A Conservative Theory for Long Term Reliability Growth Prediction

P.G. Bishop
Adelard
London E3 2DA
UK

R.E. Bloomfield
Adelard
London E3 2DA
UK

Abstract

This paper describes a different approach to reliability growth modelling which should enable conservative long term predictions to be made. Using relatively standard assumptions it is shown that the expected value of the failure rate after a usage time t *is bounded by:*

$$\bar{\lambda}_t \leq \frac{N}{et}$$

where N *is the initial number of faults and* e *is the exponential constant. This is conservative since it places a worst case bound on the reliability rather than making a best estimate. We also show that the predictions might be relatively insensitive to assumption violations over the longer term. The theory offers the potential for making long term software reliability growth predictions based solely on prior estimates of the number of residual faults. The predicted bound appears to agree with a wide range of industrial and experimental reliability data.*

It is shown that less pessimistic results can be obtained if additional assumptions are made about the failure rate distribution of faults.

1 Introduction

A large number of reliability growth models have been developed over the years (e.g. [1], [4], [6], [7], [9], [12]). These models employ a number of different strategies which attempt to extrapolate future reliability from the observed failures. The approaches tend to work over the short term but lack predictive power over the long term (i.e. for usage times which are orders of magnitude greater than the current usage time). This paper describes a new approach to reliability growth modelling which could enable conservative long term predictions to made with quite limited initial data. The following sections will describe the underlying concepts and assumptions, the basic theory, and some empirical evaluations of the theory applied to available reliability growth data.

2 Underlying concepts

The observed reliability of a system containing design faults is based on three main factors:

- the number of faults
- the size and location of faults
- the input distribution (operational profile)

This is illustrated in the following diagram.

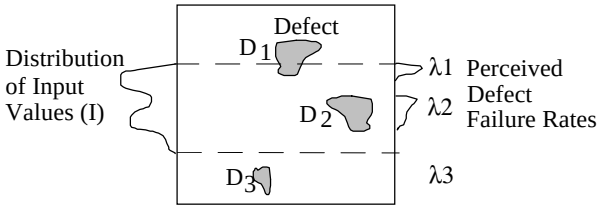

Figure 1 Illustration of the software failure process

A failure occurs when an input value is selected which activates a fault. The perceived defect failure rate λ_i should depend on the probability/unit time, *P(j)*, of activating points in input space D_i covered by defect *i*, i.e.:

$$\lambda_i = \sum_{j \in D_i} P(j)$$

Under a stable input distribution with no fault correction, λ_1 .. λ_N should be stable but the number of faults and their failure rates are unknown. While there are several methods for estimating the likely number of faults, at first sight there seems no way to establish the defect failure rates. But in fact we can use additional knowledge about the usage time of the software to place a bound on the failure rate contributions of the faults. This is the basis of the theory developed below.

2.1 Basis of the new model

The new model makes the relatively standard reliability modelling assumptions that:

1. removing a fault does not affect the failure rates of the remaining faults
2. the failure rates of the faults can be represented by $\lambda_1, \lambda_2 .. \lambda_N$, which do not change with time (i.e. the input distribution is stable)
3. any fault exhibiting a failure will be detected and corrected immediately

Note that assumption 2 implies that failures conform to the standard exponential arrival time model.

The basic idea behind the model is very simple; once the software has been operating for some time, faults with the highest failure rates will be removed, while faults with low failure rates only make a small contribution to the residual software failure rate. Thus for any time *t* there is an optimal defect failure rate which maximizes the residual software failure rate.

Put more formally, using the assumptions given above, a fault *i* with a perceived failure rate λ_i can survive a usage time *t* with a probability of:

$$e^{-\lambda_i t}$$

A defect can only contribute to the future unreliability of the program if it survives, so the average failure intensity of the program due to defect *i* after time *t* will be:

$$\overline{\lambda}_i | t = \lambda_i \, e^{-\lambda_i t}$$

Differentiating with respect to λ_i, the maximum value of $\overline{\lambda}_i | t$ occurs when $e^{-\lambda_i t} - \lambda_i t \, e^{-\lambda_i t} = 0$

i.e. when $\lambda_i = \frac{1}{t}$

Substituting back, it follows that the maximum failure rate contribution of any fault after the software has operated for a time *t* is:

$$\overline{\lambda}_i | t \leq \frac{e^{-1}}{t} \qquad (1)$$

This result is *independent of the actual failure rate of the fault.* The failure-rate independence of the bound is illustrated in the figure below.

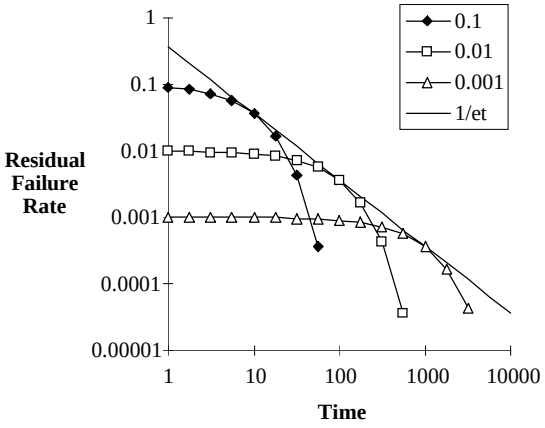

Figure 2 Failure rate independence of the bound

It is clear that, regardless of the defect failure rate, the residual contribution after time *t* is bounded by 1/*et*.

By summing the bounds for all N faults, we obtain a residual failure rate bound for the program of:

$$\overline{\lambda}_t \leq \frac{N}{et} \qquad (2)$$

where $\overline{\lambda}_t = \Sigma \, \overline{\lambda}_i | t$. This is a surprising result because it permits long term predictions to be made about a system *without extrapolating from the observed failures*. If the model assumptions are valid and we can estimate the number of faults at the time of release (e.g. using estimation methods such as [5] or [11]), the reliability growth can be bounded at *any time in the future*. Note that it does not tell us when (or even if) the faults will be found, but it does set a quantitative bound on the residual failure rate of the program and this bound always decreases with increasing test or operating time. The disadvantage of this simple bounding model is that it can be overly pessimistic. As the subsequent analyses will show, realistic failure rate distributions could exhibit significantly better growth than the worst-case bound.

3 Modelling defect failure rates

In a more general model, we need to consider the likely failure rates of the faults under a given input distribution *I*. We represent this by a fault density function $\sigma(I,\lambda)$ where $\sigma(I,\lambda)\delta\lambda$ gives the number of faults that exist with failure rates in the range $\lambda .. \lambda + \delta\lambda$. Using this density function, the expected number of faults surviving after time *t* will be:

$$\int_0^\infty \sigma(I,\lambda) e^{-\lambda t} d\lambda \qquad (3)$$

and the expected value of the failure rate of the overall program after usage time *t* is simply:

$$\overline{\lambda}_t = \int_0^\infty \sigma(I,\lambda) \lambda e^{-\lambda t} d\lambda \qquad (4)$$

The main advantage of this reformulation is that we can introduce additional constraints on the fault density function based on external knowledge. If these constraints are valid, worst case bounds can be set on the future reliability of the software.

As discussed earlier, one particular piece of additional knowledge we can incorporate is an independent estimate of the total number of faults within the software, $\hat{N}$. Knowing this number places a constraint on the initial fault density function, $\sigma(I,\lambda)$, i.e.

$$\hat{N} \approx \int_0^\infty \sigma(I,\lambda) d\lambda$$

While this fixes the integrated value $\sigma(I,\lambda)$, the actual shape is unknown, but this relatively weak constraint still allows us to establish worst case bounds on reliability growth. The following analyses examine both discrete

and continuous fault density functions. Both analyses confirm that equation (2) represents the worst case bound, but they also show that the actual bounds can be significantly better than this if we impose additional constraints on the overall shape of the fault density function.

4 Analysis for discrete failure rates

For any given program and input distribution, each fault will have a single specific failure rate. So the fault density mass is located at discrete points on the failure rate axis, as illustrated below.

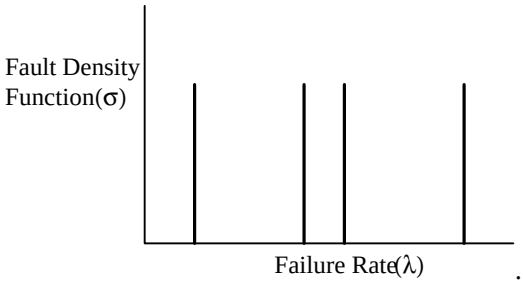

Figure 3 Density function for a discrete set of faults

To simplify the analysis we first define an "effort factor" M_t which normalises the achieved failure rate λ_t with respect to the operating time *t*, i.e.

$$M_t = \overline{\lambda}_t t \qquad (5)$$

or put another way, the residual failure rate is M_t times greater than 1/*t*. This dimensionless effort factor is almost identical to the "heroic debug" factor (t_i /$mttf_i$) discussed by Littlewood and Strigini [10], i.e. it is a multiplying factor which relates the testing time *t* to the achieved MTTF.

Since the failure rates of individual faults can be added to obtain the overall failure rate, it follows that the effort factors for individual faults can also be added, i.e.

$$M_t = \sum_{i=1,N} M_{ti} \qquad (6)$$

where M_{ti} is the effort factor for a single fault, i.e.

$$M_{t_i} = \overline{\lambda}_{i_t} t = t\lambda_i e^{-\lambda_i t}$$

Since the "effort factors" for the individual faults are additive, the following analysis only considers the variation in the effort factor for a single fault.

The time *t* for which the effort factor (M_{ti}) is a maximum occurs when $dM/dt = 0$, i.e. when:

$$\lambda_i e^{-\lambda_i t} - t\lambda_i^2 e^{-\lambda_i t} = 0$$

hence: $$t = \frac{1}{\lambda_i}$$

Substituting back into M_{ti} we obtain a maximum value for M_{ti} of:

$$\max M_{ti} = \frac{1}{e} \qquad (7)$$

This can be illustrated in the following figure which shows the variation of M_{ti} against *t* normalised so that time is expressed in units of $1/\lambda_i$.

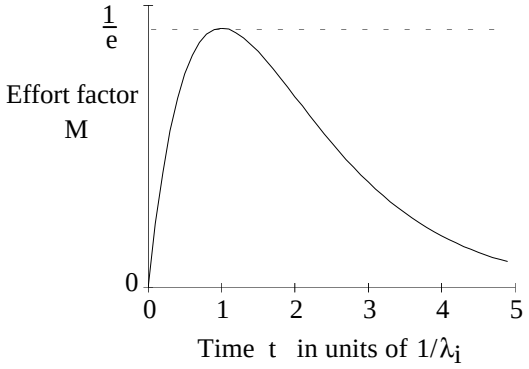

Figure 4 Variation in effort factor for a single fault

This shows that the relative cost of testing to achieve a given MTTF is greatest when $t=1/\lambda_i$. This is because, even prior to testing, the software is likely to have a finite failure rate. Small amounts of testing are unlikely to reveal the fault so there is little reduction in residual failure rate for the time spent. Since the residual failure rate, $\overline{\lambda}_{i_t}$, is virtually constant, the effort factor initially rises almost linearly with time. It is only when $t \geq 1/\lambda_i$ that the exponentially decreasing survival probability becomes dominant.

From equation (6) the effort factor for *N* faults is the sum of the individual effort factors. This is illustrated in the following diagram.

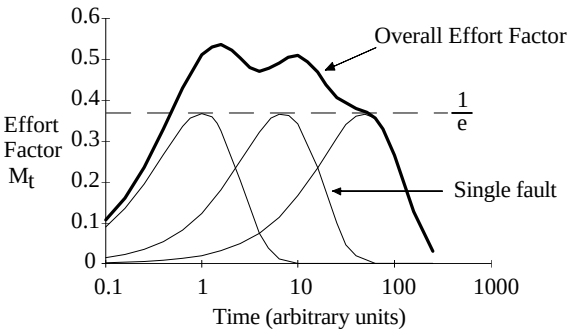

Figure 5 Variation in effort factor for multiple faults

We know from (7) that for any single fault, the effort factor is bounded by 1/*e*, so we know that for *N* faults the effort factor is bounded by:

$$M_t \leq \frac{N}{e} \qquad (8)$$

In fact this effort factor bound can only be reached if all N faults have an identical failure rate, i.e. the fault density function $\sigma(I,\lambda)$ is a single "spike" (so the individual effort factor terms peak at the same time t). Any other density function will have a maximum effort factor which is less than N/e.

Hence from the definition of M_t it follows that:

$$\bar{\lambda}_t \leq \frac{N}{e}.\frac{1}{t} \qquad (9)$$

This confirms the result in equation (2), but it can seen from Figure 4 that if the faults are widely spaced over the failure rate range, only one or two faults make a significant contribution to the effort factor at any given time, so the actual reliability growth could be much better than the worst case bound given by equation (2).

Conversely, if we take an extreme case where the software contains an infinite number of infinitely small faults, the failure rate remains unaltered by testing and correction, and consequently the effort factor would always increase with testing time. A continually rising effort factor has been observed empirically [10], but the above analysis shows that for any finite number of faults, the effort factor should be bounded by *N/e* and could even decrease in the long term.

The variation in growth rate for different fault density functions is considered in more detail in the following section where we examine a continuous fault density function based on the gamma distribution.

5 Worst case bounds for a continuous fault density function

As an alternative to a discrete density function, a continuous fault density function can be used to model the expected number of faults existing over the failure rate spectrum. One equation that can be used to represent the spread of defect failure rates is the gamma distribution. The gamma distribution is defined as:

$$\frac{1}{\Gamma(\alpha)\beta^{\alpha}}.\lambda^{\alpha-1}.e^{-\lambda/\beta} \qquad (10)$$

where $\Gamma(\alpha)$ is the gamma function which has the property that $\Gamma(\alpha+1)=\alpha\Gamma(\alpha)$.

The gamma distribution was chosen because it is quite flexible; the same family can approximate to a number of distributions, e.g.:

$\alpha \to 0$, $gamma(\alpha,\beta,\lambda) \propto \lambda^{-1}$ (reciprocal)

$\alpha = 1$, $gamma(\alpha,\beta,\lambda) \propto e^{-\lambda/\beta}$ (exponential)

$\alpha \to \infty$, $gamma(\alpha,\beta,\lambda) \propto gaussian$ (spike)

This is illustrated in Figure 6.

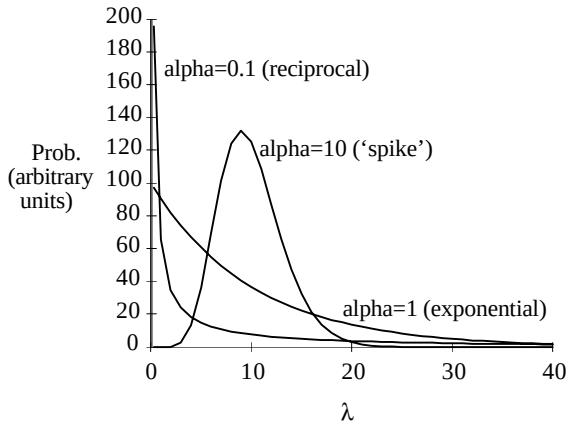

Figure 6 Examples of gamma distributed failure rates

The fault density function has to take into account the overall number of faults, *N*, hence:

$$\sigma(I,\lambda) = N\frac{1}{\Gamma(\alpha)\beta^{\alpha}}.\lambda^{\alpha-1}.e^{-\lambda/\beta} \qquad (11)$$

Substituting this fault density function into equation (3), it can be shown that:

$$\bar{\lambda}_t = \frac{\alpha\,\beta\,N}{(1+\beta\,t)^{\alpha+1}} \qquad (12)$$

We can relate the reliability improvement to the usage time using the "effort factor" M_t defined in equation (5) where:

$$M_t = N\frac{\alpha\beta t}{(1+\beta t)^{\alpha+1}} \qquad (13)$$

We can now examine the effects of some of the extreme cases of the fault density function σ on the effort factor.

When $\alpha \to 0$ the fault density function approximates to the reciprocal function ($\sigma \propto \lambda^{-1}$). For any finite failure rate it follows that $\beta \to \infty$, so equation (13) approximates to:

$$M_t \cong \alpha N$$

This implies there will be a fixed ratio between the test time and the MTTF (regardless of the test time). This corresponds to the case discussed in the previous section where the effort factor peaks are evenly spaced on a logarithmic time axis (Figure 4) so that only a few faults contribute to the effort factor at any given time.

An exponential fault density function (α=1) gives:

$$M_t = \frac{\beta\,N.t}{(1+\beta\,t)^2}$$

Since the reliability does not change very much initially, the effort factor increases linearly for small *t*. This is consistent with observations of "heroic debug". However when $t>1/\beta$ the effort factor is proportional to $1/t$ and so drops towards zero. This effect is even more

pronounced for larger values of α where the density function is more "spiky". These variations in reliability growth "effort" are illustrated in the figure below.

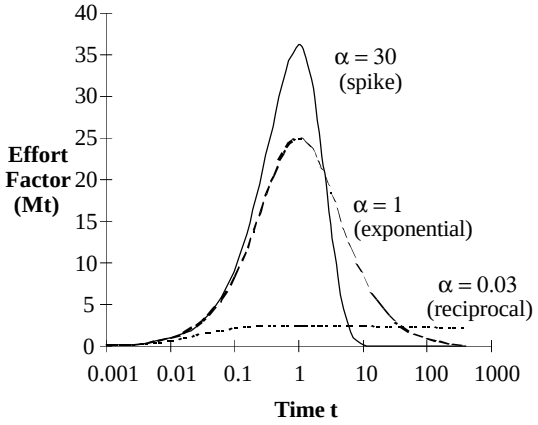

Figure 7 Examples of variation in effort factor (100 faults)

We can derive a worst case bound on the effort factor for a gamma density function. The maximum value of M_t occurs when $\frac{dM_t}{dt} = 0$.

So by differentiating equation (13), it can be shown that the maximum value of M_t occurs when:

$$t = \frac{1}{\alpha\beta} \tag{14}$$

Substituting in equation (13) we can show that:

$$\max M_t = N\left(\frac{\alpha}{\alpha+1}\right)^{\alpha+1} \tag{15}$$

There is a standard result that

$$\left(1+\frac{x}{n}\right)^n \to e^x \text{ as } n \to \infty.$$

Using this result it can be readily shown that

$$\left(\frac{\alpha}{\alpha+1}\right)^{\alpha+1} \to \frac{1}{e} \text{ as } \alpha \to \infty,$$

Hence $\max M_t \to \frac{N}{e} \quad (\alpha \to \infty)$

Using equation (15), max M_t for some key fault density shapes are summarised in the table below.

Shape Parameter	max M_t
$\alpha \to 0$ (reciprocal)	$N\alpha$
$\alpha = 1$ (exponential)	$N/4$
$\alpha \to \infty$ (spike)	N/e

Table 1 Maximum effort factor vs. fault density shape

The variation of max M_t over the whole range of α (for N=1) is shown in the graph below.

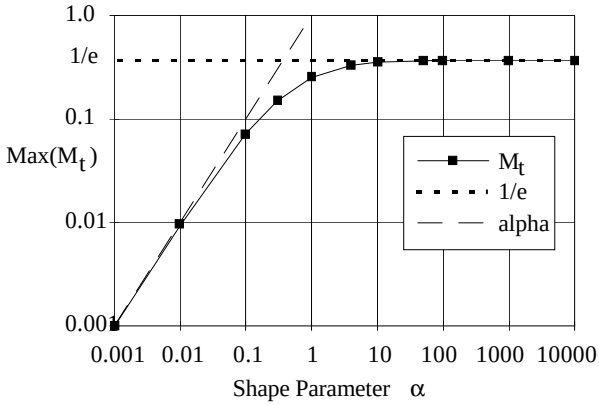

Figure 8 Variation of maximum effort factor with shape parameter (N=1)

So we can assert that the effort factor M_t is bounded by:

$$M_t \le \frac{N}{e} \tag{16}$$

As M_t is equivalent to $\bar{\lambda}_t t$, it follows that:

$$\bar{\lambda}_t \le \frac{N}{e} \cdot \frac{1}{t} \tag{17}$$

This upper bound on the failure rate would apply regardless of the actual parameters of the gamma distribution. The bound is identical to the general result for N faults with discrete failure rates derived earlier. This is not surprising as the limiting case is the same—a "spike".

The best case reliability growth occurs for the reciprocal density function when the failure rate tends to αN/t for all *t*. This expression is hard to interpret directly but if we consider an explicit reciprocal density function, the meaning becomes more obvious. If the density function is:

$$\sigma(I,\lambda) = \frac{k}{\lambda}$$

it can be shown that the number of faults in the interval λ_0 to $e\lambda_0$ is *k* and the "effort factor" remains at a constant value of *k* for all *t*. It can also be shown that there will be 2.3*k* faults ($\log_e 10\ k$) for a tenfold change in failure rate.

From the gamma distribution analysis, the equivalent effort factor is αN. So the interpretation is that the gamma model approximates to a truncated reciprocal function where the N faults are spread over 1/2.3α decades of failure rate.

It is this spread of failure rates that determines the level of pessimism in the worst case bound. If the faults are reciprocally spread across *R* orders of magnitude in failure rate, the "best case" effort factor can be shown to be *N*/2.3*R* (assuming the truncation effects are negligible). By contrast the "spike" assumption yields an effort factor of N/e. So the ratio of the worst case bound

and the "best case" reliability is at most 2.3R/e. For example, if the faults spanned 5 orders of magnitude in failure rate, the worst case bound prediction and the best case reliability would differ by a factor of 4.2. So for most realistic software systems the worst-case bound would be pessimistic by less than an order of magnitude.

6 Sensitivity to the model assumptions

The assumptions inherent in this analysis are most likely to apply to simple, mass-produced software items where diagnosis and correction are easy, and the input distribution has been "averaged out" over many different applications. However, over the long term, the model still appears to be applicable even when assumptions are violated. We will examine three cases where the assumptions could be violated: non-stationary input distributions, faulty corrections and imperfect diagnosis.

6.1 Non-stationary input distributions

In a stationary input distribution, there is a fixed, time-independent probability for each possible input value. In practice however, the software may have different modes of use at different times. To illustrate this, let us assume there are P disjoint partitions of the input space, which represent P different modes of use. Let us further assume that *N/P* faults can be activated in each partition, and that there is continuous execution in each mode for a time t/P. This is a "pseudo-stationary" model where each space is effectively an independent program. For this model we would predict a "saw-tooth" bound for the MTTF since the usage time is effectively zero when a new partition is entered. After time *t*, when all partitions are covered, the MTTF bound in each partition will be:

$$MTTF_{t,p} \geq e.\frac{P}{N}.\frac{t}{P} = \frac{e}{N}t$$

This shows that, once the entire input space is covered, the MTTF bound is identical to that achievable in time t with only a single partition (P=1). Of course it is unlikely that such a simplistic input distribution would apply in practice, but for any periodic input distribution we would expect short-term "peaks" in the failure rate (when new areas of the input space are being explored). In the long term however, the *average* probability of activating an input value tends to a fixed value, so the predicted bound should apply once the usage time significantly exceeds the periodic interval.

6.2 Unreliable fault correction

A faulty correction may replace one fault with another which could be of arbitrary size, and potentially be located anywhere in the input space. Nevertheless, as time progresses, the potential failure rate contribution of the new correction-induced fault, $\bar{\lambda}_c|t$ will be bounded by:

$$\bar{\lambda}_c|t \leq \frac{1}{e}.(t - t_c)^{-1}$$

where t_c is the time of the correction. Once $t >> t_c$, the failure rate bound for the new fault will be very similar to the one it replaced, so in the long term the prediction based on the original number of faults will still apply.

The same argument can also be applied to the additional faults that are introduced when there is a functional upgrade to the software.

6.3 Imperfect diagnosis

In some cases (e.g. real-time systems) it is difficult to identify the faults from the symptoms so multiple failures will occur before the problem is identified and corrected. If we take a simple model where *d* failures have to occur for each fault before it is corrected, the effective failure rate of a fault after time *t* will be:

$$\bar{\lambda}_{i_t} = \lambda_i e^{-\frac{\lambda_i t}{d}}$$

If this model is used, it can be shown that, for *N* faults, the expected program failure rate after time *t* is:

$$\bar{\lambda}_t \leq \frac{Nd}{e}.t^{-1}$$

So poor diagnosis has the effect of "scaling up" the failure rate contribution of each fault. Any system where there was low probability of correction would have to include this factor when making a reliability growth prediction.

7 Empirical support for the theory

The theory suggests that, providing we can make an independent estimate for the number of faults in the software, $\hat{N}$, we can make a worst case bound prediction for the expected future MTTF at any time *t*, namely:

$$MTTF_t \geq \frac{e}{\hat{N}}t$$

Even in cases where the operational profiles are non-stationary, we might expect the bound to apply once sufficient time has elapsed to average out the variations. In addition, it seems that a perturbation introduced by a faulty correction only has a marginal impact on the long term prediction of the failure rate bound. Poor diagnosis can have a major effect on reliability growth, but we do have some grounds for believing that the model could be used to predict the worst case bounds for reliability for quite a broad range of applications, and the bound should typically be within one order of magnitude of the achieved reliability level.

7.1 Analysis of the PODS experimental data

The PODS project [2] was concerned with the evaluation of dependency in diverse programs, but the programs, faults and test data were retained for a follow-up study [3]. The main merit of using this particular data source is that the test input distributions were known and had a stable input distribution profile, which is one of the key assumptions in this model. The testing on the PODS programs was performed back-to-back; when discrepancies were observed, the fault was corrected and testing continued. The following analyses treat the program triple as a single program. The triple contained 31 distinct faults (some of which existed in more than one program version). We have used this number as the fault estimate, although in practice this figure would have to be derived by independent means (e.g. from the lines of code).

There is a problem of defining "usage time" for this test data. After a fault correction, the tests were repeated from the beginning. On a model of perfect fault removal, the tests prior to the last failure should always succeed and so contribute nothing to fault detection. An alternative measure of usage time would exclude the earlier test data, except when correction-induced faults are found. In the following analyses the second definition of usage time is used. Figure 9 shows the growth in time-to-failure (TTF|t) using random input distribution test data.

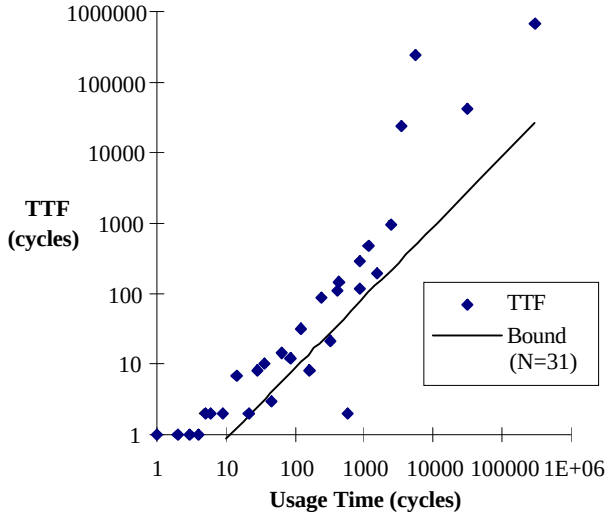

Figure 9 Growth of time to failure: PODS uniform random test data (*note axes are logarithmic*)

The predicted lower bound is also plotted on the figure, assuming N=31. It can be seen that most TTFs lie above the bound. The bound actually relates to the *average* TTF, so statistically some TTFs could fall outside the limits. The one point that falls a long way below the line is known to be a correction-induced fault, but this has little impact on subsequent reliability growth.

A similar analysis was performed using test data based on simulated plant operation, and a very similar pattern was observed.

7.2 Analysis of Musa SYS1 and SS3 data

The Musa data sets [12] do not give much background information about the operating context, so we do not know whether the operational profiles were static or varied with time. This may account for the extreme short term variations in TTF observed in the data.

As there are many observed failures and relatively low growth, it seems reasonable to take the average of 10 TTFs to form an estimate for the MTTF. The results are shown in Figures 10 and 11. Error bars on each point represent one standard deviation. The actual number of faults is unknown, but it will be at least equal to those already observed. In the examples, we have assumed that the actual number of faults is around 1.5 times the number of known faults. The worst case bound prediction was applied to the data series for SYS1 and SS3.

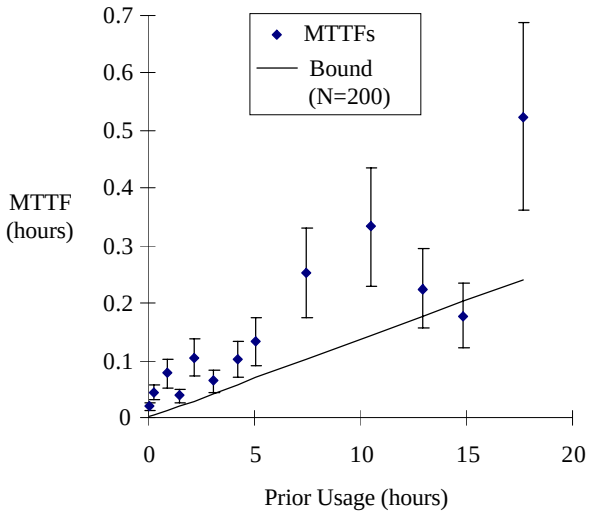

Figure 10 Reliability growth (Musa SYS 1)

The wide swings in reliability (even when using averaged data) suggest that the input distributions are far from stable. The decreases in reliability could relate to new modes of operation or new types of testing, so it is not clear that the model assumptions have been satisfied. The observed MTTF is sometimes lower than the predicted bound, but this would be expected with a varying operational profile which could well result in a "saw-tooth" pattern of failure rates. In the longer term the growth does seem to be in reasonable accord with the model. This may indicate that all the major modes of operation have been explored, so that the averaged input distribution is beginning to stabilise.

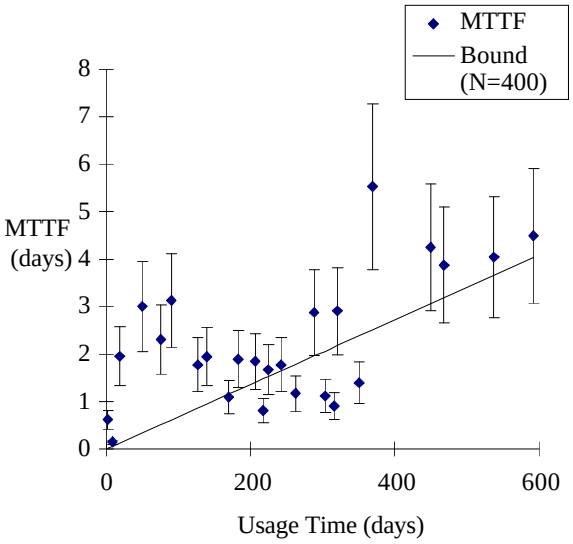

Figure 11 Reliability growth (Musa SS3)

7.4 Tropico-R teleswitch data

In [8] the reliability growth of three generations of teleswitch equipment have been analysed. Much of the detailed data are confidential, but information is available about: the number of known faults; the software size; and the failure rate over time. Most of the reliability growth data are based on operation in the field. One complicating factor is that new systems were being progressively installed on different sites, each with a different operational profile and possibly different software options, so that new parts of the input space could be covered for each new installation. The results for one generation of teleswitch (Tropico-R PRB) are shown below. We have again used a fault estimate which is 50% greater than the known faults.

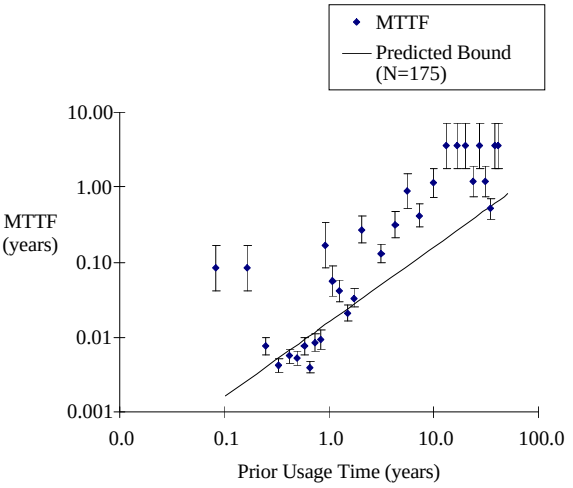

Figure 12 Tropico R PRB reliability growth
(note axes are logarithmic)

As in the previous example the error bars shown represent one standard deviation. It can be seen that the MTTF sometimes exceeds the lower bound by more than an order of magnitude. The high MTTFs at the initial stages are probably associated with validation testing (which may not match operational use). The low values observed after extensive usage appear to be related to the installation of new switches, where the system configuration or usage pattern may activate new faults. The same pattern can be observed in the development of the other generations of system (PRA and PRC).

8 Reducing prediction pessimism

For cases where the fault density function is not a spike, the bound is unduly pessimistic. Additional information from the observed failures could be used to reduce this pessimism while remaining conservative, e.g. by:

1. updating the residual fault estimate when faults are detected
2. estimating the shape of the fault density function based on failure data.

In the first approach we still assume the density function for the residual faults is a spike, but of course N is smaller, and the revised value can be used to compute a new bound. The main problem with this approach is in maintaining conservatism about the remaining number of faults; a simplistic updating procedure could result in a negative number of faults. To be effective more sophisticated residual fault estimation techniques are needed (e.g. [13]) which retain some conservatism.

In the second approach, we could assume that the fault density function follows a gamma distribution and estimate the α and β parameters from the observed failures, (e.g. from the variation of the effort factor (M_t) with usage time). Such approaches have been used in many reliability models [4], but the estimates are usually unstable due to short-term variations in the time-to-failure.

A less ambitious approach is to introduce a constraint on the σ function to fix the total failure rate for the N defects and then calculate the worst case density function at any given time. It can be shown from equation 12, that the initial failure rate is:

$$\lambda_o = N\alpha\beta$$

If we use a measurement of the initial failure rate, we can infer that the average failure rate for a fault is λ_o/N and this fixes the location of the "spike" in the failure rate spectrum. With the gamma model, the maximum fault density always occurs at $\lambda = \lambda_o/N$, but for usage times greater than N/λ_o the density function that gives the worst case growth is no longer a spike. The optimum shape for worst case growth gradually changes as time increases (the spike is gradually "squashed" to cover higher and lower failure rates around the peak). In the

limit, the distribution tends towards that for the reciprocal model.

This behaviour can be best visualised by considering the "effort factor" for reliability growth. The larger this number is the harder it is to achieve reliability growth. To achieve the worst case growth estimate, we would therefore seek to maximise M_t for all values of t. The effort factors for different distribution shapes (different α values) are shown in Figure 13 below. The worst case effort factor is the envelope of maximum values traced out as we vary α. On the left-hand side of the peak, the effort factor is a maximum when $\alpha = \infty$ (the "spike" density function). On the right hand side the maximum effort factor occurs with a greater spread of failure rates. For large values of t, the worst case growth occurs when $\alpha \to 0$ (the reciprocal distribution discussed earlier).

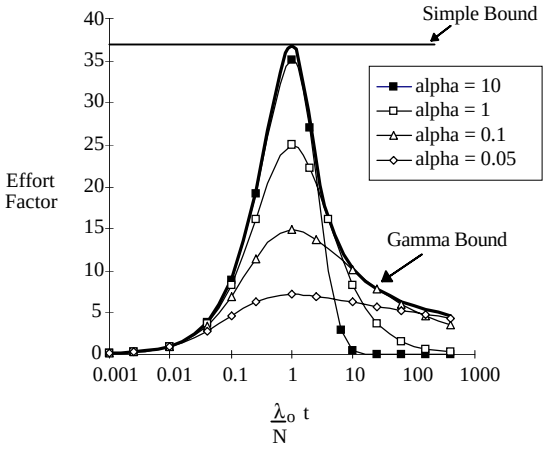

Figure 13 Worst case effort factor over time (N=100)

The figure also shows the effort factor for the simple theory. In the simple theory, we have no data on the position of the fault density peak so we have to assume the worst case value for all usage times. It can be seen that less pessimistic growth rates are obtained for the gamma model, and the simple model represents a bound for the gamma model.

This gamma model still operates on quite limited data, namely an estimate of the number of residual faults combined with a measurement of the initial failure rate. Since the model assumes the worst possible fault density function for all usage times, the model should still be conservative but less pessimistic than the simple theory. It should be noted that, since it computes a worst-case bound, the gamma estimate has the same basic shape for any reliability prediction; the same basic curve is simply rescaled based on the number of residual faults and the measurement of the initial failure rate.

This revised model has been assessed against empirical reliability growth data and does seem to be less pessimistic. The bound for the Tropico PRB teleswitch system is shown below.

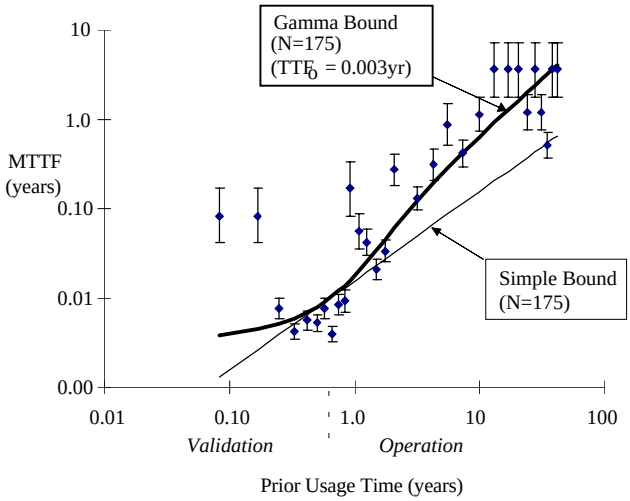

Figure 14 Tropico PRB reliability growth – simple vs. gamma worst case bounds

For the Tropico PRB development there was a nine month period of validation testing prior to operation. It can be seen that some unrealistic failure rate values were obtained during the early period of validation testing. While the validation time has been included in the figure above, the initial failure rate is estimated from the first period of actual operation. An error in the estimate of the initial failure rate affects the point where the gamma bound touches the simple bound, but it does not have a major impact on the long term trend because the ratio of gamma bound to the simple bound tends toward a relatively stable value. For long-term growth, i.e. one or two orders of magnitude beyond the time where the two bounds touch, the gamma bound is typically 3 to 5 times less pessimistic than the simple bound.

The results illustrate a difficulty of using the gamma model. It presupposes that the initial failure rate is "typical" and hence that the input distribution is relatively unchanging over time. When the input distribution does change, the bound can be too optimistic. This also applies to the simple theory, but this is inherently more conservative. so violations of the bound are less frequent.

Nevertheless, the general fit of the worst case gamma curve is quite good. Starting with an initial failure rate of around one per day, the model makes forward predictions to failure rates of around one per year and these predictions are accurate within a factor of three. This is an improvement on the simple model, which can underestimate the achieved reliability by up to one order of magnitude.

9 Concluding remarks

Based on some relatively standard modelling assumptions, we have derived a worst case bound for software reliability growth, where:

$$\bar{\lambda}_t \leq \frac{N}{et}$$

This prediction depends on a number of assumptions, namely: a stable input distribution, perfect diagnosis and perfect correction; but we have also shown that the predictions might be relatively insensitive to assumption violations over the longer term. The empirical data drawn from field experience seems to support the general results of the theory, but much of the data is taken from high-volume industrial systems where the assumptions are most likely to apply. Further work is desirable to check the applicability of the model assumptions and the theory.

It has also been shown that the worst case bound prediction can be unduly pessimistic, possibly by as much as one order of magnitude. It is possible to make less pessimistic, long-term bound predictions by assuming gamma-distributed failure rates and including a measurement of the initial failure rate. This extension to the simple bound theory also seems to be consistent with the empirical field data and estimates bounds within a factor of three.

By its very nature, a theory that makes a bounding estimate will always be less accurate than a conventional reliability growth model over the short term. However the general approach of incorporating prior knowledge about the software does seem to be a powerful one and, unlike conventional reliability growth theories, it can make conservative predictions of reliability growth over the long term (e.g. many hundreds of years of usage time).

Another attractive feature of the theory is that it makes a quantitative link between the number of faults and reliability. This provides a numerical justification for the conventional wisdom encapsulated in existing standards for high integrity software, which seek to minimise faults by implementing high quality production methods and by placing restrictions on software complexity.

Acknowledgements

We wish to thank Professor B. Littlewood for his constructive comments and criticism.

This work was funded by the UK (Nuclear) Industrial Management Committee (IMC) Nuclear Safety Research Programme under Scottish Nuclear contracts 70B/-0000/006384 and PP/74851/HN/MB with contributions from British Nuclear Fuels plc, Nuclear Electric Ltd and Scottish Nuclear Ltd.

References

[1] A.A. Abdel-Ghaly, P.Y. Chan and B. Littlewood, "Evaluation of Competing Software Reliability Predictions," *IEEE Trans. on Software Engineering*, vol. SE-12, no. 9, pp. 950–967, 1986.

[2] P.G. Bishop et al, "PODS a Project on Diverse Software", *IEEE Trans. Software Engineering*, Vol. SE-12, No. 9, 929-940, pp. 929–940, 1986.

[3] P.G. Bishop et al, "STEM: a Project on Software Test and Evaluation Methods", *Safety and Reliability Society Symposium 1987 (SARSS 87)*, Manchester, Elsevier Applied Science, ISBN 1-85166-167-0.

[4] S. Brocklehurst, P.Y. Chan, B. Littlewood and J. Snell, "Recalibrating software reliability models," *IEEE Trans Software Engineering*, vol. SE-16, no. 4, pp. 458–470, 1990.

[5] J.R. Gaffney, "Estimating the Number of Faults in Code", *IEEE Trans. Software Engineering*, vol. SE-10, no. 4, 1984.

[6] A.L. Goel and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software and Other Performance Measures," *IEEE Trans. on Reliability*, vol. R-28, no. 3, pp. 206–211, 1979.

[7] Z. Jelinski and P.B. Moranda, "Software Reliability Research," in *Statistical Computer Performance Evaluation*, pp. 465–484, New York, Academic Press, 1972.

[8] K. Kaaniche, K. Kanoun, M. Cukier and M. Bastos Martini, "Software Reliability Analysis of Three Successive Generations of a Switching System", in *First European Conference on Dependable Computing* (*EDCC-1*), (Berlin, Germany), pp. 473–490, Oct. 1994.

[9] B. Littlewood, "Forecasting software reliability," in *Software Reliability Modelling and Identification*, pp. 141–209, Heidelberg, Springer, 1988.

[10] B. Littlewood and L. Strigini, "Validation of ultra-high dependability for software-based systems," *CACM*, vol. 36, no. 11, 1993.

[11] M.Lipow, "Number of Faults per Line of Code", *IEEE Trans. Software Engineering*, vol. SE-8, no. 4, 1982, pp. 437–439, 1982.

[12] J.D. Musa, "A Theory of Software Reliability and its Application," *IEEE Trans. on Software Engineering*, vol. SE-1, pp. 312–327, 1975.

[13] N.F. Schneidewind, "Software Reliability Model with Optimal Selection of Failure Data", *IEEE Trans. on Software Engineering*, vol. 19, no. 4, pp. 1095–1104, 1993