Better Reliability Assessment and Prediction Through Data Clustering [*]

Jeff Tian[†], *Member, IEEE*

Abstract

This paper presents a new approach to software reliability modeling by grouping data into clusters of homogeneous failure intensities. This series of data clusters associated with different time segments can be directly used as a piecewise linear model for reliability assessment and problem identification, which can produce meaningful results early in the testing process. The dual model fits traditional software reliability growth models (SRGMs) to these grouped data to provide long-term reliability assessments and predictions. These models were evaluated in the testing of two large software systems from IBM. Comparing to existing SRGMs fitted to raw data, our models are generally more stable over time and produce more consistent and accurate reliability assessments and predictions.

Keywords: *Software reliability, data grouping, cluster analysis, software reliability growth models (SRGMs), input domain reliability models (IDRMs), data cluster based reliability models (DCRMs).*

1 Introduction

The reliability of a software system can be defined as its probability of failure-free operations for a specific time period or for a given set of input under a specific environment [11, 12]. Failures are behavioral deviations caused by internal faults (or defects) in the software. One fundamental assumption in reliability analysis is the infeasibility of complete elimination of faults in software systems. Therefore, failure-free operations cannot be guaranteed, but only statistically assured through the use of various software reliability models.

Two commonly used approaches to software reliability analysis are: input domain approach using various input domain reliability models (IDRMs) and time domain approach using various software reliability growth models (SRGMs) [1, 5, 11, 14]. At product release or other important project milestones, various IDRMs based on repeated random sampling are often used to assess product reliability and help make important project decisions. During software testing, the effect of reliability change (or *growth*) due to defect removal is analyzed by various SRGMs to provide reliability assessments and predictions.

In practical applications, it is common to have considerable data fluctuations due to product and development process dynamics and variations, which may lead to unstable modeling results. Properly treated or grouped data can generally reduce such fluctuations and produce models that fit the observations better and provide better reliability assessments and predictions [2, 9, 16, 17].

After a brief examination of the suitability of existing SRGMs during testing for large software systems in Section 2, we propose a new approach in Section 3: our data cluster based reliability models (DCRMs). The overall data are divided into different time segments, with each segment possessing a homogeneous failure intensity. This kind of grouped data can be used directly as piecewise linear models or fitted to traditional SRGMs for product reliability analysis. Modeling results are compared in Section 4 to demonstrate the superior stability and accuracy of our models in reliability assessments and predictions. Finally, we summarize the paper and discuss future research directions in Section 5.

2 Reliability Analysis for Large Software Systems

There are several common assumptions for various SRGMs [5]: The software is assumed to be used in an environment that resembles its actual usage by target

[*]The final version of this paper appeared in *IEEE Trans. on Software Engineering*, 28(10):997-1007, Oct., 2002.

[†]J. Tian is with the Department of Computer Science and Engineering, Southern Methodist University, Dallas, Texas 75275. Phone: (214) 768-2861. Fax: (214) 768-3085. E-mail: tian@engr.smu.edu.

customers, so that the analysis results can be extrapolated to the likelihood of in-field product failures. Failure intervals or period failure counts are assumed to be independent, which implies randomized testing. Time is used as the basis to define reliability, which implies equivalence of time units and requires proper time measurement to reflect usage. We next characterize testing for large software systems and examine the validity of these assumptions.

This study is a continuation of a series of studies [18, 19, 20] for large commercial software systems, which include relational database products, compilers, and computing environments from IBM. A large software system can generally be characterized by its large size, usually exceeding several hundred thousand lines of source code, high complexity, diverse functionality, many components developed over a long period, large user population, and diverse usage environments. The overall testing effort, which usually lasts over several months to more than a year, is divided into functional areas and sub-phases to allow for test execution and progress tracking on a smaller, more manageable scale. Test activities and defect discoveries are generally associated with test runs or executions of specific test cases.

Test workload usually varies considerably due to various reasons, including: shifting focus of testing among different functional areas and sub-phases, progression of test cases, learning curve and staff-level variations. Such large variations make usage independent time measurements unsuitable for reliability modeling. Based on our previous study of different time measurements for reliability modeling [19, 20], two variations of usage dependent time measurements are used in this paper: 1) the number of *test runs* as the rough time measurement; and 2) the detailed workload measure, generically referred to as *transactions*. Both of these measurements are referred to as *usage time* hereafter.

Because of the large size and the lack of precise customer usage information due to diverse user population, a *scenario-based* testing strategy is commonly used. The test scenarios consist of some randomized workload executed within a framework that roughly describes customer usage situations. In addition, testing is not only used to evaluate product quality, but also used to help locate and remove defects (faults). Once a failure is observed, usually a series of related test runs are conducted to help isolate the failure cause(s) and later to verify the defect fix(es).

Overall, this mixture of structured (centered around the framework of scenarios) and clustered (focused on fault localization and fixing) testing with some randomized workload, rather than purely randomized testing, dominates for large software systems. However, we also observe the following:

- *Approximating scenario-based testing with random testing:* Despite the individual dependencies due to testing according to scenarios, testing is generally conducted by many testers in parallel, interleaving in some arbitrary fashion. As a result, the overall testing still resembles random testing. The key to this approximation is parallelism and interleaving, randomized workload, and proper data granularity.

- *Run correlation due to defect fixing is limited to a short period*, due to the effort made to locate and fix discovered defects quickly, usually within a few days. In addition, hundreds or more defects are discovered and fixed during testing for large software systems. Such large number of defects, each of which fixed within a short period, does not lead to long-term dependencies among test runs.

To summarize, the overall testing process still resembles random testing without long-term dependencies at a coarse granularity, although there may be dependencies among test runs within a short time window. The short-term dependency implies that existing SRGMs fitted to raw data may produce biased results because period independence assumption is violated, and points to the need for alternative ways to analyze product reliability. The long-term independency suggests the possibility of using properly treated data with existing SRGMs. Data treatment usually involves data grouping [16, 17] or data censoring [2, 9]. The availability of usage time associated with test runs and failure observations under our environment implies that we do not need to use data censoring to screen or condense data. Instead, appropriate data grouping technique(s) can be used to contain the short-term dependency in our data so that the grouped data will show little or no long-term dependency, as described in the next section.

3 DCRMs: Model Construction and Usage

We next derive our data cluster based reliability models (DCRMs), describe their primary applications, and examine their relations to existing models.

3.1 Clustering periods with homogeneous failure intensities

The failure intensity (or failure rate) of a given time period, defined to be the number of failures per unit

Table 1. Failure intensity computation for an individual period

1. Determine the period P.
2. Identify each test run (run i) falling into the period ($i \in P$).
3. Count the number of failures, f, for the period P.
4. Compute the time duration for the period, t, as the summation of the individual time for all the individual runs, t_i for the i-th run, falling into the period. That is, $t = \sum_{i \in P} t_i$.
5. Compute the failure intensity, λ, as the number of failures divided by the time duration, i.e., $\lambda = f/t$.

Table 2. Segment failure rate computation for grouped data

0. Given: failure rate λ_j for individual period j of length t_j.
1. Select the external delimiter d_j for data point j. Let l_i and u_i be the lower and upper bounds that delineate segment i, i.e., $l_i \leq d_j < u_i$.
2. Compute the total time T_i for segment i as $T_i = \sum_{j, l_i \leq d_j < u_i} t_j$.
3. Compute the total failures F_i for segment i as $F_i = \sum_{j, l_i \leq d_j < u_i} \lambda_j t_j = \sum_{j, l_i \leq d_j < u_i} f_j$.
4. Compute the segment failure rate Λ_i for segment i as $\Lambda_i = F_i/T_i$

time, directly reflects the reliability of the software product at that time [11, 12]. There are essentially two ways to identify a period: 1) directly use the run sequence information; and 2) use external delimiters, commonly associated with calendar dates, weeks, etc. For a given time period, the failure intensity can be calculated as in Table 1.

When performing data grouping for our data, we can group related test runs into the same cluster to reflect their short-term dependency, and group unrelated runs into separate clusters to reflect their long-term independency. Equivalently, the overall testing duration can be partitioned into different segments. Failure rate Λ_i for a segment i can be computed from individual failure rate and timing information associated with test runs falling into the segment, as in Table 2. To determine the boundaries between neighboring segments, content-based run dependency analysis can be performed by examining the relations among individual runs. However, such analysis could be too costly due to the large amount of data to be examined. Automated data grouping are needed for practical applications.

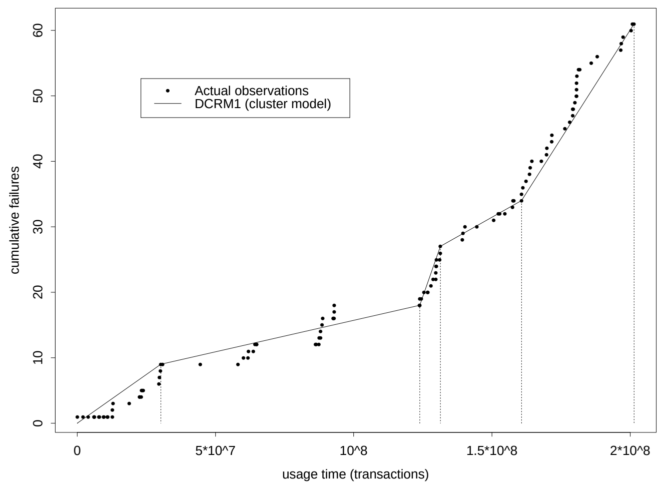
Because correlated runs typically lead to similar observed behavior, we can group neighboring data points with homogeneous failure intensities into the same segment and those with different failure intensities into different ones. The grouping of individual data points into such homogeneous clusters can be carried out using various statistical analysis techniques for clustering [21]. In this case, we have a single response variable (failure intensity), and a single predictor variable (time), a simplified clustering algorithm using tree-based models [3] supported by a commercial tool S-PLUS[1] can be used. A brief summary of tree-based modeling technique and its algorithm can be found in the appendix.

When using tree-based models to find data clusters of homogeneous failure intensities, we start with the complete set of data, and recursively partition it into smaller subsets. Conditions defined on usage time τ in the form of $\tau < c$ or $\tau \geq c$ defines a binary partition. Each recursive partitioning selects a cutoff value c to minimize the difference between the predicted (mean) failure rates and the actual individual failure rates in the partitioned subsets. However, the arithmetic mean is no longer appropriate, because it gives the predicted failure rate for segment i with N_i data points as:

$$\frac{\sum_j \lambda_j}{N_i} = \frac{\sum_j \frac{f_j}{t_j}}{N_i}$$

which is generally different from Λ_i in Table 2, unless all the t_j's are equal. Instead, we can use the weighted

[1]S-PLUS is a trademark of Insightful Corporation.



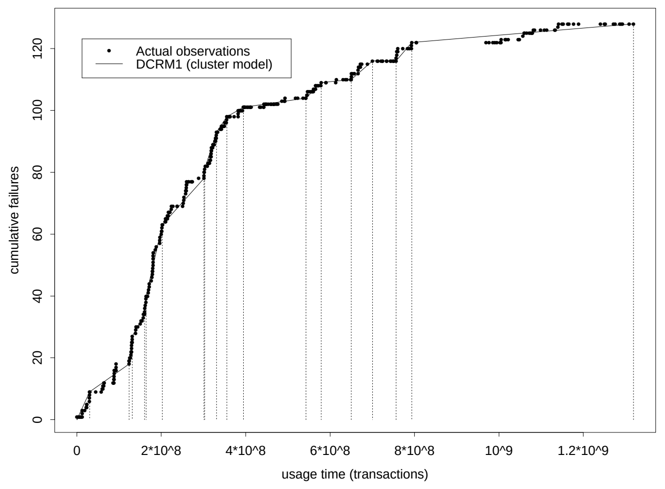


Figure 1. a. a DCRM1 at half-way of testing (left) and b. a DCRM1 at test completion (right)

average, an option available in the tool S-PLUS that supports our tree-based modeling, with individual failure rates, λ_j's, weighted by the corresponding usage time interval, t_j's, i.e.,

$$\frac{\sum_j t_j \lambda_j}{\sum_j t_j} = \frac{\sum_j f_j}{\sum_j t_j} = \frac{F_i}{T_i} = \Lambda_i$$

Consequently, these partitions derived from the corresponding tree-based model give us a series of segments. Different segments have different predicted (mean) failure rates, while individual data points within a segment have homogeneous failure rates close to the predicted (mean) failure rate for the segment.

3.2 DCRM1: Data clusters as a piecewise linear model

When represented graphically, the data clusters give us a piecewise linear model for the cumulative failures, maintaining constant failure rate for each time segment. We call this direct usage and interpretation "data cluster based reliability model, type 1", or DCRM1 for short. Fig. 1 is an example of a DCRM1 for product E studied in [20]. The cumulative failure arrivals are plotted against usage time (cumulative transactions), with our DCRM1 shown as a piecewise linear curve with dashed vertical lines to separate the time segments.

In this product, time stamp for each test run is available, thus the precise order for the run sequence and the corresponding cumulative usage time can be determined. The failure rate (Λ_i) over the time segments i's can be represented by discrete functions such as in Table 3, taking different values for different time periods defined by their cutoff values for cumulative usage

Table 3. Data clusters (grouped data) according to DCRM1

segment	τ cutoff	cumulative failures	# of runs	failure rate (Λ_i)
1	0	0	0	NA
2	30261536	9	31	2.974e-07
3	123855436	18	30	9.616e-08
4	131290212	27	16	1.211e-06
5	160634323	34	12	2.385e-07
6	164363318	40	6	1.609e-06
7	202584007	63	34	6.018e-07
8	301334843	78	44	1.519e-07
9	302830110	81	6	2.006e-06
10	330965459	93	39	4.265e-07
11	355372545	98	22	2.049e-07
12	394683837	101	24	7.631e-08
13	542796251	104	56	2.025e-08
14	578748899	109	22	1.391e-07
15	650221776	110	12	1.399e-08
16	700575963	116	17	1.192e-07
17	756410215	116	14	0
18	793301593	122	12	1.626e-07
19	1318682282	128	58	1.142e-08

time. For example, the last segment ($i = 19$) with $\Lambda_i = 1.142 \times 10^{-8}$ (or 1.142e-08 in the *scientific notation* used hereafter) is delimited by the usage time τ cutoffs, $793301593 \leq \tau < 1318682282$. The cutoff points for the partitions are determined by the corresponding tree-based model, using point failure rate λ_j ($\lambda_j = 1/t_j$ if j-th run within the segment is a failure with transactions t_j, and $\lambda_j = 0$ if it is a success, according to Table 1) as the response variable, and usage time τ as the predictor variable.

When exact run sequence cannot be determined, such as for product D studied in [18], the time index used, instead of the cumulative usage time associated with each run, can be used to delimit the partitions and to build DCRM1.

3.3 interpreting DCRM1 results

The predicted failure rate for each segment in our DCRM1 can be interpreted as applying various existing input domain reliability models (IDRMs) to a restricted subset of data. For example, in the Nelson model [14], one of the earliest IDRMs, the predicted reliability R based on repeated random sampling with n runs and f failures is given as:

$$R = 1 - \frac{f}{n} = 1 - \lambda$$

When test run count is used as the time measurement, the predicted failure rate λ for each segment from our DCRM1 is the same as Nelson model restricted to the same segment.

The predicted reliability in the Brown-Lipow model [1], another IDRM, is give by:

$$R = 1 - \sum_i \frac{f_i}{n_i} p_i$$

where f_i is the number of failures, n_i number of runs, and p_i the probability, all for subdomain i. When transaction is used as the time measurement, or when individual periods are associated with multiple runs, our DCRM1 for each segment is the same as the Brown-Lipow model, if we make the following substitutions: individual period j for subdomain i, measured usage time interval t_j for n_i, and the ratio between t_j to the total usage time for this segment $(t_j / \sum_j t_j)$ for p_i.

Our DCRM1 is also related to several SRGMs. Similar to the Jelinski-Moranda SRGM [7], we assume constant failure rate for each time period within a data cluster. However, we do not assume a functional form for failure rates over different time segments, in contrast to the fixed-size failure rate reduction after each observed failure in the Jelinski-Moranda model. Our model is data driven, similar to the approach in the Littlewood-Verrall SRGM [10] where the failure rate varies with latest observations. A key difference to these SRGMs is that each time segment in our DCRM1 consists of multiple runs and possibly multiple failures, while both Jelinski-Moranda and Littlewood-Verrall models are time-between-failure (TBF) models, requiring exactly one failure per period.

3.4 Using DCRM1 in reliability analysis

For any given time instance covered by the input data, one and only one specific time segment used in DCRM1 can be identified, because these segments form a partition of the whole time period. The reliability assessment is given by the corresponding predicted failure rate from the model. In particular, the current reliability can be assessed to be the predicted reliability for the last time segment. For example, the estimated reliability for product E at the product release can be characterized by its present failure rate of 1.142e-08 failures per transaction from its DCRM1. This last time segment can also be directly extrapolated to predict reliability into the near future. However, to avoid the risk of extending it too far into the next cluster, we need alternative means to predict future reliability.

Another important usage of DCRM1 is its ability to identify trouble spots or problematic areas. Certain segments with abnormally high failure rates (symptom of possible problems) can be easily detected by DCRM1. These segments or run clusters are usually related to likely problematic test scenarios, weak product functional areas, or other trouble spots. Followup analyses are generally needed to confirm the suggested trouble spots by DCRM1 and to discover the root cause for them, so that focused remedial actions can be applied to solve or alleviate the problems.

The use of DCRM1 in problem identification can be combined with our earlier work on tree-based reliability models (TBRMs) [18]. In building TBRMs, we used both time domain and input domain information associated with individual runs as predictors to identify and improve problematic areas. The reliability for a subset of runs was defined as the number of failures over the number of runs. That reliability definition is extended in this paper to cover more general cases where individual data points correspond to variable numbers of runs or time is measured using entities other than runs. This generalization allows us to build general TBRMs to link input and time domain information to observed failure rates for general situations, thus making our TBRMs more widely applicable.

Reliability growth can be qualitatively represented by the gradual reduction of failure rates as we move from earlier segments to later ones, although some fluctuations in segment failure rates are expected. However, if these kinds of “out-of-place” segments are a persistent phenomenon, it is an indication that the product is not ready for release yet. For example, Fig. 1a represents a series of data clusters, with a generally increasing trend in failure rates over successive time segments, clearly indicating the instability of the product up to that time. In fact, more than half (67) of the total testing defects (128) were yet to be found in the following few weeks. Without additional testing to detect and remove those additional defects, the product would have caused many problems to target customers. This use of DCRM1 is similar to trend analysis for software failure data in [8], where the super-additive curve for Fig. 1a could indicate the same general conclusion. A quantitative assessment of reliability growth can be carried out using DCRM2 to be described next.

3.5 DCRM2: Fitting SRGMs to data clusters

The lack of an assumed functional form among failure rates in different segments is both the strength and weakness of our DCRM1: On the positive side, it leads to a robust model, being able to fit to almost any data, and the modeling results can be used for many purposes. On the negative side, DCRM1 is hard to use for reliability predictions or for quantitative evaluation of reliability growth. DCRM1 also lacks parameters that have meaningful physical interpretations, such as N, the estimated total faults in the software system in various software reliability growth models (SRGMs).

To compensate for these weaknesses, we can extend our DCRM1 to include its dual model, our data cluster based reliability model, type 2, or DCRM2 for short, by fitting selected SRGMs to the data clusters as grouped data. To account for the different sizes of data clusters, we weigh each cluster by the number of individual data points it represents when fitting SRGMs in our DCRM2. For example, we can fit a specific SRGM, the Goel-Okumoto (GO) model [6], to the 19 segments in Table 3 as 19 data points, as shown in Fig. 2, where the fitted model is labeled DCRM2.GO.

Our use of grouped data in DCRM2 is similar to Schneidewind's approach where data-sensitive groupings and weights were selected to produce better fitted models [16]. Because short-term dependencies are generally isolated to within the same segments by our DCRM1, testing associated with different segments are now essentially independent because of the lack of long-term dependencies. This situation matches well the general assumption of independent testing periods for general SRGMs [5], providing justifications to the approach we use in DCRM2. In contrast, this assumption of independent testing periods is less likely be be satisfied by the raw data because of short-term dependencies.

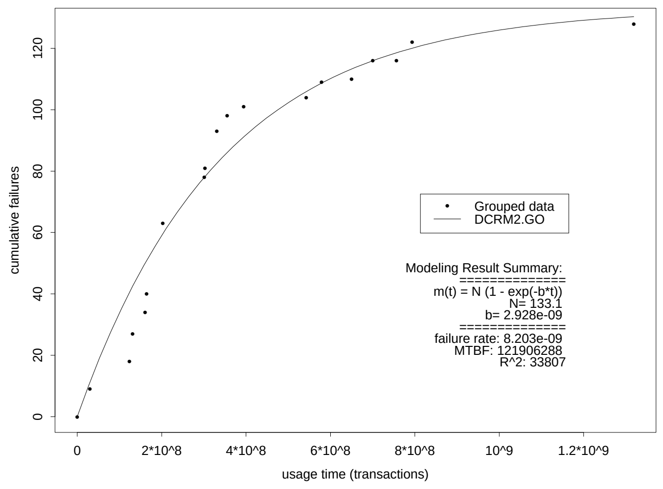


Figure 2. DCRM2: model fitted to grouped data

3.6 Selecting SRGMs for DCRM2 applications

Among the SRGMs, *time-between-failure* (TBF) ones fit models to inter-failure intervals as input data, and *failure count* (FC) ones fit models to failure counts and associated period lengths as input data [11, 12]. For DCRM2, grouped testing periods, or time segments in DCRM1, and associated failure counts are used as input data in various SRGMs. Since each segment consists of multiple test runs, and possibly multiple failure observations, TBF models are not suitable. Only FC-SRGMs can be used on such grouped data.

Among the FC-SRGMs, many treat the failure arrivals as a non-homogeneous Poisson process (NHPP), with the number of failures $N(t)$ for a given time interval [0, t] give as:

$$P(N(t) = n) = \frac{m(t)^n}{n!}\ e^{-m(t)}$$

where $m(t)$ is the mean function. Different choices of $m(t)$ result in different NHPP models. Some commonly used NHPP models include:

- Goel-Okumoto (GO) model [6], with $m(t) = N(1 - e^{-bt})$, where the model parameter N is the estimated total number of defects and b the constant for model curvature.

- Logarithmic Poisson model by Musa and Okumoto (MO model) [13], with $m(t) = \frac{1}{\theta}\log(\lambda_0 \theta t + 1)$, where the parameters are λ_0, the estimated initial failure rate, and θ, the constant for model curvature.

Table 4. Cumulative data for the last 8 weeks of testing

week	$F-7$	$F-6$	$F-5$	$F-4$	$F-3$	$F-2$	$F-1$	F
cumulative failures	102	104	109	116	122	125	128	128
cumulative transactions	4.59e08	5.43e08	6.14e08	7.01e08	9.77e08	1.08e08	1.25e09	1.32e09

- S-shaped model [22] and Schneidewind model [15].

Although S-shaped model can generally fit the calendar time failure data better than most other models, once proper usage time measurement is selected, it consistently over-estimates product reliability by a large margin compared to other models in our previous applications [19, 20]. Schneidewind model [15] requires equal length for each observation period, — an assumption clearly violated by our grouped data. Therefore, S-shaped model and Schneidewind model were not selected for our analysis.

On the other hand, GO and MO models have been successfully used by us to analyze product reliability in [19, 20]. They also complement each other nicely, as suggested by Musa [12] and confirmed by our previous experience: The GO model is usually slightly optimistic and the MO model slightly pessimistic, providing a pair of upper and lower bounds for reliability estimates. In addition, our assumptions of homogeneous testing segments, and independence among the separate segments, match well with the model assumptions. Consequently, we primarily use GO and MO models in subsequent reliability analyses. We label the DCRM2 obtained by fitting GO or MO models to our grouped data as DCRM2.GO or DCRM2.MO respectively.

With such modeling results, we can quantitatively assess product reliability by examining the estimated failure rate and other quantities, predict future reliability by extending the fitted model into the future, evaluate reliability growth by comparing the end reliability with the beginning reliability, estimate time and resource to reach a given reliability goal, and help with product release decisions.

4 Modeling Result Analysis and Comparison

Our DCRMs constructed above can be used to fulfill various practical needs for product reliability analysis. We next evaluate the effectiveness of these models in practical applications, and compare them to other alternatives.

4.1 Product and modeling alternatives

The primary object of this performance study is product E developed in the IBM Software Solutions Toronto Laboratory we studied previously in [20] for workload and reliability measurement. Detailed test measurement is available for product E, including transactions and time-stamp for each test run, and related failure observations.

Another product, product D, from the same organization we studied in [18], is also used in this study to cross validate various results. However, test runs in product D are only loosely associated with testing days, which are, in effect, already grouped by testing days. For example, an average of 25.2 runs took place per day for product D. Therefore, although DCRMs can be built for this kind of product, the data grouping effect are not expected to be as visible as that for individual runs, as will be seen later in this section.

The main alternatives to our DCRMs are existing SRGMs fitted to raw data. As described in Section 3, our DCRM2 are primarily used with GO and MO models. Therefore, the primary models for our result comparison are:

- DCRM1 applied to products D and E.
- GO and MO models fitted to the original or raw data from products D and E.
- DCRM2.GO and DCRM2.MO fitted to data clusters identified by DCRM1 above.

To strike a balance between modeling overhead and the need for timely feedback, weekly modeling activities were carried out during testing for these products. To make our comparison meaningful to actual testing environment, we attempted to fit different models to successive data set and compare their performance on a weekly basis in this study. In the case where comparative results are similar, we only present one set of results, and state the validity of the same patterns or observations.

4.2 Model applicability and points for result comparison

Our two-staged approach, data clustering through DCRM1 and fitting existing SRGMs to grouped data

Table 5. Goodness-of-fit (R^2) values for different models

week	$F-7$	$F-6$	$F-5$	$F-4$	$F-3$	$F-2$	$F-1$	F
DCRM1	382	390	456	443	453	521	582	566
GO (raw)	12971	17891	22072	24483	25773	26723	26871	26925
DCRM2.GO	14260	22178	24588	25743	26767	27563	27626	27662
MO (raw)	13061	18784	25006	30366	36254	44201	48483	50794
DCRM2.MO	14401	24218	28640	32119	37459	45065	50116	53196

through DCRM2, is in contrast to the approach of deriving general models to handle the general failure correlations in [4]. Although our approach is not as universally applicable as [4], it is easy to apply, uses existing SRGMs, and fits well with the application environments for the products we studied. In addition, the basic assumptions for our DCRMs can be easily satisfied under the testing environment for many large software systems, making our approach applicable to a wide variety of environments.

Our DCRM1 can be fitted to any data set as long as the required data, individual failure rate λ_i and usage time τ (or other time period delimiters) are available. Once these data are collected, right after testing started, DCRM1 can be produced. This early applicability is unique to DCRM1 as compared to other alternatives, because typically SRGMs can only be produced for the later part of testing when failure arrivals stabilize to a degree to demonstrate a general trend of reliability growth or a sub-additive cumulative failure curve [8].

For product E, this general trend of reliability growth only became observable by the last 8 weeks before product release, which represent less than one third of the total testing weeks. Consequently, the GO model can only be fitted to data accumulated up to those weeks. Similarly, only the last 8 weeks produced meaningful fitted MO models. Some earlier fits were available, but with $\theta < 0$, signaling reliability decline instead of reliability growth, which cannot be used to predict the overall trend of reliability growth. As expected, DCRM2.GO and DCRM2.MO also started to fit for the last 8 weeks of testing.

As a result, subsequent comparisons of modeling results for product E are based on models produced for the last 8 weeks of testing. We mark the last week as F (for final week before product release), and the preceding weeks as $F-1$ back to $F-7$. A model produced for a particular week W_i uses data cumulatively up until the end of that week. The actual data at the end of these weeks are summarized in Table 4. Model performances can be compared using models fitted to these data sets.

For product D, the overall failure arrivals seem to be more stable and better resemble general reliability growth trend. Consequently, various SRGMs can be fitted to either the raw data or the grouped data relatively early, after approximately one third of the way into testing. This allows us extended time period for our performance comparison.

4.3 Goodness-of-fit comparison

When interpreted as a piecewise reliability model such as in Fig. 1, our DCRM1 conforms well with the actual observations. In addition, for the numerous data sets we studied for various different products, only a few segments (say, around 5 to 10) are enough to produce fitted models that fit the observations better than traditional SRGMs. On the other hand, DCRM2 using the same SRGMs on the grouped data is expected to fit the raw data slightly worse that those directly fitted to the raw data, because the model parameters were optimized to fit a different data set.

This comparison can be quantified by the various goodness-of-fit statistics computed for different fitted models. One commonly used such statistic is the sum-of-residual-squares R^2: Let f_i be the cumulative failures for the ith data point, and $\hat{f}_i$ be the predicted cumulative failures by the fitted model. Then the goodness-of-fit measure R^2 can be calculated as:

$$R^2 = \sum_i (f_i - \hat{f}_i)^2$$

The smaller the calculated R^2, the better the model fit.

Table 5 shows the R^2 values for the fitted models for the last 8 weeks of testing for product E. At the end of testing, the fitted DCRM1 has $R^2 = 566$; while for the GO and MO model fitted to the same data, the R^2 values are 26925 and 50794 respectively. In general, R^2 values for DCRM1 are significantly smaller than those for SRGMs fitted to the raw data. However, the difference in R^2 values between SRGMs fitted to raw data and those fitted to the grouped data are similar, with a typical difference of about 5%. Similar patterns in R^2 values are also observed in product D.

Table 6. Reliability assessment (failure rate $\times 10^{-8}$) comparison

week	$F-7$	$F-6$	$F-5$	$F-4$	$F-3$	$F-2$	$F-1$	F
DCRM1	1.545	2.025	7.069	11.86	0	1.055	1.312	1.142
(n)	(30)	(56)	(26)	(18)	(6)	(30)	(50)	(58)
GO (raw)	22.05	14.89	9.789	6.493	2.650	1.718	.9914	.7983
DCRM2.GO	20.62	10.97	8.330	6.375	2.786	1.895	1.113	.8784
MO (raw)	22.59	16.77	12.67	9.852	6.488	5.165	4.191	3.863
DCRM2.MO	21.30	13.29	11.11	9.476	6.439	5.156	3.991	3.608

R^2 values for DCRM1 over different data sets and time intervals are also more stable as compared to those for fitted SRGMs. Because of the lack of an assumed overall functional form in DCRM1, any pattern can be fitted fairly closely by DCRM1. This is in sharp contract to SRGMs, where a general reliability growth pattern with gradual flattening is expected. Otherwise, the SRGMs won't fit, or fit badly, with huge R^2 values.

In deriving our DCRM1, a constant failure intensity is assumed for each cluster of test runs associated with a variable-time window to isolate short-term dependencies within the data cluster. This assumption is statistically validated by the close conformance between the actual failure intensities and the predicted failure intensities for different time segments, as indicated by the small R^2 values for our DCRM1 in Table 5.

4.4 Reliability assessment comparison

With DCRM1, the current reliability can be assessed to be the predicted reliability for the last segment. This kind of assessments can be provided by DCRM1 consistently from the very beginning of testing to the end. For the last 8 weeks of testing for product E, the current reliability can also be provided by the estimated failure rates by the fitted DCRM2 or by SRGMs fitted to the raw data. The reliability assessments from all these five alternatives (DCRM1, GO, DCRM2.GO, MO, and DCRM2.MO) are given in Table 6. These reliability assessments generally converge to similar values towards the end of testing. Assessments provided by DCRM2 are also more stable than those by its counterpart SRGM fitted to raw data. The same observations are also true for product D.

Notice that for DCRM1, the size of the last segment, n, or the number of test runs, is also given in Table 6, so that the reliability assessment results can be interpreted accordingly. In general, a larger n gives us a higher confidence in assessment result than a smaller n.

The estimated reliability for product E by its DCRM1 at product release can be characterized by its failure rate of 1.142e-08 failures per transaction from Table 6, which is in between those estimated by GO models (7.983e-09 by raw GO and 8.784e-09 by DCRM2.GO) and MO models (3.863e-08 by raw MO and 3.608e-08 by DCRM2.MO). This and other results from Table 6 demonstrate that DCRM1 can give us a fairly accurate assessment of product reliability during testing, provided that enough data points (observations) are included in the last segment. More importantly, in the early part of testing, e.g., before week $F-7$ for product E, DCRM1 is the only model that can provide any meaningful reliability assessments among the five alternatives, because no SRGMs can be fitted to these early data sets.

4.5 Prediction accuracy comparison

Prediction accuracy can be evaluated by using only part of the data as the training set and the rest as the testing set. For example, in product E we have identified various cutoff points in Table 4, the last 8 weeks of testing where consistent SRGMs can be fitted to cumulative observations. We can use models fitted to data accumulated up to week W_i to predict additional failures by week W_j for any $j \geq i$, and compare these predictions to the actual observations.

Let f_{ij} be the number of actual failures observed between weeks W_i and W_j; and $\hat{f}_{ij}$ be the number of failures for the same period predicted by a fitted SRGM. Then the prediction error between the two, e_{ij}, is given by $e_{ij} = \hat{f}_{ij} - f_{ij}$. The collection of these prediction accuracy evaluation results e_{ij}'s can be summarized in a upper triangular table, such as in Table 7 for DCRM2.GO for product E. Similar prediction accuracy results were also calculated for the GO and MO models fitted to the raw data, as well as for DCRM2.MO.

To better compare the prediction accuracy and present the results in a more easily interpretable form, we produced various graphs for the model pairs. Fig. 3 compares predicted failures between model weeks W_i, $F-7 \leq i \leq F$ and the final week F for DCRM2.GO and GO. Fig. 3a (left figure) is a comparison of absolute accuracy, plotting side-by-side the prediction er-

Table 7. Error in predicted additional failures by DCRM2.GO

model	prediction week (W_2)							
week (W_1)	$F-7$	$F-6$	$F-5$	$F-4$	$F-3$	$F-2$	$F-1$	F
$F-7$	0	16.7	30.1	45.8	89.6	103.6	125.5	133.4
$F-6$	NA	0	7.2	14.7	31.2	35.3	40.5	42.1
$F-5$	NA	NA	0	6.57	20.3	23.5	27.4	28.6
$F-4$	NA	NA	NA	0	12.7	15.6	19.1	20.2
$F-3$	NA	NA	NA	NA	0	2.46	5.42	6.27
$F-2$	NA	NA	NA	NA	NA	0	2.60	3.33
$F-1$	NA	NA	NA	NA	NA	NA	0	0.68
F	NA	NA	NA	NA	NA	NA	NA	0

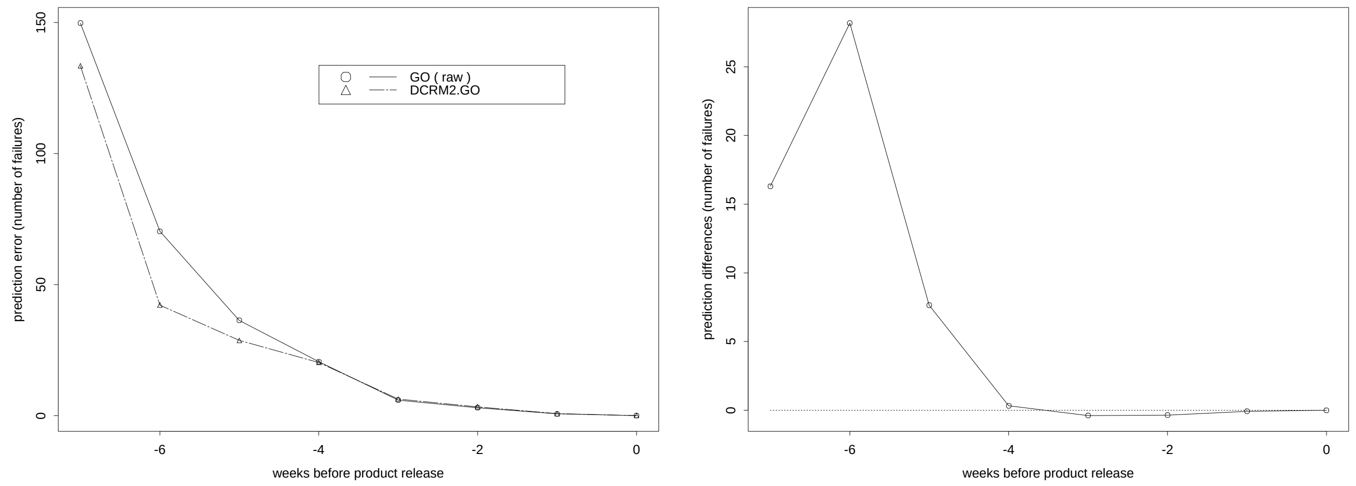


Figure 3. a. absolute prediction error (left) and b. relative prediction error (right) for DCRM2.GO & GO

rors by the two different types of models, e_{iF}(GO) and e_{iF}(DCRM2.GO). Fig. 3b (right figure) is a comparison of relative accuracy, plotting the difference in absolute prediction errors by the two types of models, i.e., it shows the values of $|e_{iF}(\text{GO})| - |e_{iF}(\text{DCRM.GO})|$ over different i values from $F-7$ to F.

As can be clearly seen from Fig. 3, DCRM2.GO consistently performs better or equally to GO. In general, comparing predictions by DCRM2.GO fitted to the data grouped by DCRM1 against that of GO fitted to the raw data, and comparing DCRM2.MO to MO, showed that DCRM2 consistently performed better or equally to it's counterpart using the same SRGM on the raw data. The same observation is also true for product D. However, since the raw data for product D are already grouped by testing days, the data grouping effect according to our data clusters is not as pronounced as in product E.

As mentioned before, reliability predictions in our approach are primarily performed by DCRM2, although DCRM1 can also be used for some short-term predictions by extending the last segment into the future. To compare the prediction accuracy of this latter usage, we also produced short-term predictions for individual runs (for product E) or individual testing days (for product D) in the week immediately following the modeling week, and compared these predictions to that by DCRM2 and by SRGMs fitted to raw data. The prediction accuracy of these models in such short-term predictions are roughly comparable, but no model consistently outperforms others. However, in the early part of testing, e.g., before week $F-7$ for product E, DCRM1 is the only model available to provide reliability predictions. Therefore, this usage of extending the last segment of DCRM1 as future reliability predictions still play an important role in reliability predictions.

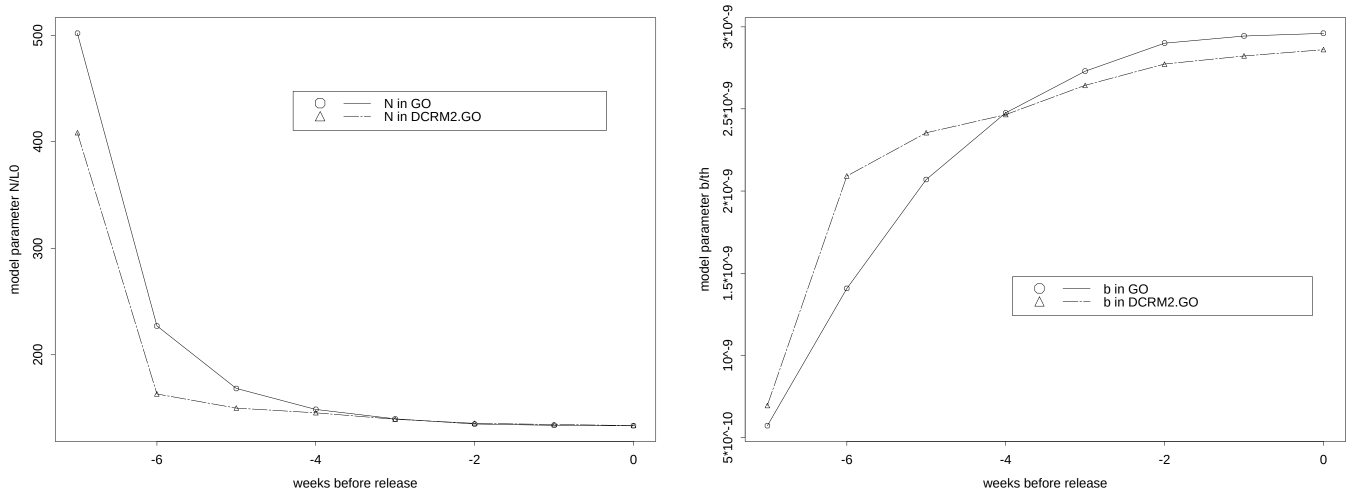


Figure 4. Model parameters N (left) and b (right) for DCRM2.GO and GO over time

4.6 Model stability comparison

It is desirable for reliability models to have good stability, so that consistent modeling results over time can be obtained and used to track reliability progress and to avoid erratic actions. For various SRGMs, the model stability can be reflected by the stability of their parameters estimated from data. This is particularly true when the estimated parameters have some physical meaning, such as N in the GO model, indicating the estimated total number of defects in the system under testing.

Fig. 4 plots the parameters N (left plot) and b (right plot) in the GO and DCRM2.GO models over the last 8 weeks of testing for product E. The estimated parameters for the GO model are based on the raw data; while that for the DCRM2.GO is based on the data clusters or grouped data. Fig. 4 clearly demonstrates the better stability for our DCRM2 as compared to corresponding SRGM based on the raw data. For product D, the same observation holds for the parameter N, but for parameter b, the stabilities are about the same. The stability of parameters for other models (MO and and DCRM2.MO for products E and D) also shows similar trend: Those based on grouped data are either more stable or equally stable as those based on the raw data.

4.7 Model sensitivity to segment size and data weight

We also examined other factors and their effect on model performance. In our DCRM1, T_s represents a lower bound on the length of the partitioned segments that can be selected to reflect short-term dependencies. Any data subset S of size $|S| < T_s$ will not be partitioned further. Usually short-term dependencies last over several test runs, thus T_s values of less than 10 would not make too much sense. Therefore, we selected T_s at 10, 20, and 40 for product E and compared the modeling results. Lower T_s yields more data clusters in DCRM1, and can be better used to analyze local variations. However, for DCRM2 based on these different data grouping thresholds, almost identical performance results were obtained. These performance results never differ by more than a few percentage points (typically within 5% of each other), well below the difference between these models and SRGMs fitted to the raw data. Consequently, we recommend using $T_s = 10$ to allow for a good combination of good performance results for DCRM2 and effective usage of DCRM1 to analyze local variations. In fact, all the results presented so far are for the DCRMs with $T_s = 10$, except Fig. 1 and Table 3, where $T_s = 40$ is used to illustrate the major data clusters and the concept of DCRMs without having to show too many data segments.

In DCRM2, SRGMs fitted to grouped data typically require the use of data weights, because different segments usually cover vastly different time periods, numbers of runs, or numbers of failures. For example, the last segment in Fig. 1 is significantly longer (in usage time or transactions) and represents more observations (58 runs, see Table 3) than many segments near the beginning (e.g., segments 6 and 9 with 6 runs each). Several possible data cluster weighting schemes can be considered:

- *No weights.* Each data cluster is used as an unweighted data point, resulting in larger clusters being severely under-represented. For the products we studied, this scheme resulted in models that perform even worse than models fitted to the raw data.

- *Point weights.* Each data cluster is weighted by the number of individual data points it represents. This is the weighting scheme we used for all our DCRM2 models discussed so far, which resulted in consistently good modeling results.
- *Time interval weights.* Each data cluster is weighted by the length of the time interval it covers. This scheme typically produces similar results to the point weighting scheme. But the results are not as stable, because some test runs with substantially longer usage time and the corresponding data clusters can overwhelm other data clusters.

Based on these results, we recommend point weights to be used with our DCRM2.

5 Conclusions and Perspectives

Short-term dependency and lack of long-term dependency among test runs and related failure data are commonly observed in large software systems. In this paper, we developed our data cluster based reliability models (DCRMs) to deal with reliability analysis under such environments. We grouped test runs and associated failures into clusters of homogeneous failure intensities to reflect this short-term dependency and to isolate this dependency within the cluster. These data clusters form a piecewise linear model, our DCRM1, that can be directly used to assess product reliability and identify problematic areas. Our DCRM2 fits existing software reliability growth models (SRGMs) to these data clusters as grouped data, to provide long-term reliability assessments and predictions.

The grouping of data in our DCRM1 is automatically done using the tree-based modeling technique supported by a commercial tool S-PLUS and our utility programs. Our DCRM1 fits actual failure data better than traditional SRGMs and provides modeling results that can help us evaluate product reliability and identify anomalies throughout the testing process, especially during the early part of testing before SRGMs can be fitted to testing data. Our DCRM2 complements our DCRM1 by providing results about reliability growth and overall reliability based on the complete testing data grouped by our DCRM1. For the IBM products we studied, the fitted DCRM2 consistently outperformed corresponding SRGMs fitted to the raw data in prediction accuracy and model stability.

There are many research issues that we would like to address in the near future: A detailed comparative study is conducted in parallel with this study, comparing our DCRMs to other data grouping methods commonly used in practical applications. The preliminary results indicate that our approach performs better than other data grouping methods in most cases. Further work in tool support can make it easier to deploy our approach in diverse industrial environments. Integration of this research with our previous work on tree-based reliability models [18] could lead to a more effective way to measure and improve reliability. Once proven effective and perfected in these followup studies, our approach can be deployed in industry to effectively analyze and improve reliability for many large software systems.

Acknowledgment

This work is supported in part by NSF/CAREER award CCR-9733588, IBM Toronto Laboratory, and THECB/ATP awards 003613-0030-1999 and 003613-0030-2001.

We thank the anonymous reviewers for their constructive comments and suggestions.

Appendix: Tree-Based Modeling

Tree-based modeling is a statistical analysis technique that attempts to establish predictive relations through recursive partitioning [3]. In tree-based models, modeling results are represented in tree structures. Each node in a tree represents a set of data, which is recursively partitioned into smaller subsets. The data used in such models consist of multiple attributes, with one attribute identified as the *response* variable and several other attributes identified as *predictor* variables. (However, only one predictor variable, usage time τ, is used in this paper.) Recursive partitioning minimizes the difference between predicted response values and the observed response values. The specific tree construction algorithm supported by S-PLUS and used in this paper is summarized below:

0. *Initialization.* Initialize a list, `Slist`, for the data sets to be partitioned, with the complete data set as the singleton element. Select the size and homogeneity thresholds T_s and T_h for the algorithm.
1. *Overall control.* Repeatedly remove a data set from `Slist` and execute step 2 until `Slist` becomes empty.
2. *Size test.* If $|S| < T_s$, stop; otherwise, execute steps 3 through 6. $|S|$ is the number of data points in set S.

3. *Defining binary partitions.* A binary partition divides S into two subsets using a *split condition* defined on a specific predictor p with a cutoff value c. Data points with $p < c$ form one subset (S_1) and those with $p \geq c$ form another subset (S_2).

4. *Computing predicted responses and prediction deviances.* The predicted response value $v(S)$ for a set S is the average over the set; i.e., $v(S) = \frac{1}{|S|} \sum_{i \in S}(v_i)$; and the prediction deviance is $D(S) = \sum_{i \in S} (v_i - v(S))^2$, where v_i is the response value for data point i.

5. *Selecting the optimal partition.* Among all the possible partitions (all predictors with all associated cutoffs), the one that minimizes the deviance of the partitioned subsets is selected; i.e., the partition with minimized $D(S_1) + D(S_2)$ is selected.

6. *Homogeneity test:* Stop if this partitioning cannot improve prediction accuracy beyond a threshold T_h, i.e., stop if $\left(1 - \frac{D(S_1)+D(S_2)}{D(S)}\right) \leq T_h$; otherwise, append S_1 and S_2 to *Slist*.

References

[1] J. R. Brown and M. Lipow. Testing for software reliability. In *Proc. Int. Conf. Reliable Software*, pages 518–527, Los Sangeles, CA, Apr. 1975.

[2] K.-Y. Cai. Censored software-reliability models. *IEEE Trans. on Reliability*, 46(1):69–75, Mar. 1997.

[3] L. A. Clark and D. Pregibon. Tree based models. In J. M. Chambers and T. J. Hastie, editors, *Statistical Models in S*, chapter 9, pages 377–419. Chapman & Hall, London, 1993.

[4] K. Gočeva-Popstojanova and K. S. Trivedi. Failure correlation in software reliability models. *IEEE Trans. on Reliability*, 49(1):37–48, Mar. 2000.

[5] A. L. Goel. Software reliability models: Assumptions, limitations, and applicability. *IEEE Trans. on Software Engineering*, 11(12):1411–1423, Dec. 1985.

[6] A. L. Goel and K. Okumoto. A time dependent error detection rate model for software reliability and other performance measures. *IEEE Trans. on Reliability*, 28(3):206–211, 1979.

[7] Z. Jelinski and P. L. Moranda. Software reliability research. In W. Freiberger, editor, *Statistical Computer Performance Evaluation*, pages 365–484. Academic Press, New York, 1972.

[8] K. Kanoun and J.-C. Laprie. Trend analysis. In M. R. Lyu, editor, *Handbook of Software Reliability Engineering*, pages 401–437. McGraw-Hill, New York, 1995.

[9] P. A. Keiller and T. Mazzuchi. Enhancing the predictive performance of the Littlewood non-homogeneous Poisson process software reliability growth model using data censoring techniques. In *Proc. 10th Int. Symp. on Software Reliability Engineering*, Boca Raton, Florida, Nov. 1999.

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

[11] M. R. Lyu, editor. *Handbook of Software Reliability Engineering.* McGraw-Hill, New York, 1995.

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

[13] J. D. Musa and K. Okumoto. A logarithmic Poisson execution time model for software reliability measurement. In *Proc. 7th Int. Conf. on Software Engineering*, pages 230–238, Orlando, FL, Mar. 1984.

[14] E. Nelson. Estimating software reliability from test data. *Microelectronics and Reliability*, 17(1):67–73, 1978.

[15] N. F. Schneidewind. Analysis of error processes in computer software. In *Proc. Int. Conf. Reliable Software*, pages 337–346, Los Angeles, California, Apr. 1975.

[16] N. F. Schneidewind. Software reliability model with optimal selection of failure data. *IEEE Trans. on Software Engineering*, 19(11):1095–1104, Nov. 1993.

[17] N. Singpurwalla. Software reliability modeling by concatenating failure rates. In *Proc. 9th Int. Symp. on Software Reliability Engineering*, pages 106–110, Nov. 1998.

[18] J. Tian. Integrating time domain and input domain analyses of software reliability using tree-based models. *IEEE Trans. on Software Engineering*, 21(12):945–958, Dec. 1995.

[19] J. Tian, P. Lu, and J. Palma. Test execution based reliability measurement and modeling for large commercial software. *IEEE Trans. on Software Engineering*, 21(5):405–414, May 1995.

[20] J. Tian and J. Palma. Test workload measurement and reliability analysis for large commercial software systems. *Annals of Software Engineering*, 4:201–222, Aug. 1997.

[21] W. N. Venables and B. D. Ripley. Multivariate analysis. In *Modern Applied Statistics with S-Plus*, chapter 12, pages 301–328. Springer-Verlag, New York, 1994.

[22] S. Yamada, M. Ohba, and S. Osaki. S-shaped reliability growth modeling for software error detection. *IEEE Trans. on Reliability*, 32(5):475–478, Dec. 1983.