAR PROJECT

M. Xie[a], G.Y. Hong[b] and C.Wohlin[c]
[a]Dept of Industrial and Systems Engineering, National University of Singapore, Singapore
[b]Corporate Research & Technology Centre, Motorola (S) Pte Ltd, Singapore
[c]Dept of Communication Systems, Lund University, Lund, Sweden

Summary

Although there are many models for the prediction of software reliability using the failure data collected during testing, the estimation is usually inaccurate, especially at the early stages of the testing phase, and hence many practitioners are hesitant to use software reliability models. On the other hand, the traditional software reliability growth models do not make use of information from earlier or similar projects. For example, software systems today are usually an improvement or modification of an earlier version or at least within the same application domain, which implies that some information should be available from similar projects. In this paper we study some approaches for the estimation of software reliability by incorporating information from a similar project. In particular, we use the Goel-Okumoto model and assume the same value of the fault detection rate. The other parameter is then estimated based on the available testing data. For an actual set of data, our approach provides much more stable estimates and when the traditional maximum likelihood estimates exist and are reasonable, our results are very close to that from a statistical point of view. In addition, our approach does not require numerical algorithm to update the estimate and hence it is convenient to use.

Key Words: Software reliability prediction, early estimation, use of information, the Goel-Okumoto model, maximum likelihood estimation, confidence interval.

1. Introduction

The most practical methods to estimate the reliability of software is by using a software reliability model during the testing phase, see e.g., Musa et al. (1987) and Lyu (1996). However, in order to obtain accurate software reliability estimates, it requires a large number of failure data which are not usually available until the system has been tested for a long time. In many cases, the statistical estimates might not exist at all and this has caused a lot of problems in practice. Many software developers would be more interested in estimating the software reliability as early as possible for their planning purpose and they are hesitant to use software reliability models.

On the other hand, the traditional software reliability models only make use of failure information for the particular software we are interested in, that is the particular version of the software system for which the reliability estimate is to be provided. Within an organization, there ought to be some information from the development of similar systems in projects which can be regarded as similar, and this information can be used to provide a way to obtain better prediction of the reliability. For example, nowadays most large software systems are developed in such a way that it is a modification of an earlier version or at least the new software system is developed in the same environment. Although two systems are not the same, some information should be useful for reliability prediction if they have been developed in a similar way. In this particular case, a similar project can be exemplified with that the two systems under study have been developed using the same development process (including the same design and programming languages), the test methods and environment are the same, and the application domain is the same for both systems.

In this paper, we study some methods for early or more accurate software reliability prediction by making use of the information from a similar project from which historical data are available. We assume that the second system is being developed, and we have some information from development of a previous system which can be used to estimate some model parameters directly. The approach used here is not a traditional Bayesian approach (Littlewood and Verrall, 1973) which when used require much interpretation.

In our presentation, we integrate a case study using an actual set of data. To highlight the idea and approach, we apply the Goel-Okumoto model (Goel and Okumoto, 1979) as an example, although any other software reliability growth model (Xie, 1991) can be used. The focus is on the description of the approach that can be used to make use of information from earlier projects and the discussion of practical issues involved.

The Goel-Okumoto model contains two parameters: one is the number of initial faults and the other is the fault detection rate in testing. We could relate the number of initial faults to some software metrics such as the number of lines of code, but it is known that metrics do not give a good estimate of fault count. In this case, we assume that the fault detection rate is the same for the testing of both systems, which is reasonable when there is a stable process, for example, when the tests are conducted in the same environment, using the same test methods and tools. When the information from the previous system development is used, it will be shown that the estimation is both more stable and easy to carry out. Furthermore, the confidence interval also contains the estimate without assuming the availability of any earlier information. Hence this is a promising method for the estimation of software reliability when information in terms of historical data from similar projects area available.

This paper is organized as follows. Section 2 presents the complete testing information for the first system (denoted System 1). The Goel-Okumoto model which is the mostly commonly used software reliability model, see e.g., Yamada and Osaki (1985), Bondi and Simonetti (1995) and Nara et al. (1995), is used and the parameters are estimated. In Section 3, we discuss some of the issues related to the estimation of the parameters for the second system (denoted System 2) developed using the information from System 1. Particularly, we look at the estimate of the number of initial faults assuming the fault detection rate is the same for both systems. This is compared to the case when assuming that no information from previous projects exists, which is discussed in Section 4. Our estimates are close to the traditional maximum likelihood estimates when the latter exist and are reasonable. Furthermore, our estimates are much more stable and easy to be updated.

2. The G-O model applied to System 1

Software systems developed by a company are often in the same application domain and the same methods and tools are used, and it is also common that a new system is a modification or improvement of a previous version. The testing information for earlier projects is usually available and it is a waste not making use of it. In this section, we use the Goel-Okumoto model for the data from System 1. The information is then used in our subsequent study of the estimation of the reliability for System 2. We present the maximum likelihood estimation in details here as it will be further discussed later on.

For System 1, which was tested for 50 weeks before the release, the failure information is available and is given in Table 1.

Table 1. Number of failures per week for System 1.

Week	Failures	Week	Failures	Week	Failures	Week	Failures	Week	Failures
1	2	11	17	21	1	31	0	41	0
2	11	12	31	22	1	32	0	42	0
3	18	13	8	23	1	33	0	43	1
4	10	14	7	24	0	34	0	44	1
5	12	15	10	25	1	35	1	45	0
6	4	16	2	26	1	36	0	46	0
7	28	17	2	27	0	37	1	47	1
8	6	18	0	28	0	38	0	48	0
9	7	19	3	29	0	39	0	49	0
10	6	20	2	30	1	40	0	50	1

The Goel-Okumoto model is a simple nonhomogenuous Poisson process (NHPP) model (Yamada and Osaki, 1985) with the following mean value function

$$\mu(t) = a(1 - e^{-bt}) \qquad (1)$$

In using this model, the parameter *a* is interpreted as the number of initial faults in the software and the parameter *b* is the fault detection rate which is related to the reliability growth rate in the testing process. The corresponding failure intensity function is given by

$$\lambda\ (t) = abe^{-bt} \qquad (2)$$

The parameters in the Goel-Okumoto model can be estimated using the maximum likelihood method based on the number of failures per interval. Suppose that an observation interval $(0, t_k]$ is divided into a set of subintervals $(0, t_1]$, $(t_1, t_2], ..., (t_{k-1}, t_k]$, the number of failures per subinterval is recorded as

$n_i\,(i=1,2,\ldots,k)$ with respect to the number of failures in $(t_{i-1},t_i]$. The likelihood function is

$$L(n_{1,}\ldots n_k) = \prod_{i=1}^{k} \frac{\{\mu(t_i)-\mu(t_{i-1})\}^{n_i}}{n_i\,!}\exp\{-[\mu(t_i)-\mu(t_{i-1})]\} \quad (3)$$

By taking the natural logarithm of both sides of Eq. 3, we have

$$\begin{aligned}\ln L &= \sum_{i=1}^{k}\ln\{\frac{[\mu(t_i)-\mu(t_{i-1})]^{n_i}}{n_i!}\exp[-(\mu(t_i)-\mu(t_{i-1}))]\}\\ &= \sum_{i=1}^{k}\{n_i\ln[t(x_i)-\mu(t_{i-1})]-[\mu(t_i)-\mu(t_{i-1})]-\ln n_i\,!\}\end{aligned} \qquad (4)$$

For Goel-Okumoto model, the derivative of the logarithm of the maximum likelihood function with respective to the parameters *a* and *b* can be calculated and we have

$$\begin{cases}\dfrac{\partial \ln L}{\partial a} = \sum\limits_{i=1}^{k}\{\dfrac{n_i}{a}+e^{-bt_i}-e^{-bt_{i-1}}\}=0\\[2ex] \dfrac{\partial \ln L}{\partial b} = \sum\limits_{i=1}^{k}(\dfrac{n_i}{e^{-bt_{i-1}}-e^{-bt_i}}-a)(t_ie^{-bt_i}-t_{i-1}e^{-bt_{i-1}})=0\end{cases} \qquad (5)$$

Solving Eq. 5, we get

$$\begin{cases} a = \dfrac{\sum\limits_{i=1}^{k} n_i}{1-e^{-bt_k}}\\[3ex] \sum\limits_{i=1}^{k}(\dfrac{n_i}{e^{-bt_{i-1}}-e^{-bt_i}}-\dfrac{\sum\limits_{i=1}^{k} n_i}{1-e^{-bt_k}})(t_ie^{-bt_i}-t_{i-1}e^{-bt_{i-1}})=0\end{cases} \qquad (6)$$

Since Eq. 6 is nonlinear, we cannot find an analytic solution and it must solved numerically. For the failure data in Table I, the parameter estimates are

$$\begin{aligned} a &= 199.48\\ b &= 0.098076\end{aligned}$$

It should be noted that because of the need for large amount of failure data, accurate estimates are difficult to obtain and the significant digits included here are for the illustration and further comparison in Section 3-4. As the Goel-Okumoto model contains two parameters, the model can be further used in decision making and reliability prediction when the parameter values are known. However, the information is usually discarded when a new project, no matter how similar it is to the current one, is started. Because of the difficulty in getting accurate estimates, information from similar projects should be incorporated. This issue will be addressed in the following sections for analysis of the reliability for System 2.

3. Early prediction for System 2

After the release of System 1, System 2 was developed and it was tested for 28 weeks. The complete data set is given in Table 2.

Table 2. Number of failures per week for System 2.

Week	Failures	Week	Failures	Week	Failures	Week	Failures
1	3	8	32	15	7	22	3
2	3	9	8	16	0	23	4
3	38	10	8	17	2	24	1
4	19	11	11	18	3	25	2
5	12	12	14	19	2	26	1
6	13	13	7	20	5	27	0
7	26	14	7	21	2	28	1

The traditional approach is to analyze this data set independently. However, since we have the information from System 1, it is reasonable to make use of such information in reliability prediction of similar systems. Here we will discuss ways of utilizing the information from both System 1 and 2 for an early reliability prediction. Still using the Goel-Okumoto model, we need to estimate the two parameters in the model, *a* and *b*. This will be discussed in the following.

3.1. Estimation of the parameter *a*

Eliminating faults from software is an important objective of all software projects. As *a* is related to the number of faults in the software, the study of parameter *a* is an important topic of software reliability investigation. Since *a* is mainly related to the size of the software, we can, for example, assume that the parameter *a* is proportional to the size, such as the LOC, complexity of the software and the knowledge of the programmers.

There has been a lot of studies related to the estimation of the number of faults in the software, although it is commonly agreed that they are not very accurate, e.g., Morgan and Knafl (1996) and Ohlsson et al. (1996) for some recent references on this issue. In any case, if an estimate of the number of faults can be derived using software metrics, it can be used as an estimate of *a*.

3.2. Estimation of parameter *b*

Software reliability estimation could be made much easier if there is a way to obtain an estimate for *b*. From Section 2, we know that the two parameters *a* and *b* of the Goel-Okumoto model can be estimated by solving the likelihood equations , Eq. 5. Parameter *a* is a simple function of parameter *b*, but the equation for solving the parameter *b* is nonlinear and it can only be solved numerically.

The parameter *b* is interpreted as the testing efficiency and it is related to the reliability growth rate in the testing. Hence, if the same well-defined process, and test methods and tools are used, it can be expected that the value of *b* remains the same. If we have earlier similar projects or earlier versions of the software tested in an similar environment, we could probably assume that the value of *b* is stable across software projects.

However, this assertion needs to be tested. A way to do so is to make use of the same *b*, and we can estimate *a* assuming the value of *b* is the same for version 1 and version 2 which is illustrated in the next section. Section 4 gives a more detailed comparison with the case when we do not make this assumption. It is shown that, at least for the data sets we have, it is a very reasonable assumption and the results are very close to the case when such an assumption is not made.

3.3. Estimation of *a* assuming the same *b*

For System 1, we denote the Goel-Okumoto model parameter *b* as b_1. Now, for System 2, the parameter b_2 can be assumed equal to *b*. In this case, the maximum likelihood estimate for parameter a_2 can easily be determined as follows

$$a_2 = \frac{\sum_{i=1}^{k} n_i}{1 - e^{-b_2 t_k}} \tag{7}$$

where

$$b_2 = b_1 = 0.098076$$

Table 3 shows the estimation of parameter a_2 of System 2 using the parameter *b* from System 1.

Table 3. Parameter a_2 estimation assuming $b_2 = b_1$.

Week	a_2 (with $b_2 = b_{1,}$)
15	270.01
16	262.70
17	258.86
18	256.97
19	254.48
20	264.58
21	268.20
22	264.58
23	261.39
24	258.57
25	256.05
26	253.82
27	251.83
28	250.05

A comparison with the case of no prior information will be made in the next section. It can be noted here that when the parameter *b* is estimated, the estimation of *a* is straightforward. This can be done whenever there is failure time data available. When testing continues, the estimate can be updated very easily. As will be discussed in the next section, the traditional approach required the solving of a nonlinear equation for which the convergence is often a problem. Although there seems to be a decreasing trend in this case, we will show that the estimate of *a* is actually more stable compared with the case when we the information from the earlier project is not used.

4. Comparison with the traditional approach

In order to compare the estimate of *a* with the case when assuming no prior information from version 1, we estimate parameter a_2 and b_2 of the Goel-Okumoto model for version 2 for the latest eight weeks. The results are presented in Table 4. Note that "NA" denotes that the estimate does not exist or is unreasonable and this is the case for all estimates prior to 11th week.

Table 4. The estimate of parameter a_2 and b_2 using no prior information.

Week	a_2	b_2
11	NA	NA
12	211.3	0.180
13	NA	NA
14	NA	NA
15	NA	NA
16	269.4	0.0924
17	483.5	0.0335
18	NA	NA
19	NA	NA
20	NA	NA
21	276.8	0.0771
22	269.5	0.0819
23	274.5	0.0833
24	260.2	0.0956
25	257.5	0.0957
26	257.4	0.0922
27	251.3	0.0991
28	249.2	0.0999

Note that when maximum likelihood estimation is used, only towards the end of testing, reasonable estimates can be obtained. In fact, the nonexistence and instability of the estimates are great problems for successful application of software reliability models in practice. This problem has been noted by Knafl (1992), Hossain and Dahiya (1996) and others, but there is no proposed solution to this problem other than we are warned of the possibility of non-existence and advised not to estimate the reliability until we have a sufficient number of failure data. This is another important motivation to the proposed approach.

We now construct a 95% confidence interval for the prediction of parameter *a*. To obtain the confidence limits, we can calculate the asymptotic variance of the maximum likelihood estimator of parameter *a* which is the inverse of the local Fisher information, see e.g., Lawless (1982),

$$Var(\hat{a}) = -1 / \left[\frac{\partial^2 \ln L}{\partial a^2}\right]_{a=\hat{a}} = \hat{a}^2 / \sum_{i=1}^{k} n_i \qquad (8)$$

For a given confidence level α, the two-sided confidence interval for parameter *a* is

$$a_L = \hat{a} - Z_{\alpha/2}\sqrt{Var(\hat{a})} \quad and \quad a_U = \hat{a} + Z_{\alpha/2}\sqrt{Var(\hat{a})}, \qquad (9)$$

where $Z_{\alpha/2}$ is the [100(1+ α)/2]th standard normal percentile.

For the given α=0.05, we have that

$$Z_{\alpha/2} = Z_{0.025} = 1.96$$

and the 95% confidence intervals for parameter a_2 using parameter $b_2 = b_1$ are listed in Table 5 for the last 8 values.

Table 5. The 95% confidence interval for parameter a_2 assuming $b_2 = b_1$.

Week	a (b_2 =b_1)	a_U	a_L	a (raw data)
21	268.20	302.56	233.83	270.14
22	264.58	298.49	230.68	266.51
23	261.39	294.88	227.90	274.45
24	258.57	291.69	225.44	260.22
25	256.05	288.86	223.25	257.51
26	253.82	286.34	221.30	257.39
27	251.83	284.09	219.56	251.28
28	250.05	282.09	218.01	249.22

In Figure 1, the estimation results for parameter a_2 using the assumption that parameter $b_2 = b_1$ with its 95% confidence intervals are drawn together with the estimate of parameter *a*. for the raw data, that is, the estimate of *a* without assuming the parameter is the same as for version 1.

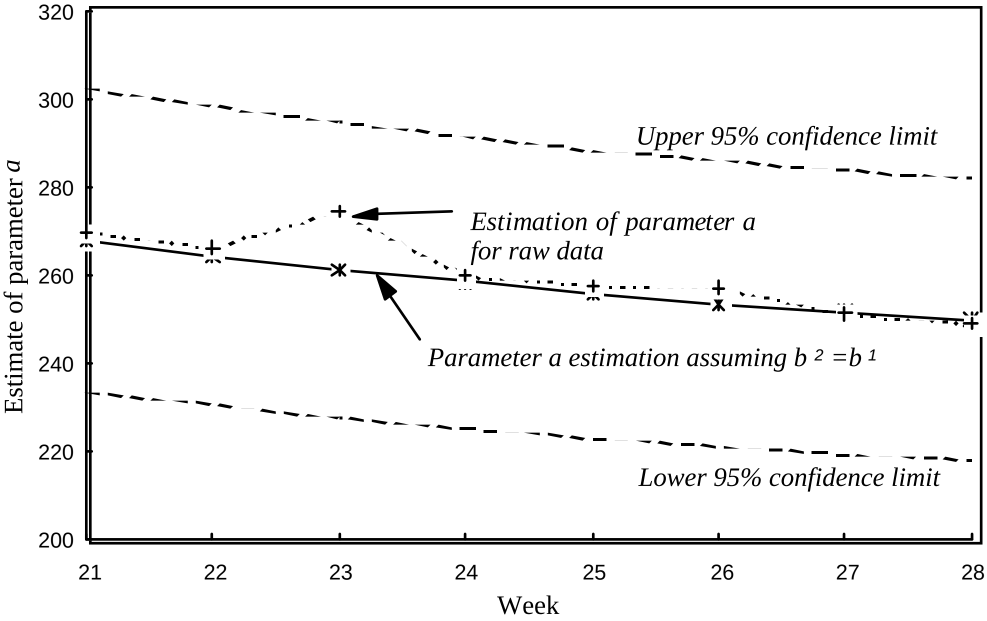

Figure 1. The 95% confidence interval for the estimate of a assuming b_2=b_1 and the comparison with the case when no assumption of earlier information.

From Table 4, we can see that the confidence interval for parameter *a* using the same parameter *b* from System 1 contains the estimated parameter *a* for the case when *b* also is estimated from the current system (System 2). This signifies, in this case, that the method of fixing *b* has the same accuracy as the general method of estimating of both *a* and *b*.

5. Discussion

Recently, due to the quick adaptation to market and fast development, most large software systems are developed in such a way that they are modifications or improvements or earlier systems, or at least a new system is developed using similar methods and tools. It is becoming more and more difficult to have enough failure data for accurate reliability prediction. In this paper, we have studied a method for the estimation of one model parameter by making use of the information from a similar system developed using the same techniques. Although, two systems are never the same, some information from the first system should be useful for the prediction of the reliability of the latter versions. This means that experiences from similar situations must be reused within a company.

The Goel-Okumoto model, which is commonly used in practice, contains two parameters with clear physical interpretations. Our approach looked into ways to provide earlier estimation of some model parameter and the estimation is shown to be clearly simplified and more stable. The stability of the estimates is actually of great concern for many users of software reliability models, see e.g., Zhao and Xie (1993) and Bondi and Simonetti (1995). The numerical results are shown for one actual set of data and it can be seen that the implementation of our procedure is straightforward. It is clear that such an approach is feasible in practice. For practitioners, this approach opens a new way to make use of information from different, but similar projects. When the development process is stable, our approach can be adopted. The main advantages of our approach are clear: the estimates do not fluctuate as much as the traditional approach when new data are available; information from earlier development can be made use of; it is computationally very simple making updating the estimates easy; and the most important of all, the reliability can be estimated at an early stage of testing as the estimates will always exist.

Acknowledgement

Part of this research is supported by the National University of Singapore under research grant RP 3950642. This research is also funded by the Swedish National Board for Industrial Technical Development (NUTEK) and "Software Engineering Research Triangle" initiative (grant: P7816-1).

References

Bondi, P. and Simonetti, G., Evaluating the reliability of the software of a switching system with a multi-variable model. *Software Testing, Verification and Reliability*, **5**(3), 181-202 (1995).

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

Knafl, G.J., Solving maximum likelihood equations for two-parameter software reliability models using grouped data. *Proc. of the 3rd Int. Symp. on Software Reliability Engineering,* North Carolina, Research Triangle Park, IEEE Computer Press, pp.205-213 (1992).

Hossain, S.A. and Dahiya, R.C., Estimating the parameters of a non-homogeneous Poisson process model for software reliability, *IEEE Transactions on Reliability*, **R-42**, 604-612 (1996).

Lawless, J.F., *Lifetime Data Analysis*, Wiley, New York, 1996.

Leung, Y.W., Optimum software release time with a given cost budget. *Journal of Systems and Software*, **17**, 233-242 (1992).

Littlewood, B. and Verrall, J.L., A Bayesian reliability growth model for computer software. *Applied Statistics*, **22**(3), 332-346 (1973).

Lyu, M., *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, 1996.

Morgan, J.A. and Knafl, G.J., Residual fault density prediction using regression methods. *Proc. of the 7th Int. Symp. on Software Reliability Engineering*, New York, IEEE Computer Press, pp.87-92 (1996).

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

Nara, T.; Nakata, M. and Ooishi, A., Software reliability growth analysis - application of NHPP models and its evaluation. *Proc. of the 6th Int. Symp. on Software Reliability Engineering,* Toulouse, France, IEEE Computer Press, pp.250-255 (1995).

Ohlsson, N.; Helander, M. and Wohlin, C., Quality improvement by identification of fault-prone models using software design metrics. *Proc. of the 6th Int. Conf. on Software Quality,* Ottawa, Canada (1996).

Xie, M., *Software Reliability Modelling*, World Scientific Publisher, Singapore, 1991.

Yamada, S. and Osaki, S., Software reliability growth modeling: models and applications. *IEEE Transactions on Software Engineering*, **SE-11**(12), 1431-1437 (1985).

Zhao, M. and Xie, M., Robustness of software release time. *Proc. of the 4th Int. Symp. on Software Reliability Engineering*, Denver, Colorado, USA, IEEE Computer Press, pp.218-225 (1993).

Biography

Min Xie obtained his Licentiate and PhD in Quality Technology from Linkoping University in Sweden in 1986 and 1987 respectively. He graduated from the Royal Institute of Technology in Sweden and received MSc in Engineering in 1984. After working for four years as a research fellow at Linkoping University, he became one of the first recipients of Lee Kuan Yew fellowship, the most prestigious award for researchers in Singapore, tenable at National University of Singapore in 1991. Currently, he is a faculty member at Department of Industrial and Systems Engineering. His research interests include quality engineering, system reliability, software reliability and engineering statistics. Dr Xie has published about 50 journal papers and he is the author of a book "*Software Reliability Modelling*" published by World Scientific in 1991. Dr.Xie is also an editor of *Int. J. of Reliability, Quality, and Safety Eng*. He is a senior member of both IEEE and ASQC.

Guan-Yue Hong graduated from Harbine Institute of Technology, China in 1992. She continued her postgraduate study in Beijing Institute of Aerospace and Aeronautics and received her MSc in 1995. She then joined the National University of Singapore as a research scholar at Dept of Industrial and Systems Engineering. Currently she is with the Corporate Research & Technology Centre of Motorola Singapore Pte Ltd as a software quality engineer. Her areas of research include software reliability modelling, software metrics, software quality and quality engineering. She has authored a number of papers in these areas.

Claes Wohlin received his MSc in Electrical Engineering and PhD in Communication Systems from Lund University, Sweden. He was a Professor in Software Engineering at Dept. of Computer and Information Science, Linkoping University and currently he is a Professor of Software Systems Engineering at Dept. of Communication Systems, Lund University, Sweden. Prior joining the university he worked for a number of years in software engineering, quality engineering and development of large software systems. Dr Wohlin is active in research in the area of software measurement and metrics, software reliability and quality and software engineering in general and he has a number of publications in this areas. Dr Wohlin has been a committee member of a number of conferences.

Dr Robert L. Glass
Computing Trends
1416 Sare Road
Bloomington, IN 47401
USA

Singapore, 20 February 1998

Dear Dr Glass:

I hereby send you the final version of the paper. Two paper copies and a disk including the soft-copy are included. The short biography of each author is at the end of the paper.

The paper is in Microsoft Word 5.1 (which can be read by higher versions) for Macintosh and the file name is XIE98.DOC

Thank you for your time and consideration.

Sincerely yours

M. Xie