Enhancing Accuracy of Software Reliability Prediction *

Naixin Li   Yashwant K. Malaiya

Computer Science Department
Colorado State University
Fort Collins, CO 80523
(303) 491-7031
malaiya@cs.colostate.edu

Abstract

The measurement and prediction of software reliability require the use of the Software Reliability Growth Models (SRGMs). The predictive quality can be measured by the average end-point projection error [9]. In this paper, the effects of two orthogonal classes of approaches to improve prediction capability of a SRM have been examined using a large number of data sets. The first approach is preprocessing of data to filter out short term noise. The second is to overcome the bias inherent in the model. The results show that proper application of these two approaches can be more important than the selection of the model.

1 Introduction

In order to achieve high reliability at an acceptable cost, developers need to be able to estimate the reliability of software under development, and, for management and planning purposes, they should be able to project the additional effort needed for their software to reach a certain reliability level. The reliability of software can be estimated statically or dynamically. The static approach is based on the software size, complexity and other static parameters and can be used even before the software being tested. It can however provide only preliminary estimates. The dynamic approach is based on the software failure data, which is collected during the system test phase. It can be used when the software has been tested for a while and some failure data have been collected. This approach, however, has the advantage of being able to predict the general trend of the software reliability improvement during the test phase.

*This work was partly supported by SDIO/IST and monitored by ONR

Quite a few SRGMs have been proposed for the dynamic approach [1, 9, 17, 20]. The steps involved in the process of using a model to make prediction about a system's reliability include: collecting software failure data during the testing phase and/or operational use; preprocessing the failure data to filter out noise; selecting the best SRGM for the data (project); applying the failure data to drive the model; applying the fitted model to make claims about the reliability level of the software and/or to make projection about the additional effort needed to achieve a desired reliability level. The accuracy of projection can be affected by each activity during the process, so appropriate decisions must be taken at each step.

First we consider the problem of the noise inherent in software failure data. Different inputs applied can have significantly different defect detection capabilities. Since the order in which inputs are applied is never truely random, it is possible for a few "weak" or a few "strong" inputs to be applied close to each other causing a relatively low or high failure intensity for a short duration. Idealy, a preprocessing step should filter out the short-term variations as noise while preserving the longer term trend. As we would expect some smoothing generally improves predictability but excessive smoothing makes it worse. Malaiya *et al* [10] tried to smooth the noise by data grouping. By grouping a few adjacent failure data points, they noticed that the predictability of SRGMs improves initially as the group size increases, and then gets worse as larger group sizes are involved. For the four data sets used, they found that there was an optimal grouping size, given by the total number of defects divided by 20, or equivalently the optimal number of groups is about 20.

In this study, we did more detailed experiments with a much larger number of data sets. We also tried some other methods to smooth the noise, including windowing and data dependent grouping. Currently we have

0-8186-4010-3/93 $03.00 © 1993 IEEE

more than forty data sets from various sources, which cover a wide range of software, including system software, realtime control, data base applications, military applications, and students' projects. These span a wide range of software sizes and defect densities. Since grouping is not appropriate if only a small number of data points are available, we used 21 software failure data sets with 73 to over 800 data points. Table 1 is a summary of the data sets used in our experiments.

Table 1: Software Failure Data Used

Data	Ref.	Code Size	#Bugs	Project Type
SS1A	[2]	$100,000^+$	112	O.S.
SS1B	[2]	$100,000^+$	375	O.S.
SS1C	[2]	$100,000^+$	277	O.S.
SS2	[2]	$100,000^+$	192	Time Sharing
SS3	[2]	$100,000^+$	278	Word Proc.
SS4	[2]	$100,000^+$	196	O.S.
T1	[2]	21,700	136	Realtime
T2	[1]	N/A	86	Realtime
T3	[2]	N/A	207	N/A
T5	[2]	2,445,000	831	Realtime
T6	[2]	5,700	73	Commercial
T18	[1]	N/A	163	Military
T40	[2, 19]	180,000	101	Military
Proj1	[22]	14,000	132	Space System
Proj3	[22]	N/A	210	N/A
Proj4	[22]	N/A	196	N/A
YT1	[20]	200,000	111	Realtime
YT3	[21]	870,000	535	Realtime
HP2	[14]	N/A	74	N/A
TSW	[5]	N/A	129	N/A
Usbar	[5]	N/A	397	N/A

The second concern is the bias that all major models have [4, 11]. While some specific models have been shown to have better fit or predictability, none of them describes the fault detection process exactly. In addition, the characteristics of individual data sets may vary because of different testing practices. One possible way to overcome this is to emphasize the most recent data points by using a weighted [12] parameter estimation. The other approach is to adaptively adjust the projections. The results suggest that using such recalibration improves the accuracy.

Finally we have also examined combinations of preprocessing and recalibration. The results show that if near-optimal choices are made, the accuracy can be greatly enhanced.

Section 2 describes the experiments performed. Section 3 presents a summary of the results obtained using real data sets, and the observations are discussed.

2 The Experiments

2.1 The SRGMs

Table 2: Software Reliability Growth Models

Models	Mean Value Function	Failure Intensity
Exponential	$\beta_0[1-\exp(-\beta_1 t)]$	$\beta_0\beta_1 e^{-\beta_1 t}$
Logarithmic	$\beta_0 \ln(1+\beta_1 t)$	$\frac{\beta_0\beta_1}{1+\beta_1 t}$
Delayed S	$\beta_0[1-(1+\beta_1)e^{-\beta_1 t}]$	$\beta_0\beta_1^2 t e^{-\beta_1 t}$
Power	$\beta_0 t^{\beta_1}$	$\beta_0\beta_1 t^{\beta_1 - 1}$

Four of the major SRGMs are considered here: Exponential model, logarithmic model, delayed S-shaped model and power model. Table 2 is a summary of these models. All of these are two parameter models, and thus can be compared [11].

The past software failure data is generally in the form <defect number, failure time>, all the data sets used contain the points: $<\mu_1, t_1>$, $<\mu_2, t_2>$, $<\mu_3, t_3>$, ... $<\mu_n, t_n>$. where μ_i is the number of defects detected by time t_i We can use $\frac{\mu_{i+1}-\mu_i}{t_{i+1}-t_i}$ as an estimation of the actual failure intensity λ_i at time t_i. With this, we can fit the SRGMs and evaluate the model parameters β_0 and β_1. For each data set, we repeated this with subsets of the data set containing 2, 3, ..., n data points. In each case, using a SRGM, we predict the number of defects p_i to be observed at the time of last failure data point t_n and compare the predicted number p_i with the actual number of defects found by that time μ_n. This allows us to obtain the relative error, $\frac{|p_i-\mu_n|}{\mu_n}$, corresponding to that fitted model. By averaging the relative error over time points from t_2 to t_n, we get the average error (AE) as introduced in [9, 11]. We will use AE as a measure to evaluate the predictive accuracy.

2.2 Data Preprocessing

As mentioned earlier, the noise in the software failure data affects the predictability of SRGMs. The main objective here is to find some guideline that will allow us to select the smoothing method and the degree of smoothing to improve the predictability of SRGMs. Five different smoothing schemes were considered.

Grouping with fixed group size By grouping a fixed number of data points into one point, we expected that the noise values may compensate each other for that period and thus the noise inherent in the failure data reduced siginificantly. This is carried out by selecting data points $<\mu_1, t_1>$, $<\mu_{1+g}, t_{1+g}>$, $<\mu_{1+2g}, t_{1+2g}>$, $<\mu_{1+3g}, t_{1+3g}>$, ..., where g is the group size, which is the number of defects grouped for most of data sets we experimented.

This scheme was experimentally examined in [10], where the fitness of the projected number of defects p_n against μ_n at time t_n was used as a measure to evaluate the goodness of different group sizes. Here we use the variable-step predictability measure, AE [9], as the measure which is a better predictability measure.

```
PROCEDURE lump_grouping();
  i:INTEGER;          /* loop variable */
  data_size:INTEGER;/* #data points before lumping*/
  data_cnt:INTEGER; /* #data points after lumping */
  rising:BOOLEAN;   /* rising failure intensity */
  rfail, rftime, lemda: ARRAY [1..N] of REAL;
                    /* failure data before lumping*/
  fail, ftime: ARRAY [1..N] of REAL;
                    /* failure data after lumping */
BEGIN
  data_cnt := 0;
  rising := TRUE;
  FOR i := 1 TO data_size DO
  BEGIN
     IF lemda[i] < lemda[i+1] THEN
     BEGIN
        IF rising THEN continue ELSE
        BEGIN
           fail[data_cnt] := rfail[i];
           ftime[data_cnt]:= rftime[i];
           data_cnt := data_cnt + 1;
           rising := TRUE;
        END
     END ELSE rising := FALSE;
  END;
  fail[data_cnt] := rfail[data_size];
              /* always keep the last data point */
  ftime[data_cnt]:= rftime[data_size];
END;
```

Figure 1: Procedure for lump grouping

Grouping failure intensity lumps When testing proceeds by focusing on one kind of defects at a time, or by applying one testing technique during a stage, the failure intensity would typically increase initially and then decrease until the next stage. When one finds a special type of fault, it is possible that he will soon find other faults of the same kind, and then as he proceeds further, since fewer such faults exist, the failure intensity decreases. Thus each lump in failure intensity could be associated with the transition from one testing stage to another during the testing process. By grouping failure intensity lumps in the data, we hope to minimize the noise associated with transition points during testing.

The data points are selected according to neighbouring failure intensities. For example, if $\lambda_i < \lambda_{i+1} < ... < \lambda_j$, and $\lambda_j > \lambda_{j+1} > ... > \lambda_k$ then all the data points between t_i and t_k, excluding $< \mu_i, t_i >$ and $< \mu_k, t_k >$, would be dropped due to grouping for model fitting. The procedure in Figure 1 describes lump grouping.

Windowing Instead of grouping a few data points into one, in this method all the data points are kept, but the failure intensity is approximated differently at each point. λ_i is estimated as $\frac{\mu_{i+w}-\mu_i}{t_{i+w}-t_i}$ where w is an adjustable factor called window width. Clearly the failure intensity calculated in this way will be smoother than the earlier approaches.

2.3 Weighted least square estimation

Normal least square parameter estimation approach gives equal weight to each data point when a model is fitted. This results in a fitted model that best fits the data. When reliability projection or reliability estimation is our major concern, we expect that if we get a better fit to the later data point, we will get better predictions for the future points in time. If during the process of fitting a model, we give more weights to the recent data points, we would expect a better predictive accuracy. It can be done as described below.

Suppose $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ are the data points and $Y = a + bX$ is a model to be fitted to this data set. Then the actual value of y_i at each data point can be written as

$$y_i = a + b \times x_i + \epsilon_i$$

where ϵ_i is the error related to the model at point x_i.

With weighted least square, the weighted sum of squares of errors as given by:

$$\sum_{i=1}^{n} \epsilon_i^2 \times \beta_i$$

where β_i is the weight associated with data point x_i. For normal least square approach, $\beta_i = 1$ for all $i \in [1..n]$.

The weight β_i can be chosen in numerous ways. We experimented with weights as a linear function of time t_i, as a linear function of cumulative number of failures μ_i, and as a linear function of data points i. The last one was found to be better than the other two schemes, which is described by:

$$\beta_i = c + e \times i$$

Let the total weights be the same as unweighted scheme, i.e.:

$$\sum_{i=1}^{n} c + e \times i = \sum_{i=1}^{n} 1 = n$$

then e is determined once c is chosen. The parameter c controls the weighting scheme. Setting $c = 1$ results in normal least square; setting $c = 0$ results in biggest differences in weights. In our experiments we adjusted the value of c from 1 to 0 in step of -0.1 for each data set.

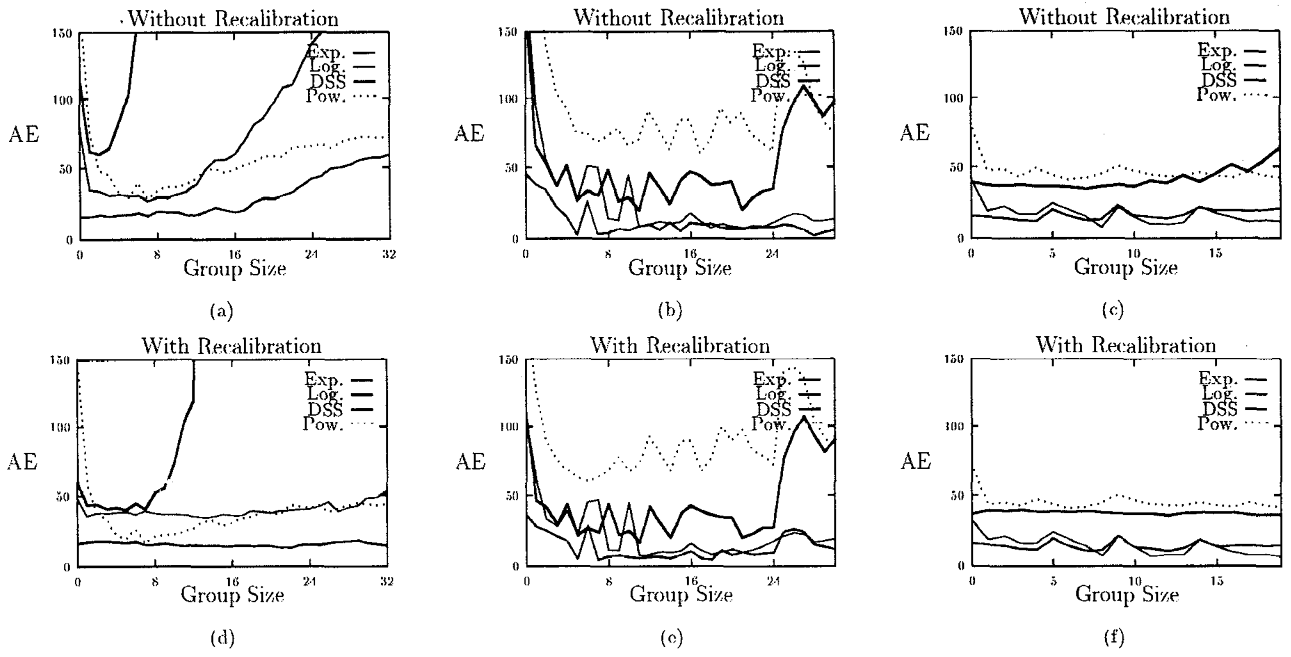

Figure 2: Representative plots for fixed-size grouping

2.4 Adaptive Approach

It was noticed [4, 6, 11, 23] that some models have a tendency to overestimate or underestimate the reliability, which make them good candidates for the adaptive approach (also called *recalibration* [4]). Once a model is fitted, it can be used to predict the reliability of software at some future time. However there is no way to tell exactly how close a prediction will be to the actual value. We can, however, measure the bias of the estimation of the software reliability at each *past* point of time. These biases can then be used to adjust the projection made using the model. We tried to adjust the prediction by using linear regression to project the trend of bias with respect to time t_i or the cumulative number of failures μ_i. We also tried to simply take the average of bias values during the past and deducting this average bias from the model projection. This later approach was found to be superior to the other two more complicated methods.

To observe the effects of the adaptive approach, we repeat each experiment with and without the adjustment. This also allows us to see the effect of combining this approach with the earlier noise smoothing techniques. For each data set in Table 1, we experimented with all the methods mentioned in this section. The result are discussed in next section.

3 Results and Observations

The results of the experiments are shown in various plots. Since there are many sets of data, four SRGMs, and a few different methods for preprocessing and bias adjustment, there are too many plots to be included here. We show only some representative plots and present the summary information along with the observations and discussions.

While some SRGMs ussally works better than others, there is no single SRGM that will work best for every possible software failure data all the time. It is also true that there is no single scheme introduced in section 2 will work best for every case all the time. If a scheme works best in a few situations but does poorly in others, we cannot claim that scheme as a good scheme. The first thing we tried to observe is to find a guideline to be followed for each scheme to get an overall good result for that scheme. Only then can we compare the performance of different schemes using their specific guidelines. Our objective is to to develop approaches that are independent of individual data set and, will generally give good predictive quality.

Fixed size grouping Three possible scenarios appear in this case as represented by Figure 2. The X-axis represents the number of data points (in most cases the number of defects) grouped together. The Y-axis represents the average relative error in percent for all the plots in this paper. Figure 2(a,d) are plots

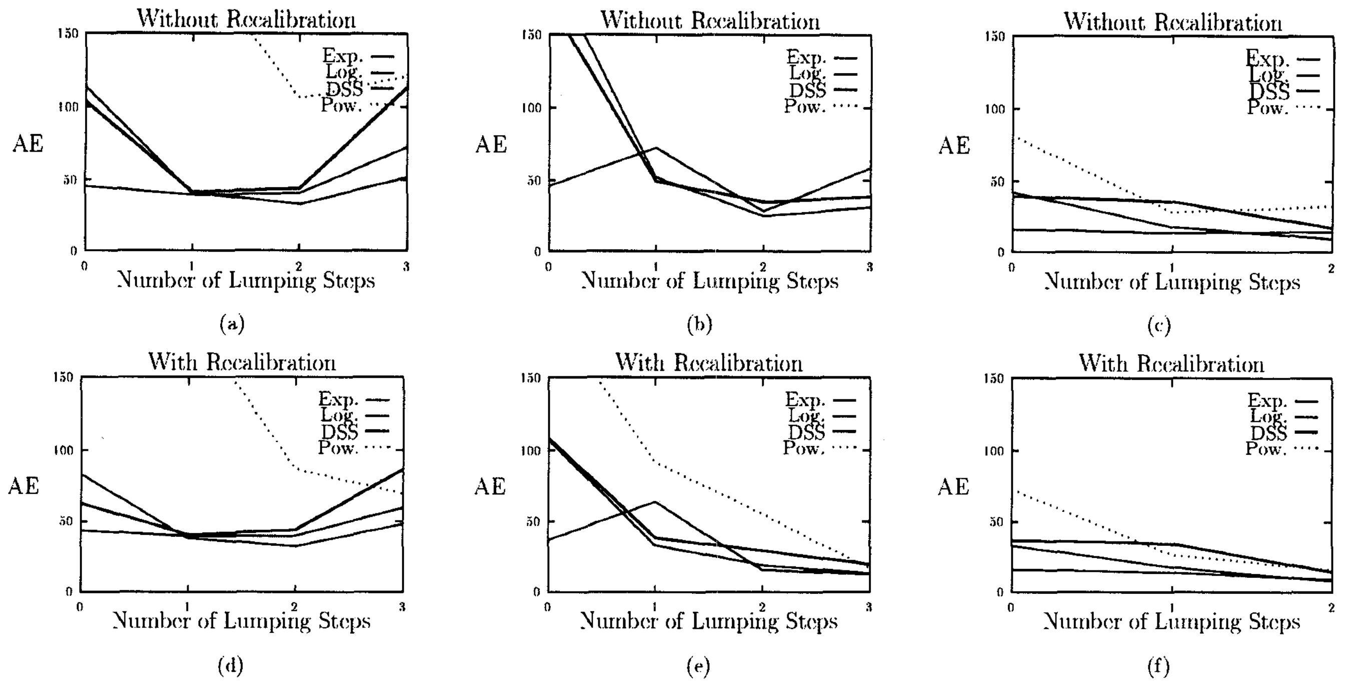

Figure 3: Representative plots for lump grouping

for data set T1. These also represents the observations for data sets T18, SS1A and some other data sets. In this class, the projection error decreases initially as the group size increases. As group size further increases, the projection error starts to increase. This increases can be very significant for some SRGMs for some data sets. Figure 2(b,e) are plots for data set TSW and they also represent the result for data sets SS4 and some other data sets. For this class the projection error decreases initially as the group size increases, and then remains relatively flat, and hence there is a wide range of preferred group sizes. Figure 2(c,f) are plots for T2 and representive of data sets HP2 and some other data sets. There the initial projection error is good and there is little significant improvement through grouping. Considering all the three class of data sets, the optimal grouping which consistently results in smaller prediction error is to have about 50 groups after grouping if no recalibration is involved, or to have about 20 groups if recalibration is used. The effect of recalibration on the range of optimal group size is obvious from the plots.

The logrithmic model gives the best predictions if no grouping is made. It is also least sensitive to grouping, i.e., the predictive quality changes little when the grouping size varies. The Delayed S-shape model is the most sensitive to group size. A little grouping can help a lot but grouping too many points together can severely reduce the predictive accuracy.

Without grouping, the performance of four models differ quite significantly for some data sets, and there is no model which works best in all the cases, though "on the average" the logarithmic model is apparently superior to the others and the exponential model is second to the logarithmic model. If the above mentioned optimal grouping is followed, and then the differences between the models become less significant.

Grouping lumps of failure intensity Figure 3 shows the effect of lump grouing on AE for data sets T18, TSW and T2. The number of lumping steps possible is data dependent. In general, larger data sets allow more lumping steps. Without recalibration, lump grouping twice gives overall best predictions. With recalibration, more lump grouping can enhance the prediction accuracy further for most of the data sets. In our experiments, we did as many lump groupings as allowed, which provides only a few data points in most cases for these plots.

Windowing Figure 4 shows representative plots for the effect of windowing. Here the X-axis represents the window width in the number of data points. The improvement in predictive accuracy is obvious when the window width increases initially, but then the average prediction error stays relatively flat with little fluctuation. The best window width can be determined by the total number of defects found (data points) divided by 20, or about the same as the best group size with recalibration.

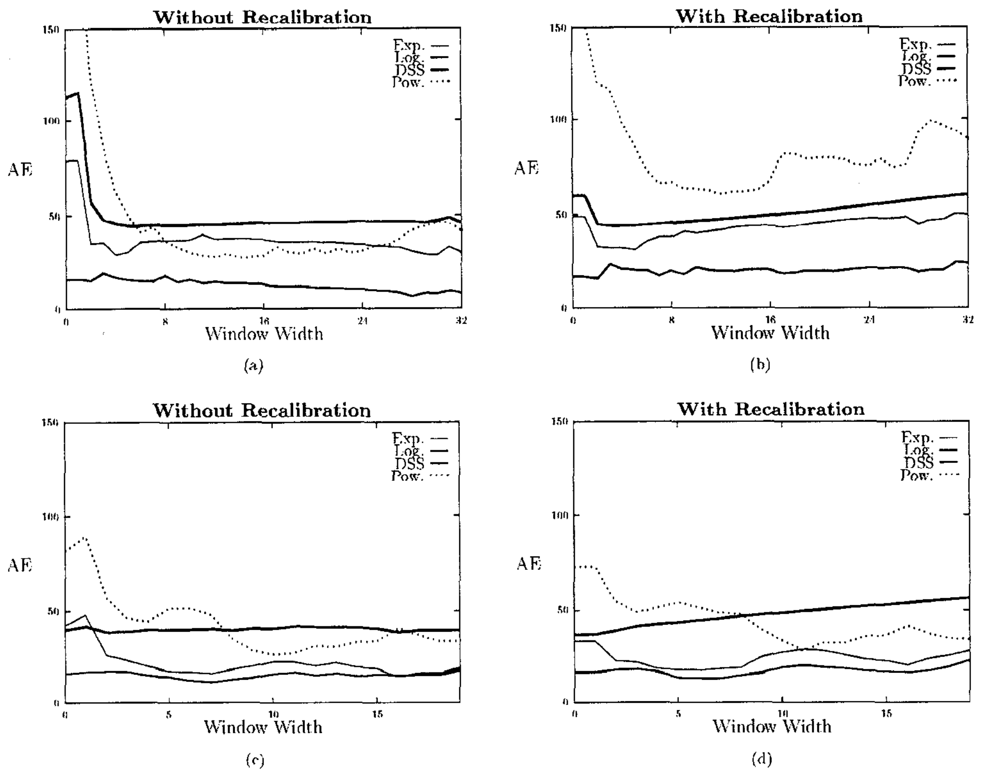

Figure 4: Representative plots for windowing

Like fixed size grouping and lump grouping, the windowing approach gives significant improvements over the raw data and is more stable than the other two approaches in that a wider range of window width can be regarded as optimal.

Recalibration enhances the power model significantly for some data sets, but weakens it for some other data sets. Overall there seems to be no need to combine recalibration with the windowing approach. The performance achieved with windowing is similar to that with fixed-size grouping and is not as good as with lump grouping.

Weighted least square Figure 5 shows representive plots on the effect of the weighted least square approach. The origin on the X axis corresponds to equal weighs for every data point; moving towards the 1 point on X axis corresponds to the recent past data points being given increasing weights. It is clear from the plots that the weighted least square approach helps little in predictive accuracy. It is not recommended based on the experiments we have done.

Adaptive approach The effect of adaptive approach (recalibration) can be observed in Figures 2 through Figure 5. This scheme reduces the average predictive error as seen by comparing the plots with and without recalibration. This is true especially for situations with large prediction errors, which are probably caused by bias in prediction and are hence more suitable for the adaptive approach.

The improvement by the adaptive approach alone is less than that achieved by the grouping and windowing approaches alone. Based on our observations, recalibration is specially recommended for use with fixed size grouping. In addition to reducing AE, it makes overgrouping less likely. It is not effective in conjunction with the windowing approach.

Best Model and Best Approach Earlier work [11] showed that the logarithmic model is superior to the other models, and the exponential model is close to the logarithmic model. It also showed that the power model is very inconsistent across different data set, and that the delayed S-shape model performed the worst. This is again observed in our experiments when no enhancing techniques are used (plots with horizontal axis at 0). However this study shows that, different models have different degrees of enhancement possible when the above techniques are used. We should take this into consideration when comparing different models.

Table 3 shows the effect of preprocessing and bias adjustment on the prediction accuracy for different data sets. In Table 3, all the values are relative AEs in percent; column *Raw* represents the AE obtained with-

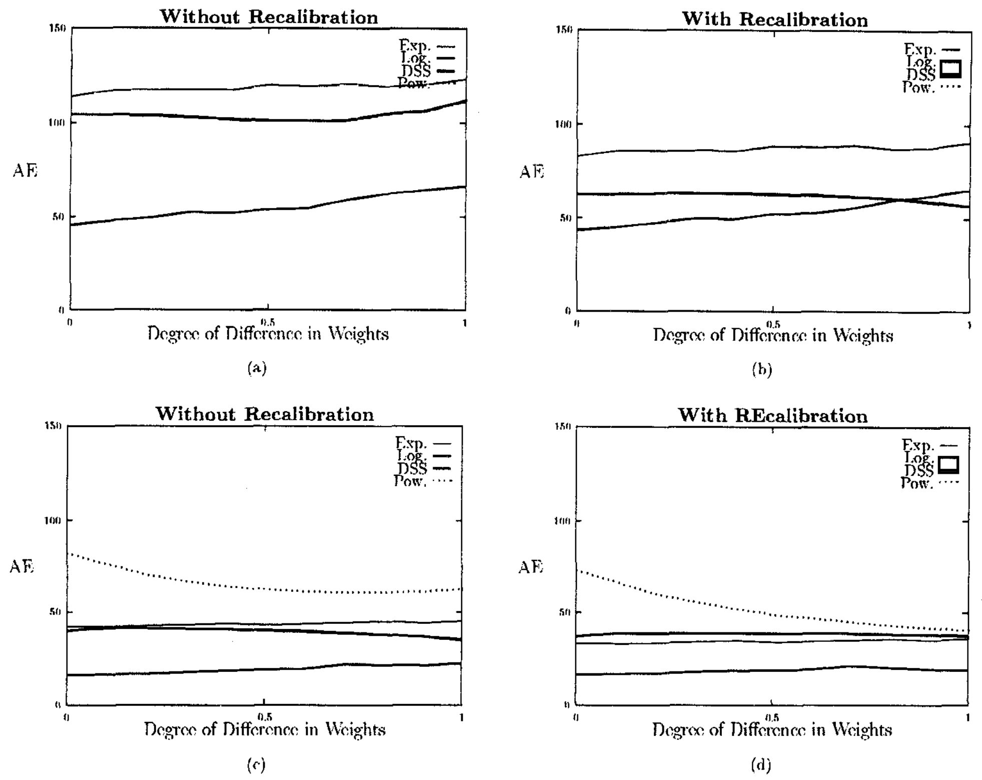

Figure 5: Representative plots for weighted least square

out using any enhancing techniques; *FSG* corresponds to fixed-size grouping using 50 grouped data points; *LG* corresponds to lump grouping twice; *Win.* corresponds to windowing with the window width given by the initial number of data points divided by 20; *Wei* corresponds to the weighted scheme with half of the total weights evenly distributed; *Rec.* stands for Recalibration; for fixed-size grouping with recalibration (*F&R*), 20 data points were used with recalibration; for lump grouping with recalibration (*L&R*), we did lumping as many times as possible followed by recalibration.

Table 4 gives the average AEs over all the data sets. Windowing, fixed-size grouping and lump grouping with recalibration can dramatically improve the prediction accuracy for "weak" SRGMs. When the enhancing techniques are properly applied, the different models perform close to each other. For instance, without enhancing, the Power model is much worse than the Logarithmic model; but with lumping and recalibration, the overall prediction error with the Power model is even lower than that with the Logarithmic model. Thus the proper application of the enhancing techniques can be more important than the selection of a SRGM.

Figure 6 depicts the effect of preprocessing (fixed-size grouping and lump grouping) followed by recalibration on AE. For comparision the effect of recalibration alone without grouping is also shown. It is easy to see that lump grouping with recalibration is the best way to achieve low prediction error for all four models.

Since lump grouping is superior to the other preprocessing methods considered, the intuitive motivation behind lump grouping is provided here. The very purpose of software failure data preprocessing is to filter out the noise and yet to retain the general trend in the data. By grouping, the noise manifested as fluctuations in failure intensities are averaged out by grouping a few data points together. With fixed size grouping, this is done somehow blindly in the sense that it does not take into consideration any characteristics of individual data. If we are grouping a data set by hand to smooth the noise, it is natural to group the most noisy data points, which are seen as either sharp peaks or deep valleys in the failure intensity plots. We did try to group a few data points manually in this way and found that it worked well. Then we came up with the idea of grouping data points with rising or decreasing failure intensities and lump grouping. Since the other two methods were not as effective, only lump grouping was reported here.

Table 3: Summary of the Relative AEs

Data	Model	Raw	Preprocessing			Bias Adjustment			
			FG	LG	Win.	Wei.	Rec.	F&R	L&R
SS1A	EXP	153	28	13	19	163	88	25	3
	POW	233	24	49	69	212	194	19	4
	DelS	126	28	23	15	183	91	40	1
	LOG	11	19	7	8	15	11	20	4
SS1B	EXP	150	35	31	27	152	100	36	29
	POW	172	17	19	13	163	119	12	18
	DelS	170	52	38	40	170	89	31	32
	LOG	28	27	26	26	26	28	25	23
SS1C	EXP	161	41	29	32	29	115	34	10
	POW	203	28	12	32	12	158	11	12
	DelS	154	44	43	39	43	93	32	19
	LOG	23	20	21	23	21	23	19	8
SS2	EXP	106	35	59	66	148	86	35	58
	POW	251	36	12	28	264	167	23	17
	DelS	215	35	24	30	243	120	34	12
	LOG	60	42	58	60	59	61	32	57
SS3	EXP	121	34	30	37	82	85	26	9
	POW	243	24	26	24	197	175	18	18
	DelS	160	39	27	35	162	77	34	10
	LOG	25	24	21	23	26	25	21	7
SS4	EXP	176	22	13	20	172	128	13	7
	POW	279	32	23	29	223	219	14	6
	DelS	500	36	30	32	150	500	37	28
	LOG	16	11	11	11	18	17	7	7
T1	EXP	79	30	28	36	79	49	38	13
	POW	184	45	35	36	167	153	21	9
	DelS	112	64	73	44	99	60	52	16
	LOG	16	16	24	17	18	16	16	8
T2	EXP	42	23	9	20	43	33	17	8
	POW	82	49	33	44	62	73	48	16
	DelS	40	37	17	39	40	37	38	14
	LOG	16	15	15	14	19	16	11	9
T3	EXP	57	23	18	21	62	31	17	16
	POW	126	57	21	28	105	95	40	24
	DelS	60	36	35	40	49	37	39	16
	LOG	17	14	17	23	19	17	13	17
T5	EXP	165	19	27	25	154	107	13	13
	POW	219	65	46	17	180	157	66	8
	DelS	125	40	38	38	125	67	37	19
	LOG	16	12	16	16	17	16	9	9
T6	EXP	186	62	19	68	183	121	47	20
	POW	246	91	12	85	237	184	51	11
	DelS	177	97	32	58	190	89	51	33
	LOG	25	23	11	22	28	25	21	14
T18	EXP	114	41	33	32	120	83	37	48
	POW	263	113	106	143	259	235	75	70
	DelS	104	51	44	41	101	63	43	87
	LOG	46	43	40	47	54	44	37	60
T40	EXP	136	50	44	55	138	94	54	8
	POW	219	182	212	170	257	154	97	2
	DelS	78	109	97	57	150	51	70	9
	LOG	97	81	56	59	106	89	64	2
Proj1	EXP	79	54	12	37	79	62	26	8
	POW	64	42	26	41	62	44	15	30
	DelS	74	57	12	41	66	54	35	13
	LOG	92	86	71	56	92	72	28	66
Proj3	EXP	92	86	80	84	91	92	84	79
	POW	73	52	39	29	75	65	49	15
	DelS	80	59	50	41	78	72	57	21
	LOG	90	93	79	81	90	87	78	77
Proj4	EXP	91	82	68	73	90	86	68	64
	POW	65	40	58	60	70	52	23	2
	DelS	73	55	38	48	73	61	44	2
	LOG	94	87	76	77	94	87	90	66
YT1	EXP	24	18	8	14	19	17	16	11
	POW	55	75	67	78	60	54	71	79
	DelS	37	25	20	22	43	25	16	12
	LOG	42	12	31	19	46	26	27	12
YT3	EXP	22	8	14	18	12	16	10	22
	POW	76	63	74	59	81	75	64	39
	DelS	28	22	16	22	27	23	21	13
	LOG	67	21	44	35	76	35	7	38
HP2	EXP	56	32	13	35	56	40	19	18
	POW	42	28	43	14	45	32	22	0
	DelS	60	36	26	44	60	43	28	10
	LOG	66	38	10	33	65	40	14	5
TSW	EXP	197	56	25	48	188	107	48	13
	POW	375	142	177	218	344	198	64	17
	DelS	172	53	35	32	157	108	24	20
	LOG	46	34	29	67	84	37	5	13
Usbar	EXP	134	35	47	35	123	85	29	22
	POW	281	130	157	190	272	193	181	33
	DelS	101	46	34	38	97	70	49	35
	LOG	78	72	68	84	75	69	49	33

Notes on Table 3 and 4:

1. Raw: AE obtained without using any enhancing techniques;
2. FG: Fixed-size grouping using 50 grouped points;
3. LG: Lump grouping twice;
4. Win.: Windowing;
5. Wei.: Weighted least square;
6. Rec.: Recalibration only;
7. F&R: Fixed-size grouping with recalibration;
8. L&R: Lump grouping with recalibration.

Table 4: Average AEs Over All the Data Sets

Model	Raw	Preprocessing			Bias Adjustment			
		FG	LG	Win.	Wei.	Rec.	F&R	L&R
EXP	111	38	29	38	103	77	32	22
POW	178	63	59	67	159	133	46	20
DelS	126	48	35	37	109	87	38	20
LOG	46	37	34	38	49	40	28	25

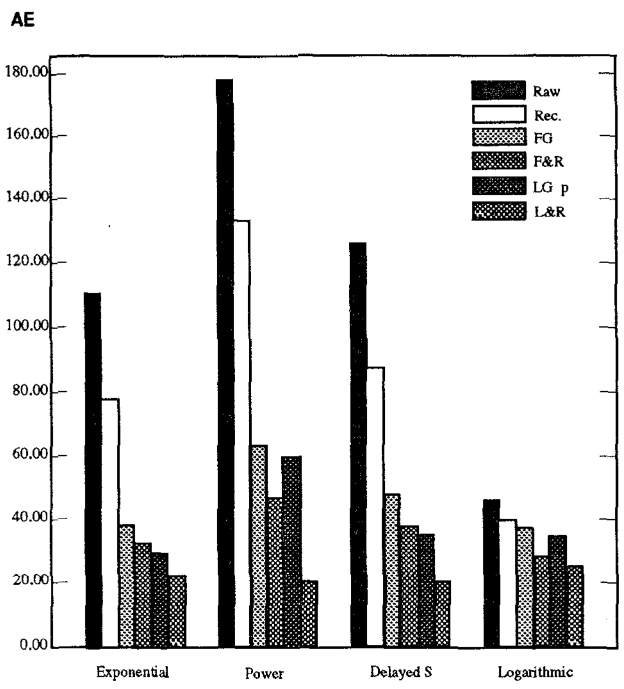

Figure 6: The effect of preprocessing and recalibration

4 Concluding Remarks

As we can see from the experimental results, different SRGMs have different characteristics with respect to the enhancing techniques. Logarithmic model gives best or close to the best predictions in most of the cases with raw data. Its performance is improved only modestly using the techniques considered here. For other models, windowing, lump grouping or fixed size grouping with recalibration can significantly reduce the average prediction error. Lump grouping combined with recalibration leads to the best prediction.

One may argue that a comparision is fair only if different SRGMs are compared using their optimal achieveable performance. The logarithmic model is indeed superior if raw software failure data and the model are used directly to make projections. However proper application of the enhancing techniques can render the difference among the models to such a small degree that it can be more important to select proper combination of the enhancing techniques than to choose a SRGM.

5 Acknowledgements

We would like to thank A. Nikora, S. Brochlehurst, N. Schneidewind, Y. Tohma for providing us some of the data sets used in this study.

References

[1] J. D. Musa, A. Iannino, K. Okumoto, *Software Reliability - Measurement, Prediction, Applications*, McGraw-Hill, 1987.

[2] J.D. Musa, "Software Reliability Data", Data and Analysis Center for Software, *Rome Air Development Center, Rome, NY.*

[3] A. A. Abdel-Ghaly, P. Y. Chan , B. Littlewood, "Evaluation of Competing Software Reliability Predictions", IEEE Trans. Reliability, Vol. SE-12, pp. 950-967, September 1986.

[4] S. Brocklehurst, P. Y. Chan, B. Littlewood, J. Snell, "Recalibrating Software Reliability Models", IEEE Trans. Software Engineering, Vol. 16, pp. 456-470, April 1990.

[5] S. Brocklehurst, B. Littlewood, "New Ways to Get Accurate Reliability Measures", IEEE Software July 1992, pp. 34-40.

[6] N. Karunanithis, D. Whitley and Y. K. Malaiya, "Prediction of Software Reliability Using a Connectionist Models", IEEE Trans. Software Engineering, Vol. 18, No. 7, July 1992, pp. 563-574.

[7] Y. K. Malaiya, A. von Mayrhauser, P. K. Srimani, "The Nature of Fault Exposure Ratio", Proc. Int. Symposium on Software Reliability Engineering, pp. 23-32, October 1992.

[8] Y. K. Malaiya, A. von Mayrhauser, P. K. Srimani, "The Constant Per-Fault Hazard Rate Assumption", Proc. 2nd Bellcore/Purdue Symposium on Issues in Software Reliability Estimation, pp. 1-9, 1992.

[9] Y. K. Malaiya, N. Karunanithi, P. Verma, "Predictability Measures for Software Reliability Models", Software Reliability Models, IEEE CS Press, pp. 60-65, 1991.

[10] Y. K. Malaiya, Sumit Sur, N. Karunanithi, Y. Sun, "Implementation considerations for software reliability", *8th Annual Software Reliability Symposium*, June 1, 1990.

[11] Y. K. Malaiya, N. Karunanithi, Pradeep Verma, "Predictability of software reliability models", IEEE Trans. Reliability, Dec., 1992, pp. 539-546.

[12] P. Verma, Y. K. Malaiya, "In Search of the Best Software Reliability Reliability Model", Proc. 7th Annual Software Reliability Symposium, May 1989, pp. 76-92.

[13] M. Chen, A. P. Mathur, V. J. Rego, "Effect of Testing Techniques on Software Reliability Estimates Obtained Using Time-Domain Models", 10th Annual Software Reliability Symposium, Denver, pp.116-123, 1992.

[14] G. A. Kruger, "Validation and Further Application of Software Reliability Growth Models", *Software Reliability Models*, IEEE, Inc., pp. 66-70, 1991.

[15] K. Matsumoto, K. Inoue, T. Kikuno and L. Torii, "Experimental Evaluation of Software Reliability Growth Models", *IEEE Proc. FTCS-19*, pp. 148-153, June 1988.

[16] M. Ohba, "Software reliability analysis models", *IBM Journal of Research and Development*, Vol. 28, pp. 428-443, 1984.

[17] M. Xie, M. Zhao, "The Schneidewind Software Reliability Model Revisited", Proc. Int. Symposium on Software Reliability Engineering. pp. 184-192.

[18] N. D. Singpurwalla, R. Soyer, "Assessing software reliability growth using a random coefficient autoregressive process and its ramifications", *IEEE Trans. Software Engineering*, Vol. SE-11, pp.1456-1465, 1985.

[19] Y. Tohma, K. Tokunaga, S. Nagase, Y. Murata, "Structural approach to the estimation of the number of residual software faults based on the hypergeometric distribution", *IEEE Trans. Software Engineering*, Vol. 15, pp. 345-355, 1989.

[20] Y. Tohma, H. Yamano, M. Ohba, R. Jacoby, "The Estimation of Parameters of the Hyper-Geometric Distribution and its Application to Software Reliability Growth Model", *Tech. Report No 900830*, Dept. of Computer Science, Tokyo Institute of Technology, 1990.

[21] Y. Tohma, H. Yamano, M. Ohba, R. Jacoby, "Parameter Estimation of the Hyper-Geometric Distribution Model for Real Test/Debug Data", *Tech. Report No 901002*, Dept. of Computer Science, Tokyo Institute of Technology, 1990.

[22] A. Nikora, Personal Communication, 1992.

[23] M.R. Lyu, A. Nikora, "Applying Reliability Models More Effectively", IEEE Software, July 1992, pp. 43-52.