IMPROVING SOFTWARE RELIABILITY FORECASTING

BERNARD BURTSCHY
Telecom Paris, ENST, Dep. INF. 46 rue Barrault, 75634, Paris Cedex 13, France

GRIGORE ALBEANU
Faculty of Mathematics, Bucharest University, Academiei, 14, RO-70109, Romania

DRAGOS N. BOROS
Coopers & Lybrand MCS, 6 Splaiul Unirii St. B1.3, Bucharest, RO-75101, Romania

FLORIN POPENTIU
University "Politehnica" of Bucharest, UNESCO chair, Splaiul Independentei 313, 77206,
Bucharest, Romania

and

VICTOR NICOLA
University of Twente, TIOS, PO Box 217, 7500 AE Enschede, Holland

(Received for publication 23 April 1996)

Abstract—This work investigates some methods for software reliability forecasting. A supermodel is presented as a suited tool for prediction of reliability in software project development. Also, times series forecasting for cumulative interfailure time is proposed and illustrated. © 1997 Elsevier Science Ltd. All rights reserved.

1. INTRODUCTION

Software reliability deals with behaviour of a software system in operation and it is defined as the probability of working without failure for a specified interval of time.

In a number of recent disasters the computer has been wholly or partially to blame. In Ref. [1] an alarm signal is given by the following message: "Sometime soon, software reliability is going to become a highly visible and important field. Unfortunately, given human nature, its thrust into prominence will only happen once we experience the software equivalent of the Chernobyl, Bhopal, or space shuttle Challenger disasters. Such a disaster is likely to happen in the next few years."

Therefore, software reliability forecasting is a problem of increasing importance for many critical applications. Selection of a particular model is very important in software reliability predictions.

In spite of much research effort [2], there is no universally applicable software reliability growth model (SRGM) which can be trusted to give accurate predictions of reliability in all circumstances.

This paper suggests some "ways" of obtaining more accurate predictions by building a supermodel. A comparison is made between this approach and the general method for predicting the medians using actual data sets from different software projects. We are also investigating the application of autoregressive moving average processes (ARMA) to software reliability forecasting. Our analysis shows that accuracy of the predictions in different applications can be judged using the proposed methods.

2. SUPERMODEL FOR IMPROVING SOFTWARE RELIABILITY FORECASTING

In this part we consider a supermodel building as a weighted sum of several Software Reliability Growth Models (SRGMs). The weight factors will depend on the values of the prequential likelihood functions as calculated for each model, the values varying each time a new error is observed.

There are no restrictions regarding the models which form the supermodel. The only constraint is that of having the possibility of calculating the estimated probability density function (or cumulative distribution function) for each of them.

The supermodel predictors are based on a linear combination between the initial predictors (cdf) which correspond to each basic model:

$$F_i(t) = \sum_{k=1}^{m} w_k F_i(t), \quad k = 1, 2, \ldots$$

(1)
or, respectively, to the pdf:

$$f_k(t) = \sum_{i=1}^{m} w_i f(t), \quad k = 1, 2, \ldots, n,$$

(2)

where $$w_i$$ are the weights factors

$$\sum_{i=1}^{m} w_i = 1, \quad k = 1, 2, \ldots, n$$

(3)

and $$m$$ is the number of models which form the supermodel.

The difficulty in using the supermodels is caused by the way chosen to obtain the weight factors. There are several ways to achieve this goal. One of them is to use the maximum likelihood method using the expression of pdf of the supermodel, which due to the great number of parameters (usually $$m$$ is greater than three) is difficult to implement and is time consuming. Another approach is computationally more simple and is based on a Bayesian inference weighted decision which, in contrast, is more difficult to understand.

In this approach, the weighted factor for prediction system $$r$$ at a certain stage $$n$$ is:

$$w_r = \frac{PL_{L_{i=1}^{n}}} {\sum_{r=1}^{m} PL_{L_{i=1}^{n}}},$$

where $$PL$$ is the notation of the prequential likelihood function. It is obvious that the weight factor $$w_r$$ satisfies relation (4) but the reasons behind its formula are more subtle.

3. SOFTWARE BEHAVIOUR PREDICTION USING TIME SERIES TECHNIQUES

3.1. About the ARMA model

In this section, we present the application of autoregressive moving average processes (ARMA) to software quality modelling. Several recent papers concerning this subject are available [3, 4].

It is clear that future predictions must use the history of past observations. Another remark is that the faults found early in the software project development could be rejected and the prediction process will continue with the rest of them.

Two classes of reliability model arise: time-dependent models and time-independent models. Here only time-dependent models will be considered. In this context, the cumulative interfailure time sequence is considered and used for investigations. Time independent models will be considered elsewhere.

The basic idea is the use of the Box–Jenkins methodology for the design and prediction time series

[5]. This methodology, based on the Wold’s theorem [6] decomposes the time series $$Z(t)$$ into past values $$z(t-i)$$ and past impacts $$w_{j-k}$$ sustained by the series according to the ARMA (Autoregressive Moving Average) model

$$\text{ARMA}(p, q):$$

$$Z(t) = \sum_{i=1}^{p} \phi_i Z(t-i) + \sum_{j=1}^{q} \theta_j w(t-j) + w(t),$$

(5)

where

- $$\{w(t)\}$$ is an uncorrelated process with zero mean and finite variance.
- $$\{\phi_i, 0 \leq i \leq p\}$$ is a finite sequence with $$\phi_0 = 1$$, $$\phi_0 \neq 0$$.
- $$\{\theta_i, 0 \leq i \leq q\}$$ is a finite sequence with $$\theta_0 = 1$$, $$\theta_0 \neq 0$$.

The integers $$p$$ and $$q$$ are, respectively, the AR order and the MA order. The sequence $$\{\phi_i, 0 \leq i \leq p\}$$ and $$\{\theta_i, 0 \leq i \leq q\}$$ are, respectively, the AR parameters and the MA parameters.

If $$q = 0$$ then we obtain the finite autoregressive (AR) processes defined by

$$Z(t) = \sum_{i=1}^{p} \phi_i Z(t-i) + w(t).$$

3.2. ARMA parameter estimation

Let us rewrite eqn (5):

$$w(t) = \sum_{i=1}^{q} \phi_i z(t-i) - \sum_{j=1}^{p} \theta_j w(t-j).$$

(6)

When $$p$$ and $$q$$ are known, the parameter vector is

$$\mathbf{x} = (-\phi_1, \ldots, -\phi_p, \theta_1, \ldots, \theta_q)^T.$$

The parameter vector can be estimated by minimization of the weighted sum of squares of the impacts [7]. The impact $$w(t)$$ is related to $$Z(t)$$ through eqn (6). To estimate the parameter vector $$\mathbf{x}$$, we use the recursive maximum likelihood algorithm (ARMA/RLM) also called the recursive nonlinear least-squares algorithm.

Differentiation with respect to $$\phi_i$$ and $$\theta_i$$ yields

$$\frac{\partial w(t)}{\partial \phi_i} = Z(t-k) - \sum_{i=1}^{q} \theta_i \frac{\partial Z(t-i)}{\partial \phi_i},$$

(7)

and

$$\frac{\partial w(t)}{\partial \theta_i} = -w(t-k) - \sum_{i=1}^{p} \phi_i \frac{\partial Z(t-i)}{\partial \theta_i}.$$

(8)

Let $$\{u(t)\}$$ and $$\{v(t)\}$$ denote, respectively, the responses of $$\{Z(t)\}$$ and $$\{w(t)\}$$ to the linear time-invariant filter $$1/\theta(e^s)$$, where $$j = -1$$ and

$$\theta(e^s) = 1 + \sum_{k=1}^{q} \theta_k e^{-js}.$$
Improving software reliability forecasting

Then
\[
\frac{\partial w(t)}{\partial \phi_k} = u(t - k), \quad \frac{\partial w(t)}{\partial \theta_k} = -v(t - k).
\] (9)

The ARMA/RLM algorithm estimates \( w(t), u(t), \) \( v(t) \) at each new time point using the recursions
\[
\hat{w}(t + 1) = Z(t + 1) + \sum_{i=1}^{p} \hat{\phi}_i Z(t + 1 - k),
\]
(10)
(11)
(12)

We obtain the following ARMA/RML algorithm.
(1) Set \( P_{-1} = \epsilon I \), where \( I \) is the unit matrix and \( \epsilon \) is a large number. Set also \( \hat{\theta}_0 = 0 \) (approximate initialization).
(2) \( \hat{w}(t) = \hat{u}(t) = \hat{v}(t) = 0, -q \leq t \leq -1 \).
(3) For \( t = -1, 0, \ldots, d_0 \)
   (a) Apply relations (10), (11) and (12) in this order.
   (b) Let
   \[
   \eta(t) = [\hat{u}(t), \ldots, \hat{u}(t - p + 1), \hat{v}(t), \ldots, \hat{v}(t - q + 1)]^T.
   \]
   (c) Let \( K(t) = P(t) \eta(t)(\hat{\lambda} + \eta(t)^TP(t)\eta(t))^{-1} \). Here, \( \hat{\lambda} \) is used to control the algorithm fluctuations and tracking ability. Practical values of \( \hat{\alpha} \) are 0.9 and above.
   (d) Let \( P_{t+1} = \hat{\lambda}^{-1}[I - K(t)\eta(t)]P_t \).
   (e) Update the parameter vector:
   \[
   \hat{x}_{t+1} = \hat{x}_t + K(t)\hat{w}(t + 1).
   \]

3.3. Numerical example

In order to present the supermodels and their usage, we have chosen a data set of 832 interfailure times [8]. The raw SRGMs used were: JM (Jelinski–Moranda) [2, 9], GO (Goel Okumoto) [2, 10], DU (Duane) [11], LV (Littlewood–Verall) [2, 12] and KL (Keiller–Littlewood) [13].

In Fig. 1 we present the evolution of the median for these models and for the supermodel (SUP) built from all of them. In Fig. 2 we are presenting the median of three supermodels: SUPT (based on JM, GO, DU, LV, KL), SUPGODU (based on GO, DU) and SUPDUKL (based on DU, KL). As you can see there are striking differences between these plots. In order to have a more clear picture of the influence each particular SRGM has upon the supermodel estimates we present in Fig. 3 the evolution of weight factors for each corresponding to the SUPT supermodel while in Fig. 4 we present the weight factor of KL SRGM in the supermodels SUPT and SUPDUKL. Practically we observe that there is a tendency for the supermodel to eventually follow the best SRGM (KL in this case) while neglecting the others (\( w_{KL} \) is starting with the 170th or 320th failure).

The fact that KL is the best model for describing the failure phenomenon for this data set can be clearly seen in Fig. 5 where we plotted PLR for each

![Fig. 1. Median evolution for five SRGMs and their corresponding supermodel SUPT.](image-url)
SRGM against the SUPT supermodel and where KL is slightly under the axis. In Figs 6 and 7, we can observe the PLR for the other two supermodels and conclude that in both cases the supermodel offers better predictions than the raw models. In Fig. 8 we have a global picture of the three supermodels by drawing the PLR of supermodels SUPGODU and SUPDUKL against SUPT. We can now observe that the most suitable model is SUPDUKL (positive PLR).

The plots are starting with the failure number 22th, the 22 interfailure times being necessary to build reasonable PL functions for calculating the weight factors for the supermodels. Finally, we conclude that there are situations when it is not wise to use many models for building a supermodel (Fig. 8), but rather to selectively choose

---

Fig. 2. Comparing medians between three supermodels.

Fig. 3. Weight factor evolution for the five SRGMs which form the supermodel SUPT.
the SRGM with a similar behaviour (DU, KL) to avoid great variations in the median estimate (Fig. 2). Also it is clear that, generally, building a supermodel can offer an improvement of estimates with a relatively small effort (the necessary computational effort to build a supermodel is smaller than the one necessary for estimating the raw SRGMs, by an order of magnitude).

Several ARMA models were also tested. Here we present the cumulative interfailure time (cit) ARMA approach. Using 136 interfailure times we compute 136 numerical values of the cits and we try to obtain predictions for different cits using some ARMA models. The numerical results for the 136 cits are presented in Table 1. Here the prediction for the 136\textsuperscript{th} cit is printed, using the first 135 values. Table 2
Fig. 6. PLR between the two SRGMs GO, DU and their corresponding supermodel.

Fig. 7. PLR between the two SRGMs KL, DU and their corresponding supermodel.

presents similar results when the data set with 832 values is considered. Here the next cit is printed.

In the above tables $q = 0$ corresponds to the AR modelling and the remaining entries in the table are for the general ARMA model.

4. CONCLUSIONS AND FUTURE DEVELOPMENTS

In this paper some strategies for software reliability forecasting were considered. The supermodel approach and ARMA modelling are considered. More testing is necessary for different projects, but the
Improving software reliability forecasting

Fig. 8. PLR between the two supermodels each made of two SRGMs and the supermodel made of five SRGMs.

Table 1. Using 135 interfailure times

<table>
<thead>
<tr>
<th>p - q</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>86618</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>86614</td>
<td>86664</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>86797</td>
<td>86025</td>
<td>87288</td>
</tr>
<tr>
<td>4</td>
<td>86661</td>
<td>85266</td>
<td>87704</td>
</tr>
<tr>
<td>5</td>
<td>86541</td>
<td>85458</td>
<td>86392</td>
</tr>
<tr>
<td>Average</td>
<td>86646</td>
<td>85853</td>
<td>87161</td>
</tr>
<tr>
<td>std. dev.</td>
<td>37.87</td>
<td>272.4</td>
<td>336.8</td>
</tr>
</tbody>
</table>

Table 2. Using 832 interfailure times

<table>
<thead>
<tr>
<th>p - q</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.166729E + 7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2.166239E + 7</td>
<td>2.165606E + 7</td>
</tr>
<tr>
<td>3</td>
<td>2.165862E + 7</td>
<td>2.165615E + 7</td>
</tr>
<tr>
<td>4</td>
<td>2.165936E + 7</td>
<td>2.166886E + 7</td>
</tr>
</tbody>
</table>

preliminary results are encouraging. The ARMA approach for the prediction of the next cumulative interfailure time shows that software reliability modelling continues to be an interesting field when classical prediction theory can be applied.

Multiple regression, nonlinear regression and neural networks must be investigated, but a collection of real data from software developers is necessary.

Acknowledgement—This work was supported by NATO under Grant HTECH.LG 941434.

REFERENCES