Recalibrating Software Reliability Models

SARAH BROCKLEHURST, P. Y. CHAN, BEV LITTLEWOOD, AND JOHN SNELL

Abstract—**In spite of much research effort, there is no universally applicable software reliability growth model which can be trusted to give accurate predictions of reliability in all circumstances. Worse, we are not even in a position to be able to decide *a priori* which of the many models is most suitable in a particular context. Our own recent work has tried to resolve this problem by developing techniques whereby, *for each program*, the accuracy of various models can be analyzed. A user is thus enabled to select that model which is giving the most accurate reliability predictions for the particular program under examination. One of these ways of analyzing predictive accuracy, which we call the *u-plot*, in fact allows a user to estimate the relationship between the predicted reliability and the true reliability. In this paper we show how this can be used to improve reliability predictions in a very general way by a process of *recalibration*. Simulation results show that the technique gives improved reliability predictions in a large proportion of cases. However, a user does not need to trust the efficacy of recalibration, since the new reliability estimates produced by the technique are truly predictive and so their accuracy in a particular application can be judged using the earlier methods. The generality of this approach would therefore suggest that it be applied as a matter of course whenever a software reliability model is used. Indeed, although this work arose from the need to address the poor performance of *software* reliability models, it is likely to have applicability in other areas such as reliability growth modeling for hardware.**

Index Terms—**Prediction accuracy, recalibration, reliability growth model, reliability prediction, software reliability.**

I. Introduction

THE earliest attempts to measure and predict the reliability of software occurred about twenty years ago. In spite of considerable research work in the intervening years, there is still no definitive method or model which can be universally recommended as "best." Perhaps this should not be surprising. Estimating and predicting software reliability is not easy. Perhaps the major difficulty is that we are concerned primarily with design faults.

This situation is very different from that tackled by the conventional hardware reliability theory. Here the dramatic advances of the past quarter century have come from a concentration on the random processes of *physical* failure. Thus, for example, we now have a good understanding of how the reliabilities of complex hardware systems depend upon, on the one hand, the detailed system structure, on the other, the reliabilities of the constituent components. The very success of this physical hardware reliability theory, however, is now revealing the importance of design faults to the overall reliability of complex systems. Our ability to use intelligent strategies to minimize the effects of physical failure of components results in a higher proportion of system failures being caused by flawed designs. Such flaws in hardware systems are very similar to software faults: they represent the result of human misunderstandings. It seems likely, as a result of this, that obtaining good methods for measuring the effect of such flaws on hardware system reliability will be as difficult as measuring software reliability.

Software has no significant physical manifestation. Software failures are merely inherent design faults revealing themselves under appropriate operational circumstances. These faults will have been resident in the software since their creation in the original design or in subsequent changes. We currently do not have good theories of how software faults come into being. Presumably such theories would require better understanding of human problem solving and the social processes involved in writing software; if so, we should perhaps look to social and psychological sciences, rather than physics, for solutions. In view of the comparative lack of success of these sciences in arriving at *quantitative* understanding, it would be wise not to expect any dramatic breakthrough in the short term.

These difficulties notwithstanding, there have been important advances in software reliability modeling recently. In fact, there is now a plethora of models from which the user can choose in order to make reliability estimates and predictions. However, none of these has been shown to be applicable in all circumstances, and we are not presently able to decide in a particular context which would be the most appropriate model to use. This presents difficulties for a potential user, who is solely interested in obtaining reliability measures in which he/she can have confidence.

Our own recent work [1] has attempted to tackle this problem by devising means whereby judgements can be made about the accuracy of *past predictions on a particular data source*. The intention is that a user could apply such techniques, for each data source (program), to the results produced by several models and select the model which has so far performed best by giving the most accurate reliability predictions. It would then be sensible, in the absence of any other information, to use that model for the next prediction *on that data source*. This "horses

Manuscript received June 14, 1989; revised November 10, 1989. Recommended by F. B. Bastani. This work was supported in part by the UK Science and Engineering Research Council and Alvey Directorate under project SE-072 (SB and BL), in part by the CEC ESPRIT programme under project PDCS (SB and BL), in part by ICL plc (PYC), and in part by the National Aeronautics and Space Administration under Grants NAG 172 and NAG 1-771 (BL).

S. Brocklehurst, P. Y. Chan, and B. Littlewood are with the Centre for Software Reliability, City University, Northampton Square, London EC1V 0HB, England.

J. Snell is with the Department of Computer Science, City University, Northampton Square, London EC1V 0HB, England.

IEEE Log Number 8933745.

for courses'' approach obviates the need for *a priori* selection of a model; instead each data source is provided with its ''best'' model. Indeed, this ''best'' model may change as more data is collected.

These new methods of model selection work by analyzing the closeness between predicted and actual failure behavior. In particular, they provide information about two especially important types of departure which we call *bias* (or ill-calibration) and *noise* (or variability). The key idea in the present work is that this knowledge of the nature of past errors of prediction can be used to improve future predictions. The techniques to be described here are quite general and are not model-dependent. They will be shown to be effective in improving predictive accuracy in a high proportion of cases, but users need not take this efficacy on trust: their predictive accuracy in a particular case can be analyzed, just like any other model, using our earlier techniques [1].

II. Reliability Growth and Predictive Accuracy

In its simplest form, the software reliability growth problem concerns the random variables $T_1, T_2, \cdots, T_n$, representing the execution times between successive failures as a program is being debugged. It is generally assumed that attempts are made at each failure to fix the fault which caused that failure. Models vary in the way that they represent this fault-finding and fixing operation: details of different approaches can be found elsewhere [1], [8], [15], [16].

At stage i, when observations $t_1, t_2, \cdots, t_{i-1}$ have been made of the first $i-1$ interfailure times, the objective is to predict future failure behavior represented by the unobserved $T_i, T_{i+1}, \cdots$ random variables. Informally, the prediction problem is solved if we can accurately estimate the joint distribution of any finite subset of $T_i, T_{i+1} \cdots$. This statement, however, begs the question of what we mean by ''accurately,'' and it is this issue which forms a major part of our earlier work [1].

In practice, of course, a user will be satisfied with much less than a complete description of all future uncertainty. In many cases, for example, it will be sufficient to know the current reliability of the software under examination. This could be presented in many different forms: the reliability function, $P(T_i < t)$; the current rate of occurrence of failures (ROCOF) [3]; the mean (or median) time to next failure (mttf). Alternatively, a user may wish to predict when a *target reliability*, perhaps to be used as the criterion for termination of testing, will be achieved.

If we accept that prediction is our goal, it can be seen that the usual discussion of competing software reliability growth *models* is misleading. We should, instead, be comparing the relative merits of *prediction systems*. A prediction system which will allow us to predict the future $(T_i, T_{i+1} \cdots)$ from the past $(t_1, t_2, \cdots t_{i-1})$ comprises:

1) the *probabilistic model* which specifies the distribution of any subset of the T_j's conditional on a (unknown) parameter vector α;

2) a statistical inference procedure for α involving use of available data (realizations of T_j's);

3) a *prediction procedure* combining 1) and 2) to allow us to make probability statements about future T_j's.

Of course, the *model* is an important part of this triad and it seems unlikely that good predictions can be obtained if the model is not ''close to reality.'' However, a good model is not sufficient: stages 2) and 3) are vital components of the prediction system. In fact disaster can strike at any of the three stages.

In principle, it ought to be possible to analyze each of the three stages separately so as to gain trust in (or to mistrust) the predictions. Unfortunately, it is our experience that this is not possible. There are several reasons.

In the first place, the *models* are usually too complicated for a traditional ''goodness-of-fit'' approach to be attempted. Even the simplest exponential order statistic model [15] does not allow this kind of analysis. This should not surprise us: the goodness-of-fit problem for independent *identically* distributed random variables is hard in the presence of unknown parameters. The reliability growth context is much worse because of nonstationarity.

Secondly, statistical properties of the estimators of unknown parameters for a non-Bayesian analysis of these models are usually not available. For example, several models assume, quite reasonably, that the software contains only a finite number of faults and that each of these is fixed with certainty upon its first occurrence. There is thus an upper bound on the number of observable T_j's. This implies that we cannot even trust the usual asymptotic theory for maximum likelihood (ML) estimators. Their small sample properties are usually impossibly hard to obtain (see, for example, the work of Joe and Reid [11] on a particularly simple model).

Of course, there is a proper approach to stages 2) and 3) in the Bayesian framework. It involves posterior distributions of the parameters at stage 2) and Bayesian predictive distributions for 3) (see [2]). Unfortunately, this does present some analytical difficulties for the popular software reliability growth models. However, with recent advances in Bayesian numerical techniques [19], coupled with powerful personal computers, this picture may change in the near future.

Finally, it could be argued that there are models which are ''obviously'' better than others because of the greater plausibility of their underlying assumptions. We find this a dubious proposition. Certainly, the assumptions of some models seem overly naive and it might be reasonable to discount them. However, this still leaves others which cannot be rejected *a priori*. It is our belief that understanding of the processes of software engineering is so imperfect that we cannot even choose an appropriate model when we have an intimate knowledge of the software under study. At some future time it may be possible to match a reliability model to a program via the characteristics of that program, or even of the software development methodology used. This is not currently the case.

Where does this leave a user, who merely wants to ob-

tain trustworthy reliability metrics for his current software project? Our view is that there is no alternative to a direct examination and comparison of the quality of the predictions emanating from different complete prediction systems. In [1] we have described several ways in which this can be done, the most important tools being the *u-plot* and the *prequential likelihood*. The key idea in each case is that a comparison is made between what has been predicted and what is (later) actually observed. We believe that this emulates how a user would informally gain confidence in a sequence of predictions.

A. The u-Plot

For simplicity we shall concentrate on prediction of the next time to failure T_i, based on observations $t_1, t_2, \cdots, t_{i-1}$. The u-plot uses the predictor $\hat{F}_i(t)$, the estimate of the distribution function $F_i(t) = P(T_i \leq t)$, via

$$u_i = \hat{F}_i(t_i) \tag{1}$$

where t_i is the later-observed realization of the random variable T_i. Thus u_i is the probability integral transform of the observation using the predictive distribution function. If the sequence of predictions $\{\hat{F}_i(t_i)\}$ is good, it is easy to see that the sequence $\{u_i\}$ should look like a random sample from a $U(0, 1)$ distribution [1]. There are various types of departure from such an appearance which might show themselves; here we shall only be concerned with whether the $\{u_i\}$ sequence looks *uniformly distributed*. We shall do this via the *u-plot* which is the sample cumulative distribution (cdf) function of the u_i sequence. The departure of this plot from the cdf of $U(0, 1)$, the line of unit slope, is an indication of a departure of the prediction system from accuracy. We can use the Kolmogorov distance, that is the maximum vertical deviation, as a measure of this departure and use standard tables to determine whether or not it is statistically significant.

Fig. 1 shows u-plots for Jelinski–Moranda [10] and Littlewood–Verrall [14] models making predictions on a data set, called S1 [18], analyzed in [1]. These plots are each based on 86 predictions: $\hat{F}_{51}(t)$ through $\hat{F}_{136}(t)$. The Kolmogorov distances are 0.205 (JM) and 0.150 (LV). The first is significant at the 1% level, suggesting very poor prediction from JM; the second is significant at 5%, which suggests that this model is also performing poorly but is somewhat superior to JM.

More importantly for our present purposes, the shape of the plots tells us that JM is making predictions which are too optimistic, while LV predictions are too pessimistic. This can be seen as follows. The JM plot is everywhere above the line of unit slope (the true $U(0, 1)$ cdf), so there are too many small u_i values. But consistently too small u values tells us that the model is underestimating the chance of small times between failure, i.e., the model is too optimistic. A similar argument shows that a plot which is almost everywhere below the line of unit slope, such as LV, is too pessimistic.

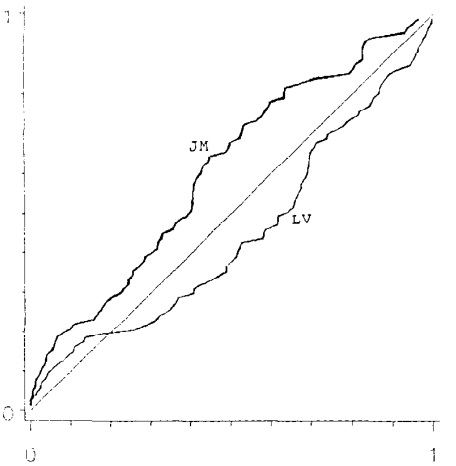


Fig. 1. u-plots for JM and LV model predictions of T_{51} through T_{136}, Musa S1 data [18].

If we knew that these deviations between predicted and actual behavior were consistent, we could attempt to measure the degree of optimism (or pessimism) and improve future predictions by taking account of this tendency. It is this idea which we shall develop in the next section. Before we do that, we shall briefly describe the prequential likelihood function (PL) which is a general mechanism for comparing the accuracy of prediction systems.

B. The Prequential Likelihood Function

The PL is defined as follows. The predictive distribution $\hat{F}_i(t)$ for T_i based on $t_1, t_2, \cdots, t_{i-1}$ will be assumed to have a probability density function (pdf)

$$\hat{f}_i(t) = \hat{F}_i'(t).$$

For predictions of $T_{j+1}, T_{j+2}, \cdots, T_{j+n}$, the *prequential likelihood* is

$$PL_n(j, t) = \prod_{i=j+1}^{j+n} \hat{f}_i(t_i). \tag{2}$$

A comparison of two prediction systems, A and B, over a range of predictions of $T_{j+1}, T_{j+2}, \cdots, T_{j+n}$, can be made via their *prequential likelihood ratio*

$$\mathrm{PLR}_n(j, t) = \frac{\prod_{i=j+1}^{j+n} \hat{f}_i^A(t_i)}{\prod_{i=j+1}^{j+n} \hat{f}_i^B(t_i)}. \tag{3}$$

Notice how, in a fashion analogous to the calculation of the u sequence, the individual contributions to the prequential likelihood are obtained by substitution into the predictor pdf for T_i of the later-observed realization t_i. Dawid [7] shows that if $\mathrm{PLR}_n \to \infty$ as $n \to \infty$, prediction system B is discredited in favour of A. For the finite samples with which we inevitably have to deal, we shall argue that PLR_n increasing consistently suggests the superiority of A over B. In [1] we give intuitive reasons why the PL

works. Specifically we show that consistent bias or noisiness of a prediction system will tend to give a smaller PL than would otherwise be the case.

To summarize, the PLR can be regarded as a general procedure for choosing the best prediction system for a particular data source. The u-plot is a means of indicating a particular kind of consistent inaccuracy of prediction which could be a contributory factor in poor predictive accuracy. Thus a poor u-plot might suggest that poor predictive accuracy (represented by a poor prequential likelihood) is due to consistent bias. For such a case, we shall show in the next section how it is possible to remove the bias and so improve the accuracy of reliability predictions.

III. Recalibration of Predictions

Consider a prediction $\hat{F}_i(t)$ of the random variable T_i, when the true (unknown) distribution is $F_i(t)$. Let the relationship between these be represented by the function G_i where

$$F_i(t) = G_i[\hat{F}_i(t)]. \tag{4}$$

Obviously, if we knew G_i we could recover the true distribution of T_i from the inaccurate predictor, $\hat{F}_i(t_i)$. The key notion in our recalibration approach is that in many cases the sequence $\{G_i\}$ is approximately stationary, i.e., it is only slowly changing in i.

If the sequence were *completely* stationary, i.e., $G_i = G$ for all i, we would have a more precise interpretation of the idea of "consistent bias" used in the previous section. We would also have the possibility of estimating the common G from *past* predictions and using it to improve the accuracy of *future* predictions.

Of course, in practice such complete stationarity is unlikely to be achieved. However, it does seem to be the case that the sequence changes only slowly in many cases. This opens up the possibility of approximating G_i with an estimate G_i^* and so forming a new prediction

$$\hat{F}_i^*(t) = G_i^*[\hat{F}_i(t)]. \tag{5}$$

A suitable estimator for G_i is suggested by the observation that G_i is the distribution function of $U_i = \hat{F}_i(T_i)$. We shall therefore base our estimate G_i^* on the u-plot, *calculated from predictions which have been made prior to* T_i, which is the sample cdf formed from the u_j's for $j < i$. The new prediction (5) recalibrates the raw model output, $\hat{F}_i(t)$, in the light of our knowledge of the accuracy of past predictions for the data source under study. The new procedure is therefore a truly predictive one, "learning" from past errors.

The simplest form for G_i^* is the u-plot with steps joined up to form a polygon (Fig. 2). Later we shall consider a version which is smoothed using a spline technique. The complete procedure for forming a recalibrated prediction for the next time to failure T_i is then:

Stage 1 Check that error in previous predictions is approximately stationary. (See [1] for a plotting technique, the y-plot, which detects nonstationarity, although we shall see later that recalibration often works well even in the presence of nonstationarity.)

Stage 2 Find u-plot for predictions made *before* T_i, i.e., based on $t_1, t_2, \cdots t_{i-1}$, and join up the steps to form a polygon G_i^*.

Stage 3 Use the basic prediction system to make a "raw" prediction $\hat{F}_i(t)$.

Stage 4 Recalibrate the raw prediction using (5).

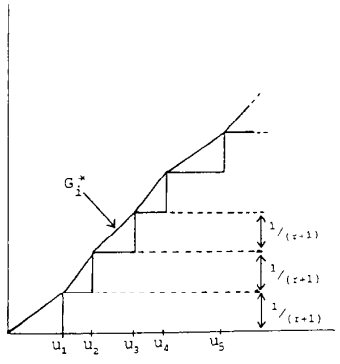


Fig. 2. Method of drawing the joined-up step recalibrating function G_i^*. Here there are r u-points and each step is of size $1/(r+1)$.

This whole procedure can be repeated at each stage so that the functions G_i^* used for recalibration will be based on more information about past errors as i increases. For the simple joined-up u-plot this is not computationally onerous: by far the greatest computational effort is needed for the statistical inference procedures used to obtain the raw model predictions.

It is important to emphasize that the procedure described above does in fact produce a genuine *prediction system* in the sense described earlier: at each stage we are using only past observations to make predictions about the unobserved future failure behavior.

Fig. 3 shows the effect of recalibration on the predictions made in Fig. 1. In the case of the JM model it is known that the raw predictions are too optimistic, and the recalibration makes them less optimistic; in the case of LV, which is initially too pessimistic, the recalibrated version is now less pessimistic. These conclusions are confirmed in the more formal analysis based on the u-plot technique: for JM* the Kolmogorov distance of the u*-plot is 0.119 (compared to 0.205 for the raw predictions), for LV* it is 0.089 (compared to 0.150). Not only are these an improvement in each case, the distances are now no longer statistically significant at the 10% level.

Notice that, although Fig. 3, for simplicity, only shows median predictions, the recalibration is working on the *complete* predictive distribution. Thus it could be expected to improve other reliability estimates, such as the rate of occurrence of failures, in the examples shown here. The recalibration procedure changes the complete shape of the distribution and can therefore correct for far more subtle errors than the mainly simple "optimism" or "pessimism" of these examples.

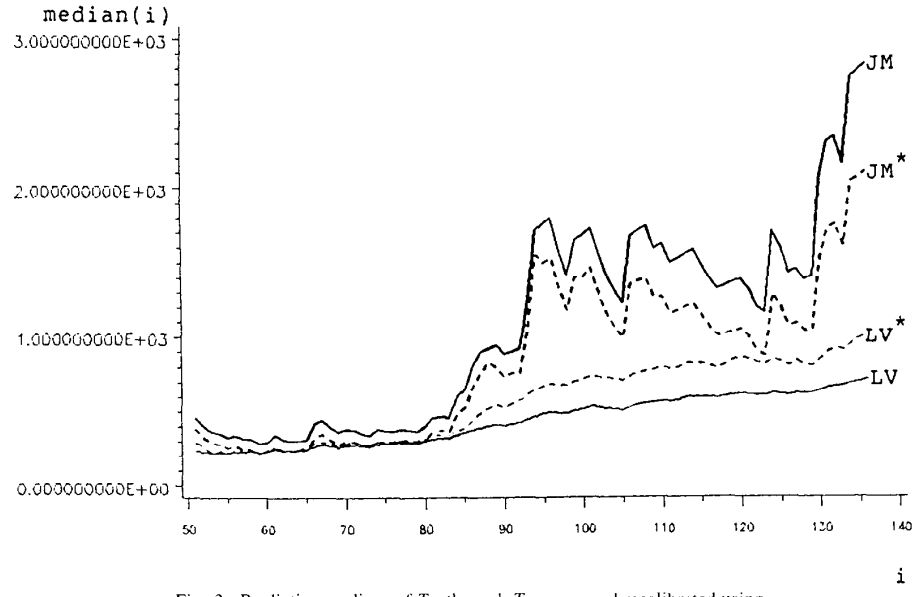


Fig. 3. Predictive medians of T_{51} through T_{136}, raw and recalibrated using joined-up recalibrator G_i^* for Musa System 1 data [18].

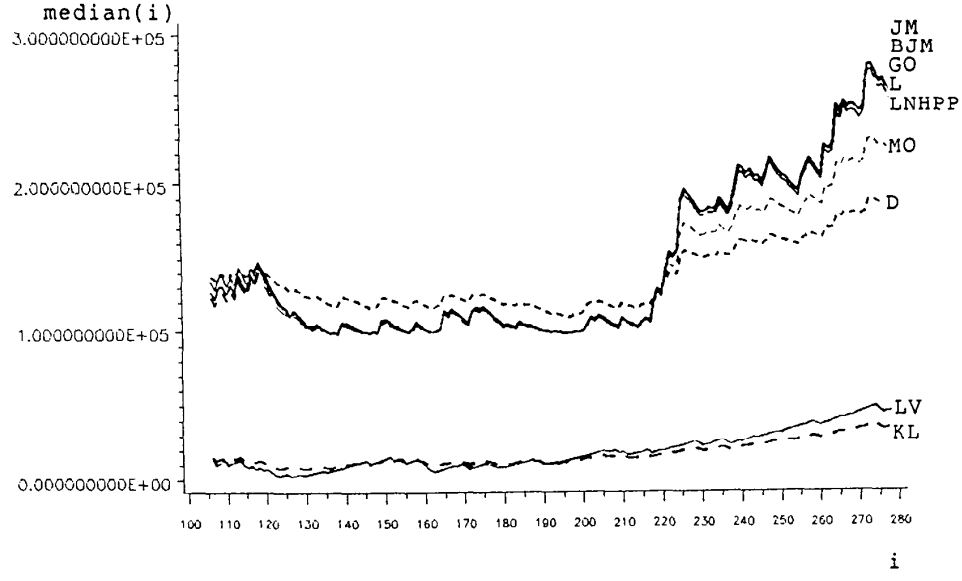


Fig. 4. Raw predictive medians of T_{106} through T_{278}, for all nine models, Musa SS3 data [18].

Fig. 4 shows an analysis of a data set, SS3 from [18], which exhibits startling disagreement between raw predictions from JM and LV models. In fact, in an analysis of this data using nine models [5], it can be seen that seven of them are in close agreement with one another and are close to the JM plot in Fig. 4; the remaining two are close to the LV plot in Fig. 4. A user might conclude that the seven models which give similar answers are closer to the truth than the more isolated pair, but this would be wrong. In fact for this data set *none* is giving acceptable answers. This is shown by the u-plots for JM and LV predictions in Fig. 5. Clearly, the JM predictions are optimistic, and those from LV pessimistic. The effect is a gross one, as can be seen from the Kolmogorov distances, 0.272 (JM) and 0.238 (LV), which are very highly significant (well beyond the 1% level, the highest tabulated). The prequential likelihood shows that LV is superior to JM [1], but neither of them, nor any other model we have used, gives accurate reliability predictions for this data source.

The detailed shape of the u-plots in Fig. 5 is interest-

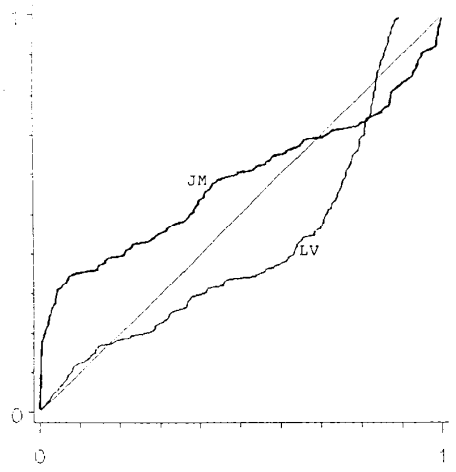


Fig. 5. u-plots for raw predictions of T_{106} through T_{278}, JM and LV models, Musa SS3 data [18].

ing. As was stated above, the most notable feature is the extreme optimism or pessimism. However, this is not a simple effect in either case. For JM the behavior of the plot at each extremity suggests too many very small u values *and* too many very large ones. For LV there seem to be too many fairly large u's and *too few* u's near to 1.0. Thus, although the statements above about optimism and pessimism are correct to a first approximation, a more detailed analysis shows that the u-plots are giving precise information about the incorrect shapes of the *complete predictive distributions*. It can therefore be seen how the recalibration procedure based on such u-plots can effect subtle changes in the complete estimated distribution function for the random variable T_i.

The recalibration technique works dramatically well for this data. Table I shows a comparison between raw model predictions and recalibrated predictions for the following nine models: JM (Jelinski–Moranda [10]), BJM (Bayesian Jelinski Moranda [12]), GO (Goel–Okumoto [9]), MO (Musa–Okumoto [17]), D (Duane [6]), L (Littlewood [13]), LNHPP (Littlewood nonhomogeneous Poisson process [1]), LV (Littlewood–Verrall [14]), and KL (Keiller–Littlewood [1]).

All nine raw u-plots have Kolmogorov distances which are significant well beyond the tabulated 1%. After recalibration, all the distances have been more than halved and *none* are significant at this high level. Fig. 6 shows the dramatic improvement given by recalibration on the JM and LV u-plots in comparison with the raw predictions (see Fig. 5). The differences in the detailed median predictions (only for JM and LV again, for simplicity) can be seen by comparing Figs. 4 and 7. There is much closer agreement between the recalibrated models than between the raw ones.

In both the above examples there is evidence that prediction systems which were in disagreement have been brought into closer agreement by the recalibration technique. Much more important, however, we have objective

TABLE I

KOLMOGOROV DISTANCES FOR u- AND y-PLOTS FOR RAW MODEL AND FOR JOINED-UP RECALIBRATED PREDICTIONS. THE LETTERS INDICATE SIGNIFICANCE LEVELS: E IS SIGNIFICANT AT THE 1% LEVEL, D AT 5%, C AT 10%, B AT 20%, A IS NOT SIGNIFICANT AT 20%. ROUGHLY: A AND B ARE VERY GOOD, C IS ACCEPTABLE, D AND E ARE POOR.

Data set (no. predictions)		JM	BJM	GO	MO	DU	L	LNHPP	LV	KL
S1 (86)	u	.2049E	.1871E	.1773E	.0982A	.1567D	.1123A	.0982A	.1504D	.1457D
	u*	.1188B	.1226B	.1341C	.0499A	.0752A	.0499A	.0499A	.0894A	.0901A
	y	.1156B	.1148B	.1190B	.0795A	.1029A	.0904A	.0793A	.1148B	.1173B
	y*	.1018A	.1016A	.1076A	.0775A	.0808A	.0893A	.0768A	.0901A	.0916A
SS3 (173)	u	.2717E	.2713E	.2705E	.2645E	.2596E	.2717E	.2704E	.2382E	.2372E
	u*	.0982C	.1042D	.0978C	.1057D	.1122D	.0987C	.0997C	.0864B	.1043D
	y	.1273E	.1379E	.1263E	.1435E	.1835E	.1291E	.1300E	.0346A	.0500A
	y*	.0577A	.0664A	.0579A	.0631A	.0968C	.0561A	.0558A	.0415A	.0596A

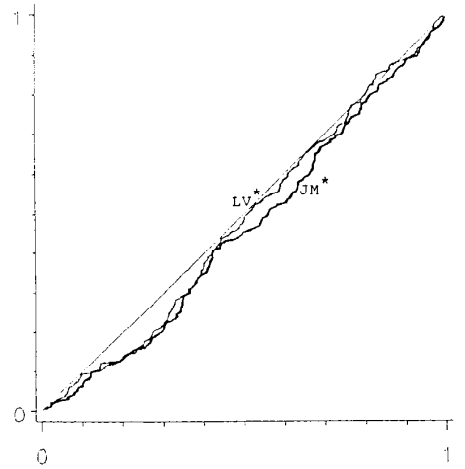


Fig. 6. u*-plots for (joined-up) recalibrated predictions of T_{106} through T_{278}, JM and LV models, Musa SS3 data [18].

evidence from the comparison of u-plot with u*-plot that recalibrated predictions are less "biased" than the raw ones.

These results are encouraging for the efficacy of the recalibration approach, but they are not sufficient grounds for assuming, even in the two examples here, that the recalibrated predictions should be preferred to the raw ones. It may be that the advantage of less bias has been bought at the expense of some other deviation between predicted and actual reliability. We have suggested in the previous section that the prequential likelihood should be used as arbiter between competing prediction systems for any particular data source. It would seem appropriate, therefore, to judge whether a raw or recalibrated prediction system is objectively best by comparing their prequential likelihoods for a series of predictions. Unfortunately this presents problems for recalibrated predictions which are based on the simple polygonal joined-up u-plots suggested above. The reason is somewhat "technical" and is due to the fact that the PL uses the *probability density*

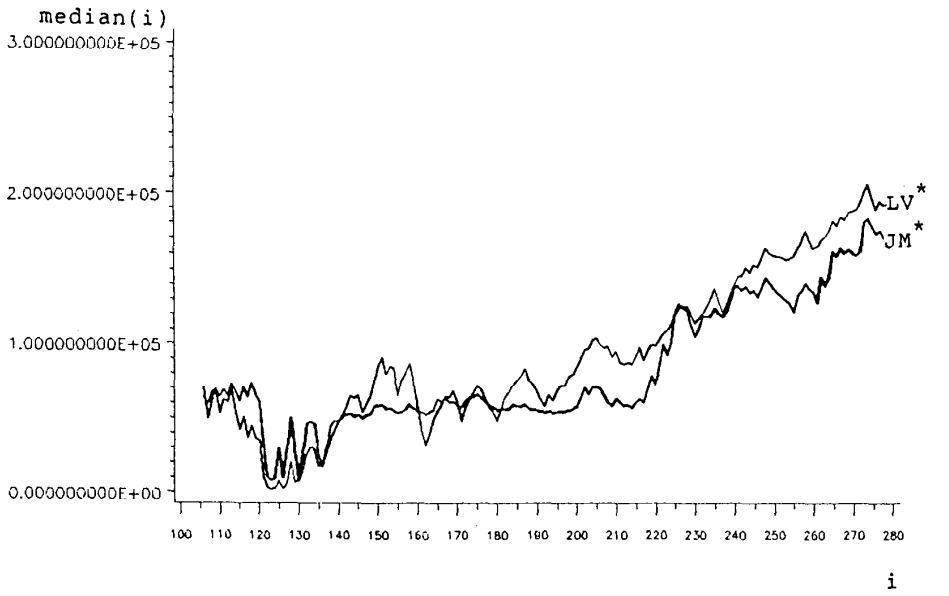


Fig. 7. Medians of (joined-up) recalibrated predictions of T_{106} through T_{278}, JM and LV models, Musa SS3 data [18].

function of the predictive distribution:

$$PL_n^* = \prod_{i=j+1}^{j+n} \hat{f}_i^*(t_i) = \prod_{i=j+1}^{j+n} g_i^*(\hat{F}_i(t_i)).\, \hat{f}_i(t_i)$$

$$= \prod_{i=j+1}^{j+n} g_i^*(u_i).\, \hat{f}_i(t_i) \qquad (6)$$

from (5), letting g_i^* denote the derivative of G_i^*.

Unfortunately, since G_i^* is a polygon, its derivative g_i^* is discontinuous. This means that $\hat{f}_i^*$ is also discontinuous: Figs. 8(a) and (b) show an example of this problem. This discontinuity generally causes PL to report badly on the predictive accuracy of a recalibrated model in competition with the raw version. A user might therefore conclude that recalibration had made the predictions less accurate. We think this can be misleading. It is true that it would be unreasonable to believe that the *true* predictive pdf is grossly discontinuous; the rejection of such a pdf by the PL criterion is therefore strictly correct. However, in practice users are not directly interested in predictive pdf's but in *probabilities*. Such probabilities will be obtained from the pdf by integration, which has the effect of smoothing out the discontinuity. It is therefore perfectly possible for PL to reject a recalibrated prediction system in favor of the raw version, even when the recalibrated (probability) predictions are the most accurate. A rejection in such circumstances is, we believe, unfair: a user needs to know which prediction system is performing best for the kinds of prediction he is likely to make.

There are two ways forward which will be described in the next two sections.

The first approach attempts to decide whether recalibration can be trusted to give improved results in a wide class of circumstances by comparing both recalibrated and

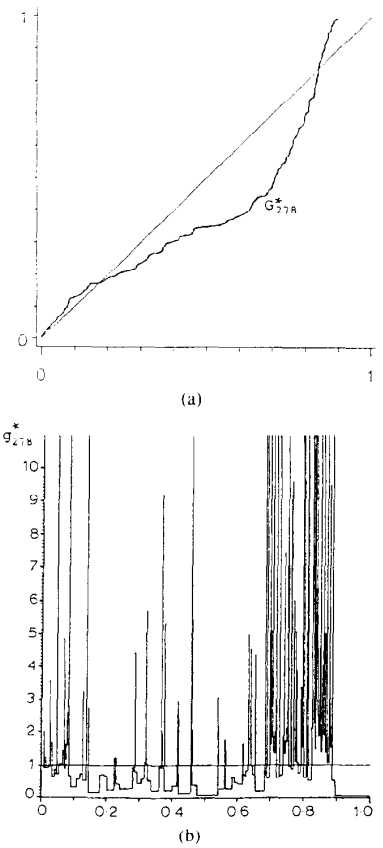


Fig. 8. (a) The joined-up recalibrator G^*_{278} based on a u-plot of 188 points, for the LV model on Musa SS3 data. This is a polygon with 188 vertices, although the resolution of the figure does not allow this to be shown. (b) The derivative of G^*_{278} in (a). This figure shows the dramatic fluctuations in the slopes of successive edges of G^*_{278}.

raw predictions with the *true* reliability when this is known. In practice, of course, such knowledge of the truth is not available so we shall have to use simulated interfailure times. We shall show that in a high proportion of cases the recalibrated prediction system is superior to the raw one. However, as might be expected, this is not always the case.

Our second approach, therefore, applies a smoothing to the polygonal G_i^* in order to give a *continuous* recalibrated predictive pdf. This allows the use of PL as a criterion for judging which prediction system is giving most accurate results. Use of this smoothing is computationally more intensive than use of the simple joined-up u-plot.

A user therefore has a choice: appeal to the general efficacy of the approach as demonstrated by the simulation results based on the simple recalibration technique, or use the smoothed version and use PL to decide whether recalibration is working in the particular example under study.

IV. Simulation Results

The simulation experiment [4] consisted of generating 100 realizations of the interfailure time sequence t_1, t_2, $\cdots$ t_{100} from each of the models JM, L, LV, KL, and D, with constant parameters being used for each model. These data sets were then analysed using the "wrong" models: thus, for example the JM data set was analyzed using the L, LV, KL, and D models.

The model parameters were estimated based on t_1, t_2, $\cdots$ t_{j-1}, to obtain $\hat{F}_j(t)$ for $j = 20, \cdots 101$. Then, for $i = 40, \cdots 101$, the u-plot using $u_j = \hat{F}_j(t_j)$, for $j = 20, \cdots i - 1$, was used to obtain G_i^* and hence $\hat{F}_i^*(t)$. It was thus possible to compare the known true $F_i(t)$ with the raw predictor $\hat{F}_i(t)$ and with the recalibrated predictor $\hat{F}_i^*(t)$ for $i = 40, \cdots, 101$.

In a particular case a user is interested in knowing whether the raw or recalibrated predicted distribution is closer to the true one. There are various ways we could examine the differences between predicted and true distributions. Perhaps the most obvious is a direct measure of the distance between the two functions, such as the Kolmogorov distance. This is defined as follows. For raw predictions let $\hat{d}_i(t) = \hat{F}_i(t) - F_i(t)$ and for recalibrated $\hat{d}_i^*(t) = \hat{F}_i^*(t) - F_i(t)$, both for $i = 40, \cdots 101$. The Kolmogorov distances are $\hat{k}_i = \sup_{t>0} |\hat{d}_i(t)| = |\hat{d}_i(\tau)|$ and $\hat{k}_i^* = \sup_{t>0} |\hat{d}_i^*(t)| = |\hat{d}_i^*(\gamma)|$. A simpler procedure is to merely check whether the recalibrated or raw *median* is closer to the true one.

The first analysis concerns only predictions of T_{101}; there are 2000 such predictions in the experiment. If we consider those predictions of T_{101} for which the u-plot (based on predictions prior to T_{101}) was significant at the 5% level, indicating that there was evidence of bias, *89% of the recalibrated predictions were superior to the corresponding raw ones*. This figure rises to 92% if we only recalibrate for u-plots which are significant at the 1% level.

Even when we recalibrated *regardless* of the u-plot evidence, the recalibrated predictions improved on raw ones in 61% of cases. Here there will be many cases where raw predictions are close to the truth; then we would not expect the recalibration to introduce an improvement and the recalibrated and raw predictions should be close to one another. However, since the recalibrated predictive distribution is polygonal ("lumpy"), the Kolmogorov distance (which compares the *maximum* deviations of the two predictions from the truth) will tend to discriminate against the recalibration in favor of the raw prediction. This figure of 61% can therefore be thought of as a conservative one.

Other simple comparisons between recalibrated and raw predictions are fairer in this situation. For example, the recalibrated median is closer than the raw one to the true median in 70% of these cases. This figure rises to 91% when we recalibrate only for u-plots significant at 5%, and 94% when we recalibrate only for u-plots significant at 1%.

These results for T_{101} are supported by the more extensive recalibrations of the predictions of $T_{40}, \cdots, T_{101}$: here recalibrated medians are closer to the true one in 86% of cases when the u-plot at stage 100 was 5% significant, and are closer in 93% of cases when the u-plot is significant at 1%.

In summary, even when we blindly used the recalibration on *all* predictions, there was an improvement in about 7 out of 10 cases. More importantly, when we adopted the more rational and discriminating approach of only using the technique when the u-plot analysis suggested recalibration might be fruitful (by indicating the presence of "bias"), *there was improvement about 9 out of 10 times*.

Of course, we do not know whether our simulated data was typical of real software reliability data. Indeed, since we were generating data according to several models with very different underlying assumptions, some of the data sets are likely to be unrealistic. However, we believe that these results are encouraging for the general power of the approach.

In practice a user might wish to have more than a belief in the general efficacy of the approach: he needs to know that it is working for the particular data source under examination. The obvious approach is to use the methods of analysis of predictive quality [1] discussed earlier. In the next section we show how this can be done.

V. Parametric Spline Smoothing

The u-plot is merely the sample cdf of the observed u's. Thus the problem of estimating the approximately stationary function G_i in (4) is simply the problem of obtaining an estimate of a cdf from a finite random sample. There are several ways in which this can be done so that the estimator is differentiable and so has a smooth pdf. We could, for example, fit an appropriate parametric family of distributions to the data. An example is the family of

Beta(α, β) distributions with pdf

$$f(u) = u^{\alpha-1}(1-u)^{\beta-1}/B(\alpha, \beta) \qquad 0 \le u \le 1. \tag{7}$$

This is a fairly flexible family, but it is not sufficiently wide to represent all the general shapes of u-plots which we have encountered in practice (see [5] for an example). This seems likely to be a problem with other candidate parametric families of distributions. Another less important difficulty is that the evaluation of the cdf is not easy for certain regions of the parameter space.

The need for a method of fitting a very general class of u-plot data suggests the use of parametric splines, which are widely used in computer graphics because of their versatility. We shall use the cumulative chord as the parameter, whereupon the spline is defined as follows. Let $\{x_i, y_i\}$, for $i = 1, 2, \cdots, r$, denote the r points of the u-plot to which we want to fit the spline, and let

$$p_i' = p_{i-1}' + \left[(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2\right]^{1/2} \tag{8}$$

with $p_0' = 0$, $x_0 = 0$, and $y_0 = 0$; i.e., p_i' is the distance from the origin, along the polygon, to the ith point. Here x_i is the ith order statistic of the u's and y_i is the height of the u-plot at x_i. For convenience we shall use the normalized chord

$$p_i = p_i'/p_r' \tag{9}$$

so that both parametric functions will have domain $[0, 1]$.

We now have two sets of data, $\{x_i, p_i\}$ and $\{y_i, p_i\}$, to each of which we fit a three knot least-squares cubic spline; call these $x = x(p)$ and $y = y(p)$. These splines are each constrained so that $x(p)$ and $y(p)$ are strictly increasing functions taking values between 0 and 1 for p in $(0, 1)$, with $x(0) = y(0) = 0$ and $x(1) = y(1) = 1$. It follows that the function defined parametrically as $(x(p), y(p))$ is also strictly increasing between 0 and 1. We call this function the parametric spline and it has the properties of a cdf. More importantly for our needs, it is everywhere differentiable with a smooth derivative. This means that if we use this function to recalibrate software reliability predictions we are certain to obtain a smooth recalibrated predictive density. We can therefore use prequential likelihood as a criterion of predictive accuracy and be confident that we shall not encounter the difficulties we met with the polygonal joined-up u-plot.

Clearly, using this spline is more tedious than recalibrating predictions from the joined-up u-plot; details can be found in [5]. However, run times are generally much less than are required for the original raw predictions. Since these raw predictions must always be computed, the small overhead involved in using the spline is worthwhile. Most importantly this technique allows a user to determine, via prequential analysis, whether the recalibrated predictions are objectively better than the raw ones for a particular data source. It also similarly allows comparisons to be made between different recalibrated prediction systems. Such knowledge about the performance in a particular instance is more valuable than the general assertions of efficacy which come from the earlier simulation exercise.

TABLE II
AS TABLE I BUT FOR *SPLINE-SMOOTHED* RECALIBRATED PREDICTIONS.

Data set (no. predictions)		JM	BJM	GO	MO	DU	L	LNHPP	LV	KL
	u	.2049E	.1871E	.1773E	.0982A	.1567D	.1123A	.0982A	.1504D	.1457D
S1	u**	.1168B	.1197B	.1277B	.0511A	.0794A	.0507A	.0526A	.1027A	.1053A
(86)	y	.1156B	.1148B	.1190B	.0795A	.1029A	.0904A	.0793A	.1148B	.1173B
	y**	.1109A	.1126A	.1102A	.0852A	.0762A	.0715A	.0853A	.0878A	.0916A
	u	.2717E	.2713E	.2705E	.2645E	.2596E	.2717E	.2704E	.2382E	.2372E
SS3	u**	.0820B	.0822B	.0782A	.0901B	.0916B	.0859B	.0846B	.0834B	.1006C
(173)	y	.1273E	.1379E	.1263E	.1435E	.1835E	.1291E	.1300E	.0346A	.0500A
	y**	.0573A	.0693A	.0560A	.0632A	.1016C	.0571A	.0557A	.0352A	.0452A

To distinguish it from the earlier polygonal G^*, we shall denote the spline smoothed recalibrating function by G^{**}. The recalibrated predictions are then

$$\hat{F}_i^{**}(t) = G_i^{**}\left[\hat{F}_i(t)\right]. \tag{10}$$

Table II shows the u-plot and y-plot Kolmogorov distances for the same data sets as those used in Table I. It can be seen that the entries in the two tables are very similar. This is to be expected since the spline recalibrated predictive distribution function is designed to be a *smooth* function close to the joined-up recalibrated predictive distribution. If these two functions are close, the u's based on them will be close and thus so will the plots. In practical terms this means that the predictions of *probabilities* from the two techniques will be very similar, and in particular their medians are very close (compare Fig. 9 to Fig. 7). However, their predictions of *probability densities* will be very different: it is this difference we wish to exploit in the use of the prequential likelihood for the spline version.

In Fig. 10 the evolution of the prequential likelihood ratios is shown for the various recalibrated predictions against raw model predictions. Notice how, for LV, the prequential likelihood seems to be suggesting that the joined-up recalibrated predictions are *worse* than the raw ones. This is a dramatic example of the effect of the discontinuity of joined-up recalibrated probability densities upon the likelihood: it causes a spurious rejection of these recalibrated predictions in favor of those from the raw model. That this is, indeed, spurious can be seen from the behavior of the spline recalibrated predictions: there is overwhelming evidence that the LV**:LV prequential likelihood ratio is increasing rapidly (it has reached more than e^{40} during these predictions!). A user could therefore be very confident that the LV** predictions here are more accurate than the LV ones.

A comparison of JM** and JM is even more dramatic: the PLR reaches e^{90} over the range of predictions shown. This is partly due to the fact that raw JM predictions are significantly less accurate than those of raw LV (although both are bad from u-plot evidence). Thus JM starts off

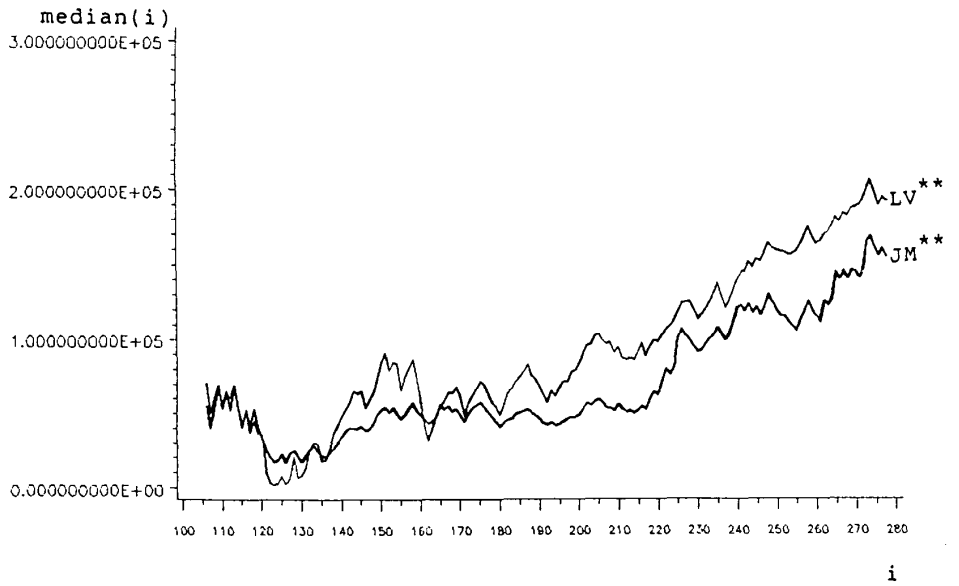


Fig. 9. Medians of spline recalibrated predictions of T_{106} through T_{278}, JM and LV models, Musa SS3 data. Note closeness to results in Fig. 7 for joined-up recalibration.

Fig. 10. Plot of log PLR against i, showing comparison of predictive accuracy between each type of recalibration and the raw predictions: JM and LV models, Musa SS3 data.

with more room for improvement. In fact, after recalibration, the two spline predictors LV** and JM** have comparable accuracy on the prequential likelihood evidence, with slight evidence of superiority for JM**.

Fig. 11 shows an example of recalibrated probability density functions at stage 278 in the SS3 data set. The two raw predictive densities from LV and JM disagree greatly, but after recalibration there is close agreement between LV** and JM**. This is illustrated even more dramatically in Fig. 12 which shows predictive densities for stage 121 in the S1 data. Notice here the curious mode which appears in each predictive density after recalibration. Neither of the raw predictive densities (exponential for JM, Pareto for LV) can have a nonzero mode, which suggests that the "learning" from past errors can give an insight not present in the raw models. What is particularly

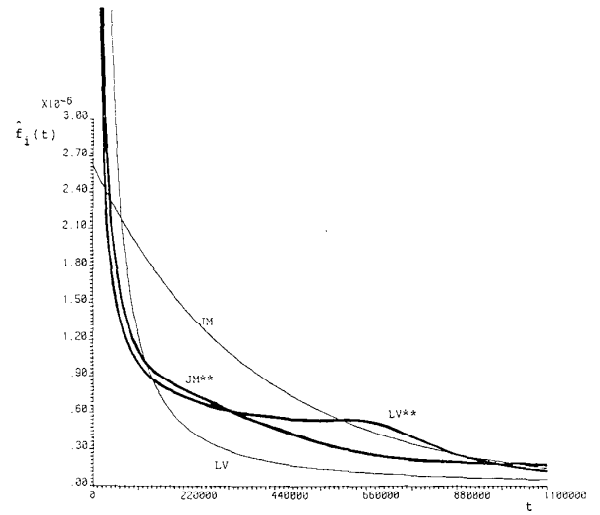


Fig. 11. Examples of individual spline recalibrated predictive probability density functions for T_{278}, using JM and LV models on Musa SS3 data. Note great difference between raw predictions, and closeness of recalibrated.

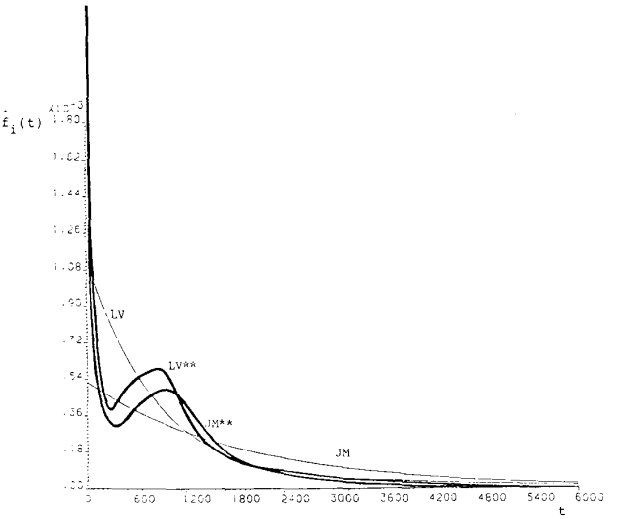


Fig. 12. As in Fig. 11, but for T_{121}, Musa S1 data. Note, again, closeness of the two recalibrated predictions and how these differ greatly from the (very different) raw predictions.

striking, we believe, in figures like this is not only the close agreement of the two predictions after recalibration, but *how dramatically these differ from the raw predictions.*

These figures give some indication of the power of the method to change fundamentally the raw prediction, on the evidence of analysis of past predictive error. Thus the improvements in simple summary statistics shown in the median plots (Figs. 3, 4, 7, 9) are merely the tip of an iceberg: when recalibration works it will do so in very general ways and a user could reasonably expect *all* reliability measures to improve in accuracy.

VI. Retrodictive Recalibration

The recalibration technique described in this paper is based on an analysis of the accuracy of similar predictions at earlier stages in the acquisition of data from testing a program. Thus when we came to recalibrate the prediction of T_{101} it was necessary to make predictions of T_{20}, T_{21}, $\cdots$, T_{101} (each based only on the data observed prior to making the prediction) in order to calculate the G_i^* (or G_i^{**}), $i = 40, \cdots, 101$, which transforms the raw prediction, $\hat{F}_i(t)$. For all the models each such prediction is quite computationally intensive, so a single recalibration can require considerable effort. If recalibration is to take place at each stage as each new interfailure time is observed, then of course this overhead disappears, since it will be necessary to calculate each raw prediction anyway.

However, the problem seemed sufficiently important that we examined a retrodictive recalibration procedure which only needs a single basic calculation (e.g., maximization of a likelihood function) for each recalibration. For those models using maximum likelihood estimation of the parameters this scheme works as follows. To predict T_{101}, we use all available data, $t_1, \cdots, t_{100}$, to calculate an estimate of the model parameters. This is used, of course, to obtain the raw prediction of T_{101}. It is also used to *retrodict* (i.e., "predict" the past) $T_1, T_2, \cdots T_{100}$. Since we have the actual observations of this past, we can compare the retrodictions with these in the same way that we do with genuine predictions. In particular we can form the *retrodictive u-plot* and use this to recalibrate the raw prediction of T_{101}.

Unfortunately, this procedure seems to be useless! The reason is fairly subtle. It seems to be the case that a prediction of T_i, based on $t_1, \cdots, t_{i-1}$, can be in error in different ways from a retrodiction of T_j ($j < i$) also based on $t_1, \cdots, t_{i-1}$. More precisely, the approximate stationarity in the errors of prediction of T_i (based on $t_1, \cdots, t_{i-1}$) as we vary i is very different from the approximate stationarity of errors of "prediction" (really *retrodiction*) of T_j (based on $t_1, \cdots, t_{i-1}$) as we vary j for fixed i. It seems that we can expect to obtain the first kind of approximate stationarity, but not the second: it is, of course, such approximate stationarity which underpins the basic idea of recalibration.

Once again this seems to suggest that in assessing software reliability we must be careful of making unfounded generalizations. Just as we cannot assume that a model performing accurately on one data set necessarily will give good performance on another [1], so we cannot assume that information gained from an analysis of the accuracy of one *type* of prediction will necessarily be trustworthy for another. Although these remarks are based on the evidence of retrodictive error being a poor guide to one-step-ahead prediction, it is likely that the implications are more far reaching. For example, the *predictive* recalibration method for one-step-ahead predictions may not be effective for predictions further ahead. Thus if we wished to recalibrate a raw 20-step-ahead prediction it may be nec-

essary to use a form of the G function which is itself based on a comparison of 20-step-ahead raw predictions with actual (later observed) data. We hope to investigate issues of this kind in future work.

VII. Discussion and Conclusion

We have shown that recalibration can be a powerful technique for improving the accuracy of software reliability growth predictions. The technique is completely general, in the sense that it is not model-dependent. It can be applied to any predictive scheme and will clearly have applications to prediction outside the software context, for example, to *hardware* reliability growth modeling. It can also be used for different types of prediction, but it should be remembered that recalibration should be based on past predictions of the same type.

Our simulation results for the simple joined-up G^* suggest that it offers an improvement in accuracy over the original models in a high proportion of cases. This alone would be sufficient reason for advocating that it be applied as a matter of course to all models: essentially doubling the number of prediction systems available to the user.

As we have demonstrated elsewhere [1], a user cannot select a model *a priori* from this plethora of available models and know that it is the best for the job. Instead, it is necessary to apply all available models to each data source and use the techniques described in [1], principally the prequential likelihood, to select the one which is giving most accurate reliability predictions for the *particular data source* (program) under study.

To make this method of discriminating between reliability prediction systems work for recalibrated models, we have introduced the notion of a spline-smoothed recalibrated prediction. The user is now in a position to apply several models, and their recalibrated versions, to his/her data and *select that which is objectively performing best*. We believe that this eclectic approach should in future be standard practice.

Our results give a new insight into reliability growth modeling. It can now be seen as essentially a two stage process: first capturing the long term trend and then using these new ideas to estimate local behavior. A rich class of new models could be formed from a *distribution-free* fitting of trend, followed by a later analysis of detailed probabilistic structure along the lines described above. We are currently investigating these possibilities: early results are encouraging.

References

[1] A. A. Abdel-Ghaly, P. Y. Chan, and B. Littlewood, "Evaluation of competing software reliability predictions," *IEEE Trans. Software Eng.*, vol. SE-12, pp. 950–967, 1986.

[2] J. Aitchison and I. R. Dunsmore, *Statistical Prediction Analysis*. Cambridge, England: Cambridge University Press, 1975.

[3] H. Ascher and H. Feingold, *Repairable Systems Reliability* (*Lecture Notes in Statistics*, no. 7). New York: Dekker, 1984.

[4] S. Brocklehurst, "On the effectiveness of adaptive software reliability modelling," Centre for Software Reliability, City Univ., London, Tech. Rep., Oct. 1987.

[5] P. Y. Chan, "Software reliability prediction," Ph.D. dissertation, City Univ., London, 1986.

[6] L. H. Crow, "Confidence interval procedures for reliability growth analysis," U.S. Army Material Systems Analysis Activity, Aberdeen, MD, Tech. Rep. 197, 1977.

[7] A. P. Dawid, "The well-calibrated Bayesian" (with discussion), *J. Amer. Statist. Assoc.*, vol. 77, pp. 605–613, 1982.

[8] A. L. Goel and F. B. Bastani, *IEEE Trans. Software Eng.* (*Special Issue on Software Reliability*), vol. SE-11, no. 12, 1985 and vol. SE-12, no. 1, 1986.

[9] A. L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE Trans. Rel.*, vol. R-28, pp. 206–211, 1979.

[10] Z. Jelinski and P. B. Moranda, "Software reliability research," in *Statistical Computer Performance Evaluation*, W. Freiberger, Ed. New York: Academic, 1972, pp. 465–484.

[11] H. Joe and N. Reid, "Estimating the number of faults in a system," *J. Amer. Statist. Assoc.*, vol. 80, pp. 222–226, 1985.

[12] B. Littlewood and A. Sofer, "A Bayesian modification to the Jelinski–Moranda software reliability model," *Software Eng. J.*, vol. 2, pp. 30–41, 1987.

[13] B. Littlewood, "Stochastic reliability growth: a model for fault-removal in computer programs and hardware designs," *IEEE Trans. Rel.*, vol. R-30, no. 4, pp. 313–320, Oct. 1981.

[14] B. Littlewood and J. L. Verrall, "A Bayesian reliability growth model for computer software," *J. Roy. Statist. Soc., C* (*Applied Statistics*), vol. 22, pp. 332–346, 1973.

[15] D. R. Miller, "Exponential order statistic models of software reliability growth," George Washington, Univ., Washington, DC, Tech. Rep. T-496/84, 1984.

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

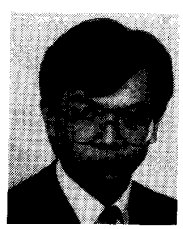
[17] J. D. Musa and K. Okumoto, "A logarithmic Poisson execution time model for software reliability measurement," in *Proc. 7th Int. Conf. Software Engineering*, IEEE Computer Society, New York, 1984, pp. 230–238.

[18] J. D. Musa, "Software reliability data," Data Analysis Centre for Software, Rome Air Development Center, Rome, NY, Rep., 1979.

[19] A. F. M. Smith, A. M. Skene, J. E. H. Shaw, and J. C. Naylor, "Progress with numerical and graphical methods for practical Bayesian statistics," *Statistician*, vol. 36, pp. 75–82, 1987.

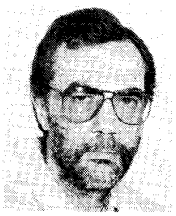
Sarah Brocklehurst received the B.Sc. degree in mathematics from the City University, London, England, in 1986.

She has since worked as a Research Assistant at the Centre for Software Reliability at the City University. Her main research interest is in software reliability modeling and prediction.



P. Y. Chan received the B.Sc. degree in actuarial science and the Ph.D. degree in computer science and statistics from the City University, London, England.

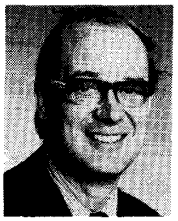
He worked as an Actuarial Trainee in industry before joining the Centre for Software Reliability at the City University as a Research Assistant where his main research interest was in software reliability modeling and prediction. He spent one year in software development work in Canada and is currently working at the Swiss Reinsurance Company in Switzerland.



Bev Littlewood received the B.Sc. degree in mathematics from Imperial College, London, the M.Sc. degree in statistics from the University of London, and the Ph.D. degree in statistics and computer science from City University.

He is currently Professor of Software Engineering at City University and Director of the Centre for Software Reliability. Since 1980 he has had an association with the advanced digital electronics programme at NASA Langley Research Centre. His research interests are in stochastic modeling, with particular reference to software reliability.

Dr. Littlewood is a Fellow of the Royal Statistical Society, a member of the IEEE Computer Society, and a member of IFIP Working Group 10.4 on Reliable Computing and Fault Tolerance. He is co-chair of the Programme Committee for the 20th International Symposium on Fault Tolerant Computing (FTCS-20) to be held in 1990.



John Snell received the B.Sc. degree in mathematics from London University, London, England.

He spent the next three years in the Royal Air Force and then worked in the aircraft industry for six years as an aerodynamicist. He became a Lecturer in Mathematics in 1960 and, following the award of the London-University postgraduate diploma in computer science, joined the Department of Computer Science of the City University in 1967. His interests are in the application of computers in curve and surface fitting, the numerical solution of differential equations, and the application of graph theory to reliability analysis.