An Experiment in Estimating Reliability Growth Under Both Representative and Directed Testing

Brian Mitchell and Steven J. Zeil

Old Dominion University
Department of Computer Science
Norfolk, VA 23529-0162

mitch_c@cs.odu.edu, zeil@cs.odu.edu

Abstract

The Order Statistic model of reliability growth offers improved

- experimental design, and
- flexibility in testing methodology

compared to conventional reliability growth models. It permits prediction of operational reliability without requiring that testing be conducted according to the operation profile of the program input space.

This paper presents the first experimental use of the Order Statistic model under a test plan that combines both representative and directed tests. Results suggest that this is an effective way to obtain quantified measures of test quality without abandoning the advantages of directed test methods.

1 Introduction

There is a substantial body of literature devoted to *directed* testing methods, which manipulate the choice of test inputs so as to increase the probability and/or rate of fault detection. These include most well-known testing methods, including functional and structural testing, data flow coverage, mutation analysis, and domain testing[1, 15]. Historically, a difficulty affecting the deployment of these methods has been the lack of any quantified measure of test effectiveness with external referents (i.e., that is not based upon properties defined by the criterion itself).

In contrast, a variety of reliability growth models provide quantified measures of test effectiveness in terms that are directly relevant to project management [8, 10, 14], but at the cost of restricting testing to *representative* selection, in which test data is chosen to reflect the operational distribution of the program's inputs. During testing, data is collected on the observed times between program failures (or, similarly, numbers of failures within a time interval). These observations are fitted to one of various models, which can then be used to estimate the current reliability of the program.

In [11, 13], we present an Order Statistic model of reliability growth. This model can employ an arbitrary mixture of *program* failure rate data, as in conventional reliability growth models with *fault* failure rate information obtained via post-mortem analysis of the debugged faults. The primary advantages of this model are:

- Test planners regain the flexibility to employ their best testing practices, whether those involve directed testing, representative testing, or a mixture of the two. The choice of testing method is no longer solely determined by the desire to predict operational reliability.
- More robust experimental designs can be formulated by taking advantage of a wider variety of options for data collection.

In [11], we introduced the Order Statistic model, and presented an experiment in which it was employed on purely representative tests, using program failure (interfailure time) data. The Order Statistic model was shown to be competitive with conventional reliability growth models "on their own turf". This result was reassuring, but the experimental design deliberately did not take advantage of the Order Statistic model's flexibility, as doing so would have prevented comparison with the other models.

In [13], we presented an experiment in which the Order Statistic model was employed with fault failure rate data obtained during directed testing. Results of this experiment were mixed. The Order Statistic model fitted well to the observed data, the model proved useful in related data analysis such as the identification of outliers in the observations. But the final predictions of program reliability were unacceptably optimistic.

We speculate that a major reason for the poor prediction lies in what we term an "integration effect". We were attempting to predict program failure rate based entirely upon observations of the failure rates of individual faults. The program failure is (approximately) the sum of the failure rates of the undetected faults. We might make an analogy between this prediction procedure and an attempt to predict the value of an unknown function given information about its first derivative. Although the shape of the function can be obtained via integration, the precise values can be offset by an arbitrary constant. Similarly, we believe that exclusive use of fault failure rate data in our model allows the estimate of program failure rate to "drift" up or down in response to minor fluctuations in the set of observations.

The obvious response to this possibility is to combine observations of program failure rate with observations of fault failure rates. Because program failure rate information is easily obtained during

This work was supported by grant NAG-1-439 from the NASA Langley Research Center and grant CCR-9312386 from the National Science Foundation.

ISSTA 98 Clearwater Beach Florida USA
Copyright 1998 0-89791-971-8/98/ 03..$5.00

representative testing, such a combination suggests a test plan involving both representative and directed testing, which we believe has many advantages from a methodological viewpoint anyway.

This paper presents an experiment using the Order Statistic model with a mixture of representative and directed testing. Section 2 reviews the Order Statistic model. Section 3 describes our experimental design, and Section 4 presents the results and our analysis.

2 The Model

2.1 Assumptions

The Order Statistic model is based upon the following assumptions:

1. The fault detection rate remains constant in the interval between corrections.
2. Faults are corrected "perfectly", without introducing new faults into the software.
3. The testing process is biased, tending to find faults with higher failure rates earlier.
4. Faults in a program have failure rates ϕ distributed with density $f(\phi)$ and cumulative distribution $F(\phi)$.
5. Failures are independent.

These assumptions are equivalent to or, in some cases, considerably relaxed from those imposed by conventional reliability growth models [5].

Assumption 1 is common to most reliability growth models, and leads to the use of the exponential distribution to describe failure times after each repair.

Assumption 2 in many models takes the stronger form:

2′ Faults are corrected perfectly and instantaneously.

The Order Statistic model is not coupled to a particular test and debug process, and so does not require instantaneous repair. It does assume perfect correction, a common requirement, though not a universal one.

Assumption 3 replaces the more common assumption that

3′ Test selection is performed in a manner consistent with the distribution of inputs under eventual operation.

This assumption generally regarded as central to reliability growth modeling [5], and is used to conclude that

3a′ Faults are found in decreasing order of failure rate (except in the Jelinski-Moranda and related models that assume, even less realistically, that all faults have identical failure rates.), and

3b′ The rate of failure during testing is identical to the rate of failure under operation.

But 3a′ is simply the most likely among many possible detection orders, and deviations from this order are almost certain in practice. Our weaker assumption 3 leads to the same ranking of probable detection orders. Furthermore, the use of order statistics allow the Order Statistic model to correct many deviations from the most probable ordering and to detect many serious deviations that cannot be corrected.

3b′ leads to the exclusive use of time to first failure as the observable quantity during reliability modeling. But time to first failure data is inherently noisy (the standard deviation of each observation is equal to its expected value). The Order Statistic model can be used with a variety of observables that provide estimates of fault or program failure rates.

Assumption 4 simply indicates that the fault failure rates follow some distribution. In practice, applying the Order Statistic model currently requires a further assumption as to the particular distribution involved. To date, we have employed the log-normal distribution because of its flexibility and its affinity for data sets that span many orders of magnitude. This can be compared to Littlewood's choice of the gamma distribution, for its similar flexibility [9] and can be contrasted with several models that assume that all fault failure rates are equal [5].

Assumption 5 is also nearly, if not completely, universal among reliability growth models. Although interactions among faults are known in practice, there is surprisingly little data indicating whether these interactions are statistically significant. Hoppa and Wilson have examined these interactions, but do not isolate their effects from differences in the fault detection order [6].

2.2 Definitions

In the process of developing a software artifact, there are many potential faults that could be injected into the code. Given a set Z of potential faults, let the operational failure rates, ϕ, associated with the faults in Z be distributed with density $f(\phi)$ and cumulative distribution $F(\phi)$. As the software is written, some of the faults from Z are inserted into the code. We model this as a selection of n faults from the set Z.

At any point in the test process, we will have detected and removed zero or more of these faults. Let $[\phi_i]_{i=1}^n$ be the *debugging sequence*, the sequence of operational fault failure rates in the order in which the corresponding faults have been (or will be) repaired.[1]

Fault failure rates and program failure rates (λ) are related. Assuming independent behavior by different faults, then after repairing k faults, we have:

$$\lambda_k = 1 - \prod_{i=k+1}^{n} (1-\phi_i)$$

If the fault failure rates are small,

$$\lambda_k \approx \sum_{i=k+1}^{n} \phi_i$$

From λ_k we can easily obtain estimates of the time to next failure and of the program reliability. Unfortunately, this is not a particularly useful expression of λ_k because it depends upon the failure rates of the faults that remain undetected. We therefore move to consider the derivation of alternate estimators for λ_k.

2.3 Order Statistics

The debugging sequence $[\phi_i]_{i=1}^n$ is ordered according to the order in which faults are repaired. Consider the possibility of sorting these

[1] It is actually only essential that the observed faults be so ordered, but it is convenient to extrapolate this sequence so that we can describe our progress at any time in terms of the number of faults detected.

failure rates into ascending order by value.[2] Let $[\phi_{i:n}]_{i=1}^{n}$ denote the sequence obtained by sorting the debugging sequence into ascending order. $\phi_{i:n}$ is called the i^{th} *order statistic* of the debugging sequence for a sample of size n.

The distribution of the $\phi_{r:n}$ is characterized by the density function [3]

$$f_{r:n}(\phi) = r\binom{n}{r} F^{r-1}(\phi) f(\phi)(1-F(\phi))^{n-r} \quad (1)$$

with the cumulative distribution

$$F_{r:n}(\phi) = \sum_{i=r}^{n} \binom{n}{i} F^{i}(\phi)(1-F(\phi))^{n-i} \quad (2)$$

After repair of k faults, we can therefore predict the failure rate of the next fault as $E(\phi_{n-k:n})$, and can estimate the current program failure rate as

$$\lambda_k = 1 - \prod_{i=k+1}^{n} (1 - E(\phi_{n-i:n}))$$

or, if the remaining fault failure rates, are small,

$$\lambda_k = \sum_{i=k+1}^{n} E(\phi_{n-i:n})$$

2.4 A Debugging-Based Modeling Process

We advocate a reliability estimation process in which

1. Programs are tested using mixture of representative and directed tests. The choice of mixture would be made largely for economic and test quality reasons, rather than simply to facilitate reliability modeling.
2. As repairs are made, data is collected on λ's and/or ϕ's, as appropriate.
3. Periodically, the λ and ϕ data are fitted to the ordered statistic reliability growth model.

Inputs to the Order Statistic model fitting process consist of estimates of program failure rates after the repair of k faults and/or estimates of the failure rates of individual faults.

These estimates can be performed in a variety of ways as described in [11, 13].

2.5 Applicability

2.5.1 *Representative Testing*

Representative testing clearly satisfies our assumption that the testing process by biased by failure rate. Conventional reliability growth models have assumed that the debugging sequence (the order in which faults are found) occurs in decreasing order of failure rate. In practice, exact adherence to this order is unlikely. Hoppa and Wilson have argued that many models are extremely sensitive to permutations in the debugging sequence [6].

Under the Order Statistic model, however, even if faults are found out of order, the explicit sorting of ϕ's eventually corrects this permutation. (If only λ data has been collected, then we are no better off nor worse than conventional models.)

[2]Descending order would be more appropriate for a testing scenario, but the traditional definition of order statistics employs ascending order.

2.5.2 *Directed Testing*

Advocates of representative testing have frequently assumed that directed testing finds faults in an arbitrary order unrelated to their operational failure rates [2, 7]. We are unaware of any experimental support for this assumption, and, in [11], we discuss a number of reasons for regarding this assumption with skepticism. The Order Statistic model can actually be applied without invoking the assumption of a biased test process, though we do not expect the results to be as useful in practice [11].

We believe it far more reasonable that some correlation will exist between the order in which faults are detected during directed testing and the operational failure rates of those faults. We propose the following:

Ordered Testing Property: For a given directed testing method, as we approach coverage of the method, the set of k faults revealed will be approximately the k faults with the largest individual operational failure rates.

This property allows for the possibility that testing to some coverage criterion may indeed find faults in unpredictable orders (e.g., depending upon the order in which the testers choose to cover statements, branches, data flow interactions, or whatever). But the aggregate set of faults found under a criteria should exhibit the desired bias toward more frequently occurring faults.

In [11] we report on one project for which the hypothesis that the orders of fault detection under directed and representative testing are uncorrelated was rejected with a statistical confidence of 99.7%.

3 Experimental Design

3.1 The Data Set

The basis for this study was a data set obtained in prior research conducted by Wild, Zeil, et al.[16], where the authors tested the Launch-Intercept-Control (LIC) software system [4] using both representative methods and a particular directed testing method known as Knowledge-Driven Functional Testing (KDFT), a technique that combines expert information about the software domain with conventional functional testing.

In the original experiment, 100,000 tests were generated according to the operational input distribution and the number of occurrences of each discovered fault was recorded. The resulting operation fault failure rate estimates appear in the "Random" column of Figure 1.

The KDFT technique produces a number of test categories, each of which must be exercised at least k times for a level-k KDFT coverage. 100 test cases were randomly generated for each test category. Again, the number of failures for each fault was observed. The highest failure rate for a given fault in any test category is appears in the "KDFT" column of Figure 1. In this experiement, this rate is used to estimate the probability of a fault's being detected during a 1-test-per-category KDFT coverage. (This estimate, in fact, is biased toward lower rates, because it takes only a single category into account for any fault.)

As noted earlier, most reliability models assume that faults are found in decreasing order of failure rate, but deviations from this sequence are almost certain in practice. To examine the effects that different debugging sequences can have upon reliability estimation, simulated debugging sequences were generated from the failure rate data and the resulting failure rate data was used as input

Fault #	# Failures / # Tests KDFT (100 tests)	Random (10^6 tests)
1.1	0	.000002
3.1	1.0	.000135
3.2	1.0	.000195
3.3	1.0	.000537
3.4	0	.000006
6.1	1.0	.000607
6.2	.89	.000511
6.3	1.0	.000032
7.1	0	.000071
8.1	1.0	.000225
8.2	1.0	.000098
9.1	.71	.000047
9.2	.15	.000006
11.1	.46	.000554
12.1	1.0	.000356
12.2	0	.000071
13.1	0	.000004
14.1	1.0	.001297
14.2	0	.000071
16.1	1.0	.000028
16.2	0	.000034
17.1	0	.000201
17.2	0	.000076
18.1	0	.000008
19.1	1.0	.000264
20.1	1.0	.000323
20.2	.46	.000697
21.1	1.0	.000085
21.2	0	.000007
22.1	1.0	.006551
22.2	1.0	.001735
22.3	1.0	.001735
23.1	1.0	.000072
23.2	0	.000008
24.1	1.0	.000260
25.1	1.0	.000014
25.2	1.0	.000080
25.3	.19	.000003
26.1	1.0	.000140
26.2	1.0	.000009
26.3	0	.000001
26.4	1.0	.000006
26.5	1.0	.000004
26.6	.15	.000368
26.7	.15	.000243

Figure 1: Directed (KDFT) versus Representative [16]

into selected reliability growth models. The process of generating the simulated debugging sequences is described in the next section.

3.2 Mixed Testing Methods

Four simulated debugging sequences for both representative testing and a mixed representative-and-direct approach to testing were generated from the failure rate data of Figure 1. The mixed approach was modeled as using representative methods at the start of testing and switching to directed methods later. Although somewhat unconventional, this plan offers significant advantages especially when very high reliability is sought [11, 13].

The mixed method data was used as input into the Order Statistics model. The representative testing data was used as input into the Jelinski-Moranda model (or the Musa Basic model, which in this treatment is equivalent), and the Musa Log model.

3.3 Generating Representative Debugging Sequences

The generation of debugging sequences from representative testing involved simulating the execution of 100,000 representative test cases. This number of representative test cases is the same number of cases generated and executed in [16]. For each simulated test case in this experiment, the probability of exposing a given fault was dictated by the operational failure rate of that fault.

For example, if given fault in the program had an operational failure rate of .000002, then on each iteration of the simulation (1-100,000), a random number between 1 and 1,000,000 was generated. If the value of this random number was less than or equal to 2 (i.e., $1,000,000 * .000002$), then this fault was considered to be revealed by this simulated test case, and the failure time (iteration number) was recorded.

Simulating 100,000 test cases iterations resulted in a set of system failure times that were converted to interfailure times, and then to program failure rates. The resulting debugging sequences are shown in Figure 2. Note that for most faults there are significant differences between the failure rates obtained by replicated measurement (in the "Random" column of Figure 1) and the rates obtained by the simulated TTFF technique of Figure 2. These differences are to be expected, given the exponential distribution of TTFF, and illustrate our concern about the noise level associated with this, the most common data collection technique for reliability modeling.

3.4 Generating Mixed Debugging Sequences

The process of simulating a mixed approach to testing was somewhat more involved than for purely representative testing.

3.4.1 Determining the Testing Cross-Over Point

Before simulating a mixed method testing process, the point at which simulated testing would switch from representative methods to directed methods had to be identified. The actual determination of this cross-over point would, in practice, be made by testers when they feel that representative testing is starting to"take too long". For this experiment, the cross-over from representative methods to directed methods was made when the first interfailure time exceeding 1000 tests was observed.

Sequence 1		Sequence 2		Sequence 3		Sequence 4	
Fault	**Failure Rate**	**Fault**	**Failure Rate**	**Fault**	**Failure Rate**	**Fault**	**Failure Rate**
12.2	0.3333333	20.2	0.0232558	24.1	0.0500000	22.1	0.0476190
24.1	0.0344828	6.1	0.0312500	22.1	0.0172414	24.1	0.0555556
22.2	0.0322581	14.1	0.0079365	22.2	0.0027933	14.1	0.0121951
22.1	0.0526316	22.2	0.0108696	14.1	0.0588235	22.2	0.0103093
22.3	0.0128205	22.1	0.0208333	3.2	0.0434783	26.6	0.0072464
14.2	0.1250000	26.6	0.0109890	22.3	0.0714286	3.3	0.0555556
20.2	0.0909091	14.2	0.0066225	6.1	0.0035211	6.1	0.0054054
14.1	0.0024390	17.1	0.0086207	12.1	0.2500000	26.7	0.2000000
6.2	0.0476190	24.1	0.0119048	20.2	0.0082645	22.3	0.0147059
8.2	0.0036232	9.1	0.0030581	23.1	0.0031250	16.2	0.0227273
26.6	0.0135135	12.1	0.0833333	17.1	0.0034364	19.1	0.0588235
20.1	0.0136986	21.1	0.0400000	11.1	0.1250000	12.1	0.0011198
3.2	0.0012953	22.3	0.0043668	6.2	0.0022371	17.1	0.0034483
12.1	0.0023041	3.3	0.0038168	3.3	0.0022422	11.1	0.0013699
6.1	0.0344828	25.2	0.0049261	3.1	0.3333333	8.1	0.0072464
3.3	0.0062893	11.1	0.0036232	8.1	0.0106383	23.1	0.0033898
6.3	0.1428571	6.2	0.0029499	26.7	0.0016103	6.2	0.0277778
11.1	0.0039370	26.7	0.0027322	20.1	0.0025510	20.2	0.0078125
3.1	0.0053763	20.1	0.0007849	26.6	0.0021277	8.2	0.0020080
19.1	0.0285714	3.2	0.0003922	26.4	0.0004348	3.1	0.0029762
21.1	0.0046296	19.1	0.0625000	8.2	0.0035587	21.1	0.0029940
8.1	0.3333333	8.1	0.0004666	6.3	0.0013477	20.1	0.0026954
26.1	0.0013245	23.1	0.0081967	7.1	0.0019231	9.1	0.0007225
17.1	0.0015432	26.1	0.0003253	14.2	0.0112360	7.1	0.0009681
7.1	0.0025974	8.2	0.0006262	21.1	0.0005845	25.2	0.0027701
17.2	0.0004836	16.1	0.0009606	19.1	0.0019685	26.1	0.0007862
26.7	0.0015152	7.1	0.0013587	17.2	0.0009615	17.2	0.0016129
23.1	0.0001931	3.1	0.0004953	16.1	0.0019380	3.2	0.0001279
13.1	0.0000548	26.5	0.0000910	26.1	0.0005599	14.2	0.0005299
25.1	0.0003589	12.2	0.0007692	12.2	0.0002147	12.2	0.0058480
9.1	0.0003834	6.3	0.0001166	25.2	0.0003804	26.5	0.0004502
26.2	0.0006345	3.4	0.0001420	9.2	0.0002263	26.4	0.0000615
3.4	0.0001058	17.2	0.0000757	9.1	0.0001856	6.3	0.0000967
16.1	0.0003194	16.2	0.0001149	21.2	0.0001381	26.2	0.0000799
25.3	0.0007133	25.1	0.0000329	25.3	0.0001907	25.1	0.0000845
23.2	0.0000544			16.2	0.0004218	18.1	0.0002861
18.1	0.0045872			13.1	0.0001182		
16.2	0.0000851			23.2	0.0000231		
9.2	0.0000910						

Figure 2: Simulated Debugging Sequences — Representative Testing

	Debugging Sequence			
	1	**2**	**3**	**4**
	Faults Detected	Faults Detected	Faults Detected	Faults Detected
Repr.	12.2 24.1 22.2 22.1 22.3 14.2 20.2 14.1 6.2 8.2 26.6 20.1 3.2 12.1 6.1 3.3 6.3 11.1 3.1 19.1 21.1 8.1 26.1 17.1 7.1 17.2	20.2 6.1 14.1 22.2 22.1 26.6 14.2 17.1 24.1 9.1 12.1 21.1 22.3 3.3 25.2 11.1 6.2 26.7 20.1	24.1 22.1 22.2 14.1 3.2 22.3 6.1 12.1 20.2 23.1 17.1 11.1 6.2 3.3 3.1 8.1 26.7 20.1 26.6 26.4	22.1 24.1 14.1 22.2 26.6 3.3 6.1 26.7 22.3 16.2 19.1 12.1 17.1 11.1 8.1 23.1 6.2 20.2 8.2 3.1 21.1 20.1 9.1
Directed, pass 1	9.2 16.1 23.1 25.1 25.2 25.3 26.2 26.4 26.5	3.1 3.2 6.3 8.1 8.2 16.1 19.1 23.1 25.1 25.3 26.1 26.2 26.4 26.5	6.3 8.2 9.2 16.1 19.1 21.1 25.1 25.2 25.3 26.1 26.2 26.5	3.2 6.3 16.1 25.1 25.2 26.1 26.2 26.4 26.5
pass 2	9.1 26.7	9.2	9.1	
pass 3				25.3
pass 4				
pass 5				9.2

Figure 3: Simulated Debugging Sequences — Mixed Representative and Directed Testing

3.4.2 *Determining Fault Exposure Under Mixed Testing*

For each mixed debugging sequence, the representative failure rate data for the sequence is taken from the corresponding generated representative debugging sequence. Thus, the data in each generated mixed testing set matches the data in the corresponding generated representative testing set up until the crossover point. For example, in the first pair of generated debugging sequences, the first 26 values of the representative data set are identical to the first 26 values of the mixed data set.

After reaching the cross-over point (1000 tests without a failure), the behavior of the software when executing five passes through the refined functional test cases of [16] was simulated. (Although most coverage criteria are satisfied by a single instance of each distinct test class, the KDFT method specifically explored the effects of requiring multiple instances.) Typically, the first pass through the directed test suite usually ended up finding over ninety percent of the faults that were found during the five iterations.

For each pass, the probability of that fault's being revealed is taken from the occurrence rate listed in the "KDFT" column of Figure 1. The resulting debugging sequences are given in Figure 3. This figure presents the list of faults detected, in the order of detection. For the purpose of this experiment, faults found under representative testing are assumed to have been observed with the same program failure rates as in the corresponding debugging sequence of Figure 2. The rate at which faults are detected during directed testing is not used by the Order Statistic model. Faults found under directed testing are associated with fault failure rates from the "Random" column of Figure 1. This is consistent with the use of a focussed, replicated sampling process to determine the fault failure rate, as described in [11, 13]. Note that some directed test passes after the first did not reveal any new faults.

3.5 Fitting

After the data sets had been generated, they were used as input to several reliability models and the results were compared. Maximum likelihood estimators for the Order Statistic model are unknown, so we employ least-squares estimation. For consistency, we chose to employ least-squares for all models in this study.

The fitting error in all cases was computed as

$$e = \sum_{i=1}^{n} \left(\frac{predicted_i - observed_i}{observed_i} \right)^2, \qquad (3)$$

normalizing all errors to treat each prediction with equal weight.

4 Results

The analysis of the models consisted of generating OP (Observed versus Predicted) plots for each model for each data set, comparing the best fits for each model for each complete data set, and comparing the stability of the models using parameter progression plots.

4.1 Predictive Accuracy Of The Models

Figures 4–7 show the OP plots that were obtained for each model for each debugging sequence. Looking at these plots, it is evident that Order Statistics model has fewer points plotted than do the

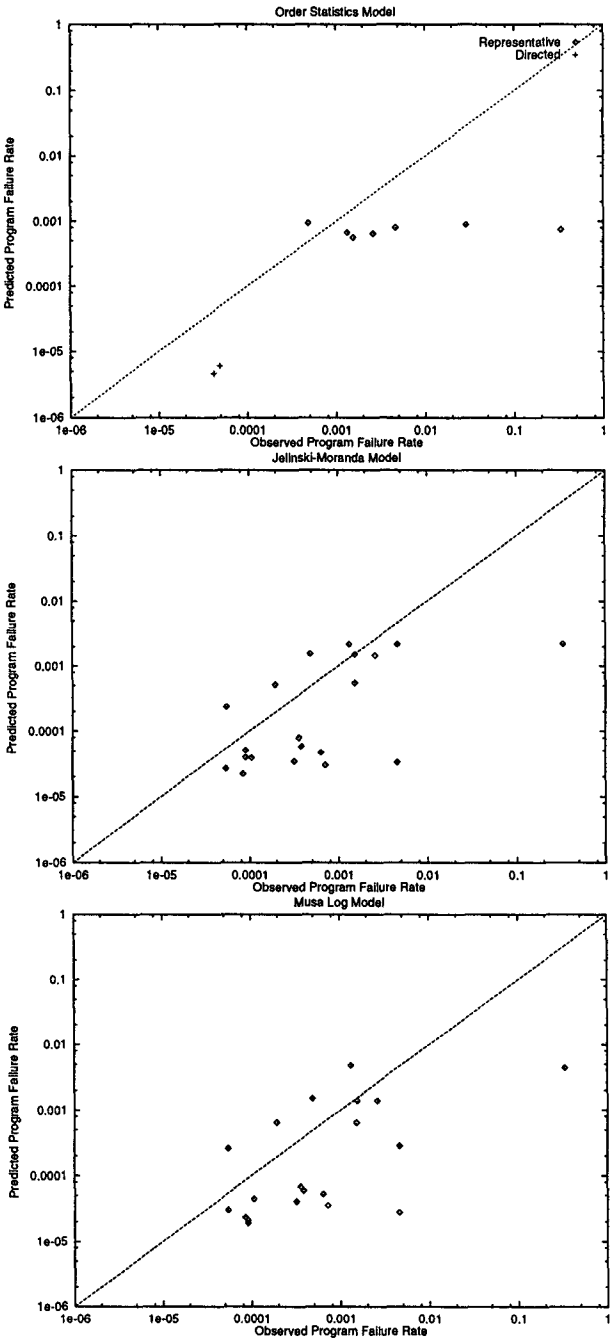

Figure 4: OP Plots For Debugging Sequence One

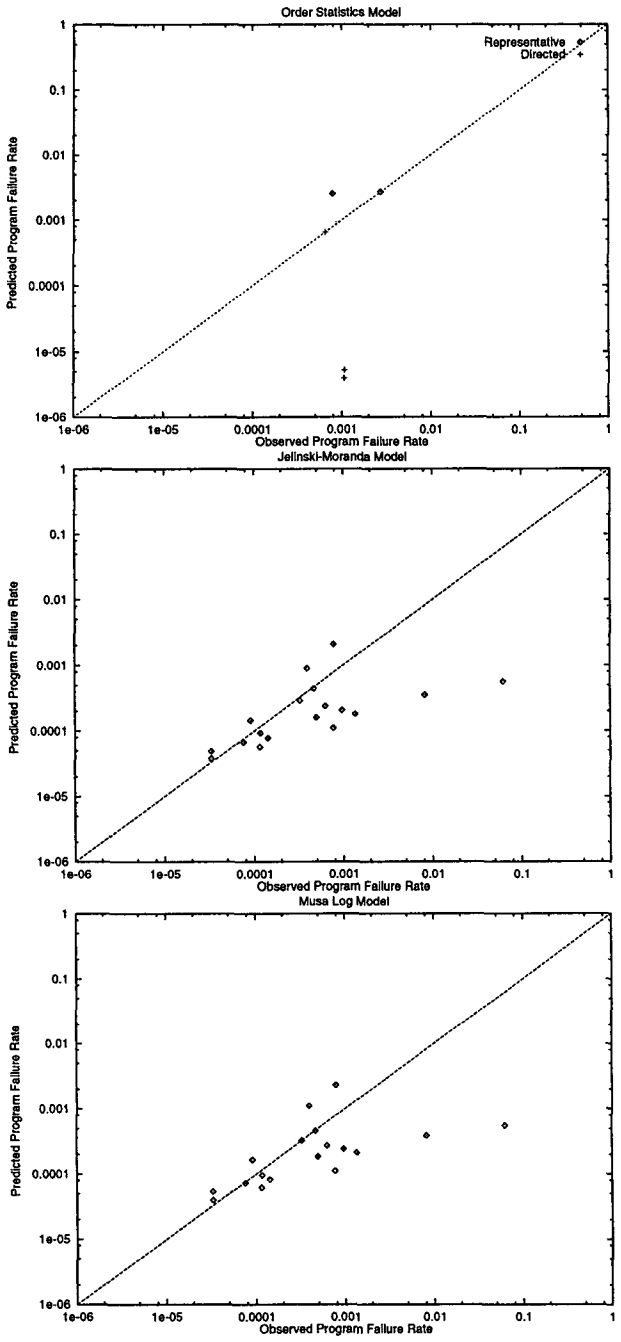

Figure 5: OP Plots For Debugging Sequence Two

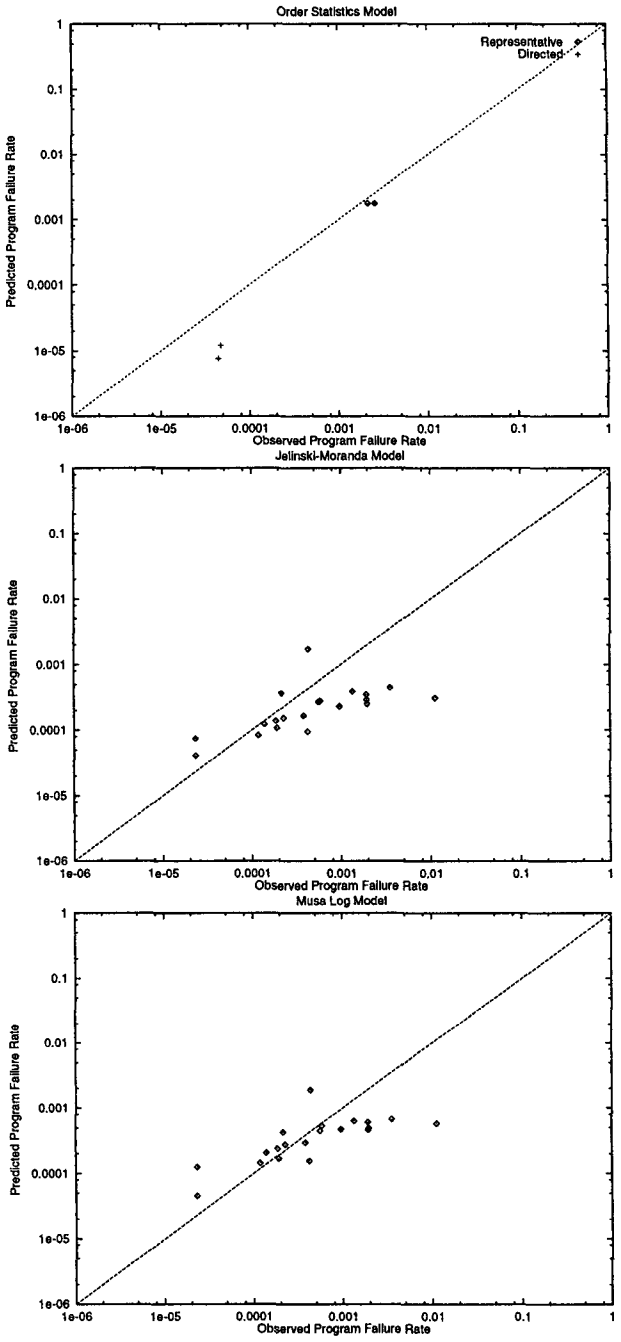

Figure 6: OP Plots For Debugging Sequence Three

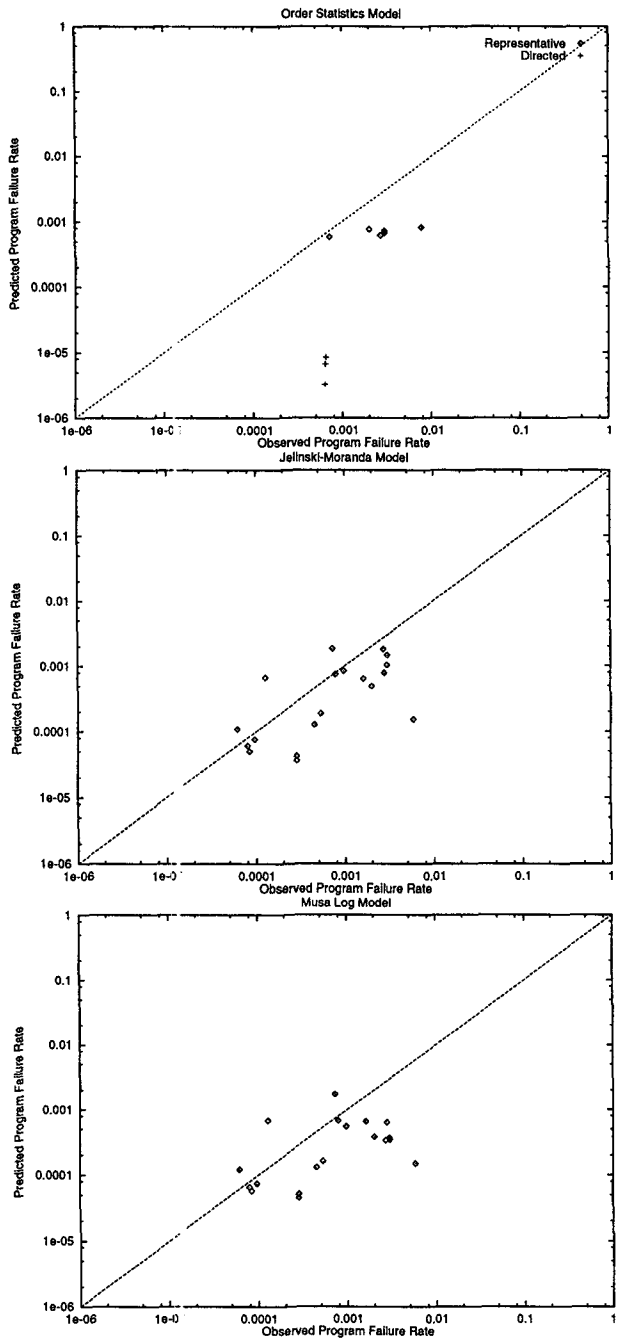

Figure 7: OP Plots For Debugging Sequence Four

Model	JM	ML	OS
Set 1	21.9155	18.43406	15.24542
Set 2	15.91654	8.871714	7.095273
Set 3	15.46484	13.69872	11.33527
Set 4	17.21274	13.63313	10.99466

Figure 8: Average Error For The OP Plots

other models. This difference stems from the Ordered Testing Property. Under purely representative testing, every fault that is found is represented on the OP plot by its program failure rate. The Order Directed Testing Property, however, dictates that predicted program failure rates will only have meaning under directed testing as testing nears coverage. For this reason, instead of plotting 10–15 points corresponding to the faults found under directed testing, only 1–5 points are plotted on the OP plot - one for each prediction made following a directed testing coverage pass that revealed additional faults.

The sum of the squared normalized error (Equation 3) for each model was averaged over the number of predictions. These average errors are shown in Figure 8. It is difficult to tell much difference between the models simply by looking at the plots, but the data in Figure 8 indicates that the Order Statistics model and the Musa Log model perform similarly. Both of these models performed better than the Jelinski-Moranda model.

4.2 Best Fits For Each Model

Figure 9 shows the fit of each model to the full data sets for debugging sequence 1. (Plots for the other debugging sequences are available in [12].)

One feature of note is the discontinuity present in the plots for the Order Statistics model. This discontinuity is caused by the switch from representative testing to directed testing in the mixed method approach to testing. When representative testing is used, the program failure rates are plotted. Once testing switches to directed methods, fault failure rates are plotted. Therefore, the discontinuity in the best fit plot reflects the fact that program failure rates values are generally an order of magnitude (or more) larger than fault failure rate values. Combination of these two different quantities into a single fitting process without biasing the results was accomplished via the normalization of equation (3).

Upon examination of the best fit plots and the data in Figure 10, it is apparent that performance of the Order Statistics model when fitting to the entire data set compares favorably to the other models. For all of the test sets, the error measure for the Order Statistics model is similar to or better than the error measure for the Musa Log model. The error measures for the Order Statistics model and the Musa Log model are much better than the error measures for the Jelinski-Moranda model.

One of the most important observations to be made about the best fit plots is that the Order Statistics model does a very good job of fitting to the fault failure rates present in the mixed method data sets. The fact that the fault failure rates are sorted into descending order by the Order Statistics model, along with the fact that the fault failure rate observations are visibly less noisy than the program failure rate observations seems to improve the goodness of fit.

4.3 Parameter Progressions For Each Model

We examined the progressive values of the fitted model parameters as successive data points were added to each debugging sequence. Detailed plots can be found in [12]. These progressions provide insight into the stability of each model in the face of perturbations to the input.

None of the models exhibited particularly wild swings in parameter values except for the initial stages of sequence 1. The final parameter values of the Jelinski-Moranda and Order Statistic models were largely unaffected by the debugging sequence, but the final parame-

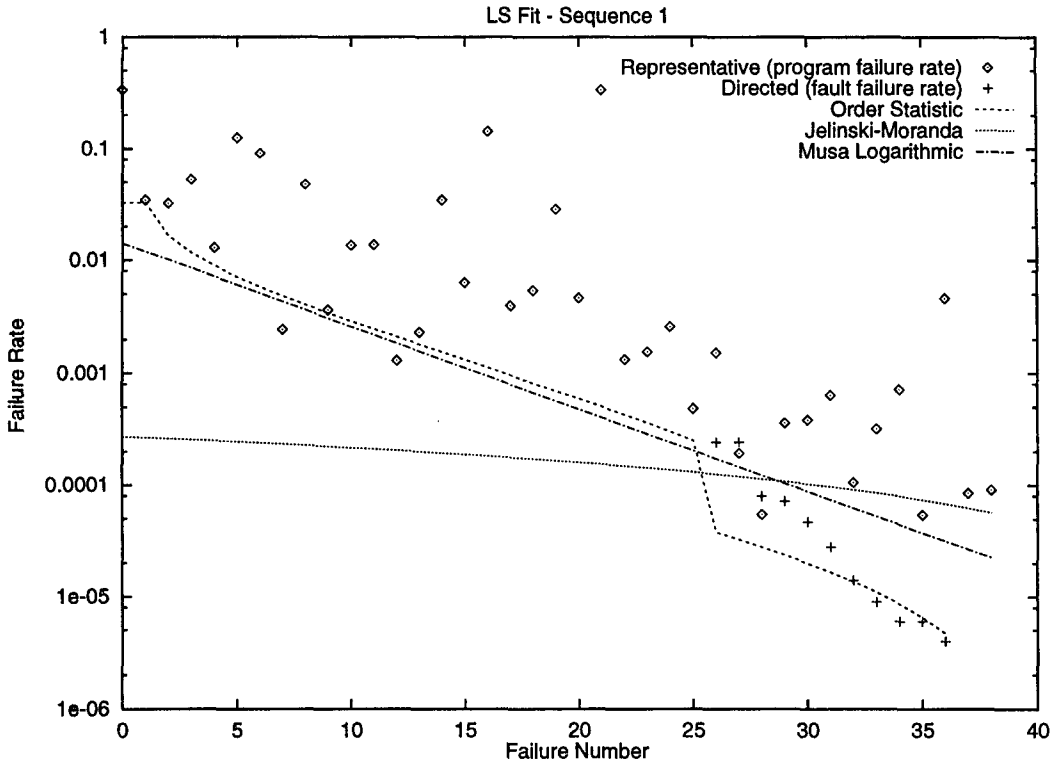

Figure 9: Best Fits For Debugging Sequence 1

ter values for the Musa log model showed more variation, indicating that this model appears more sensitive to permutations in the debugging sequence. Such sensitivity has been reported by Hoppa and Wilson [6], though they found the Jelinski-Moranda model even more sensitive. Factors that may account for this difference are:

- Our study uses debugging sequences that are much more likely to occur in practice.
- Our use of least-squares estimators may be more robust than the numerical procedures for maximum likelihood used by Hoppa and Wilson, which frequently failed to find any fit for some models.

The Jelinski-Moranda and Order Statistic models assume that a program contains a finite number of faults, while the Musa Logarithmic model assumes that the number of faults is infinite. The progression of predicted values for N, the number of faults, under the two finite methods is interesting. A typical progression for Jelinski-Moranda (Figure 11) exhibits thresholding: the estimate for N almost always increases as new data is added.

For the Order statistic model, similar thresholding was seen on debugging sequences 2 and 3, but on the other sequences the Order Statistic model tended to establish a more conservative "plateau" (Figure 12) until late in the process when a series of much smaller failure rates was encountered.

Model	JM	ML	OS
Set 1	30.3649	24.8865	18.738
Set 2	22.0017	11.7156	11.7305
Set 3	24.0289	20.3609	15.9356
Set 4	25.1962	17.5946	13.5606

Figure 10: Error For The Fits To The Full Data Set

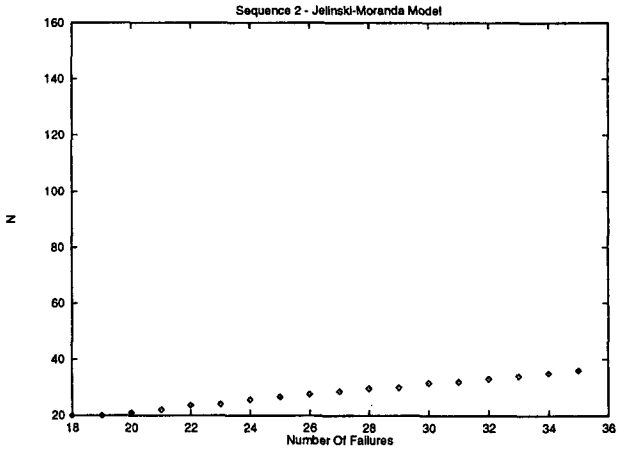

Figure 11: Jelinski-Moranda N Progression

5 Conclusions

The data presented here support the contention that the Order Statistic model is a viable approach when combining directed and representative testing.

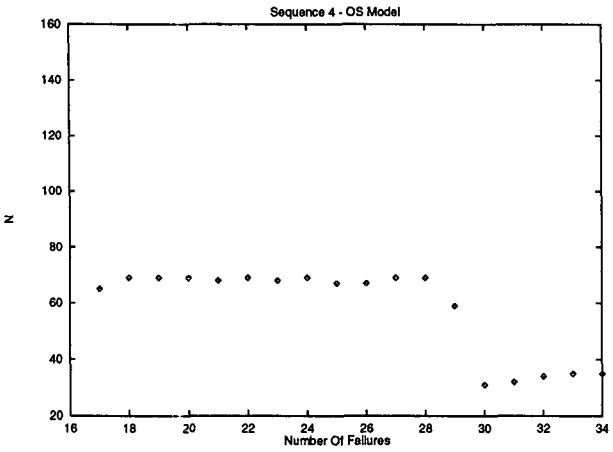

Figure 12: Order Statistic N Progression

The quality of predictions from the Order Statistics model under mixed testing and the Musa Log model under representative testing are very similar. Both of these models seem to provide better fits and predictive performance than the Jelinski-Moranda model.

An important difference between the Order Statistics model and the Musa Log model, however, is that the the Order Statistics Model required far fewer test cases to generate its failure set, because it used a mixed method approach to testing that utilized directed testing. Of course, since it is more expensive to generate a directed test case than a representative test case, this advantage may be somewhat offset or negated. Nonetheless, the use of multiple testing methods for reliability assessment should yield a more robust and comprehensive testing process than would be obtained if a single method were used.

References

[1] B. Beizer. *Software Testing Techniques*. Van Nostrand Reinhold, second edition, 1990.

[2] R. H. Cobb and H. D. Mills. Engineering software under statistical quality control. *IEEE Software*, 7(6):44–54, Nov. 1990.

[3] H. A. David. *Order Statistics*. John Wiley and Sons, New York, NY, second edition, 1981.

[4] J. R. Dunham. Experiments in software reliability: Life-critical applications. *IEEE Transactions on Software Engineering*, pages 110–123, Jan. 1986.

[5] W. Farr. Software reliability modeling survey. In M. R. Lyu, editor, *Handbook of Software Reliability Engineering*, pages 71–117. IEEE Computer Society Press, 1996.

[6] M. A. Hoppa and L. W. Wilson. Some effects of fault recovery order on software reliability models. In *Fifth International Symposium on Software Reliability Engineering (ISSRE 94)*, pages 338–342, Monterey, CA, Nov. 1994. IEEE Computer Society press

[7] R. C. Linger Cleanroom process model. *IEEE Software*, 11(2):50–58, Mar. 1994.

[8] B. Littlewood. Theories of software reliability: How good are they and how can they be improved? *IEEE Transactions on Software Engineering*, SE–6(5):489–500, Sept. 1980.

[9] B. Littlewood. Stochastic reliability-growth: A model for fault-removal in computer-programs and hardware-designs. *IEEE Transactions on Reliability*, R–30(4):313–320, Oct. 1981.

[10] M. R. Lyu, editor. *Handbook of Software Reliability Engineering*. IEEE Computer Society Press, 1996.

[11] B. Mitchell and S. J. Zeil. A reliability model combining representative and directed testing. In *Proceedings of the 18th International Conference on Software Engineering*, pages 506–514, Berlin, Mar. 1996. IEEE Computer Society Press.

[12] B. Mitchell and S. J. Zeil. An experiment in estimating reliability growth under both representative and directed testing. Technical Report TR–97–31, Old Dominion University, July 1997.

[13] B. Mitchell and S. J. Zeil. Modeling the reliability growth of non-representative testing. *Annals of Software Engineering*, 4:11–29, 1997.

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

[15] L. J. White. Software testing and verification. In M. Yovits, editor, *Advances In Computers*, volume 26, pages 335–391. Academic Press, Inc., London, UK, 1987.

[16] C. Wild, S. Zeil, J. Chen, and G. Feng. Employing accumulated knowledge to refine test cases. *Software Testing, Verification, and Reliability*, 2(2):53–68, July 1992.