32

Software Reliability Models for Practical Applications

M. Xie

Dept of Industrial & Systems Engineering, National University of Singapore, Kent Ridge, Singapore 0511

Abstract

Reliability is one of the most important aspects of product quality. Software reliability research has been going on worldwide and many software reliability models are proposed and studied. In this paper, we first review some of the important software reliability models and then discuss them with respect to their validity and usefulness. The final part of this paper is aimed to present some simple models for the analysis of software failure data. Unlike the conventional approach, we present our model based on the graphical interpretation and this approach allows us to easily test a few models before detailed study based on a selected model is carried out.

Keyword Codes: D.2.0; D.2.5; D.2.8
Keywords: Software Engineering, General; Software, Metrics; Reliability

1. INTRODUCTION

Software reliability, which deals with the study of software related failures and their consequences, is an important aspect of software quality. All products have to be working properly without failure before talking about other quality measures. Software reliability has recently attracted many researchers and practitioners because of the increasing demand of high quality software.

Reliability for both software and hardware is formally defined as the probability that a product will be working without failure during a specified time period and under a specified working environment. However, as software systems are different from the hardware counterparts, conventional technology cannot be applied in software engineering directly and specific models have to be used in the study of the reliability of software [1-4].

The aim of this paper is two-fold. First, we intend to give an overview of some existing software reliability models and their applicability in practice. We mainly discuss model validity and usefulness. Second, we present some recently developed models for software reliability study. Unlike the conventional approach, we present our model based on the graphical interpretation and this approach allows us to easily test a few models before detailed study based on a selected model is carried out.

2. SOFTWARE RELIABILITY MODELS AND THEIR USEFULNESS

2.1. Failure time based models

Reliability of software cannot be predicted without some real information about the software system we are interested in. An important and useful type of information is the failure time data collected during the testing stage. Many software reliability models are developed for the estimation of software reliability based on failure time data. Some well-known models are the exponential model, the logarithmic model and the s-shaped model.

M. Lee et al. (eds.), *Software Quality and Productivity*

These models are commonly known as nonhomogeneous Poisson process (NHPP) models and the basic assumption is that the failure process can be described by an NHPP. Using different mean value function describing the behaviour of the failure process, the reliability can be predicted.

Denote by $N(t)$ the cumulative number of failures at time t, the NHPP model assumes that the counting process $\{N(t), t\geq 0\}$ has independent increment and $N(t)$ is Poisson distributed with mean value function $m(t)$.

It follows from the standard theory of NHPP that, whenever the mean value function is determined, the reliability function at time t_0 is given as

$$R(t|t_0) = exp\{-[m(t+t_0)-m(t_0)]\}.$$

The Goel-Okumoto model [3], which is also called the exponential model, has the following mean-value function

$$m_{GO}(t) = a(1 - e^{-bt}).$$

The delayed s-shaped model is first proposed by Yamada et al. [4] and the mean value function is given as

$$m_{DS}(t) = a(1 - (1 + bt)e^{-bt}).$$

The Musa-Okumoto model [1], which is of logarithmic type, is another useful NHPP model and the mean value function is

$$m_{MO}(t) = a\ln(1+bt).$$

These models have been widely discussed by many authors and can be seen as the representation of the NHPP software reliability models. The parameter a in the Goel-Okumoto model and the s-shaped model can be interpreted as the total number of faults in the software and the parameter b is a type of fault-detection rate. However, for the Musa-Okumoto model, $m(t)$ is not finite. This is useful when we can interpret the failure process in such a way that not all faults can be removed or new faults can be introduced leading to infinite number of faults in infinite time.

It can be shown that the Goel-Okumoto model has a constant fault detection rate while the delayed s-shaped model has an increasing-then-decreasing failure rate and then the mean value function is s-shaped. For the Musa-Okumoto model, earlier failures reduce the failure rate larger than later failures. In conclusion, these models are able to describe different software failure processes. The practical difficulty is to select a suitable model although a lot of research has focused on the problem of parameter estimation.

2.2. Software-metric based reliability model

It can be argued that software reliability depends on the intrinsic property of the software. Software metrics can be used for the estimation of software reliability. There are many software metrics and a lot of study is going on in finding the most suitable metrics for reliability prediction. In a recent book edited by Fenton and Littlewood [5], some related articles can be found.

Historically, it has been observed at the beginning of the seventies, see e.g. Akiyama [6], that there is a strong dependence between the number of faults and the nature of the program, such as the number of codes, the number of paths and other algorithmic complexity measures. Since then, many software metrics are developed and also applied in software reliability study. Software metrics have been widely applied in different contexts, mainly during the design stage for which software metrics are helpful tools in allocating development resources and making management decisions.

Generally, software metrics can be used in reliability prediction. The main advantage of using software metrics is its applicability at earlier stages of the software development. This is of vital importance since it is helpful for software managers to allocate software testing

effort, to decide a preliminary release time and to determine the price of the software product. Empirical software metrics models are simple and they can easily be calculated.

It should be noted that software metrics can only be used to make an overall estimation of software fault content before the testing starts. After the testing, a number of faults are removed and it is important to be able to estimate the number of remaining faults. There are few metrics for making such estimation and this is a problem when using software metrics in estimating software reliability when the prediction is important.

Most existing software metrics, due to their static nature, do not consider the growth of software reliability. A disadvantage of the existing software metrics models is thus that they give only an estimate of the software fault content which is not equal to the probability of software failure, because different faults may have different occurrence rate. An important area of further research is just the relation between the failure intensity of the software and other software metrics or environmental factors which can be used to connect existing static models to other dynamic models.

Existing studies have been concentrated on finding a single metric to give an acceptable estimate of the number of faults. However, this may not be successful and it can be the case that more than one metric will have some impact on the reliability. A better way to utilize several metrics is to combine them using techniques such as multiple-regression analysis, see e.g. [8]. In this respect, more research is needed.

2.3. Combination of information

As discussed previously, both static models and dynamic models are useful. In practice, it is important to combine all available information about the software product and the development process in order to make a better reliability prediction. For example, software complexity measures are useful in estimating the number of software fault content, especially at an earlier stage of software development which is important from a management point of view. The estimate is, however, inaccurate and hence it should be revised as more data have been collected.

In a paper by Yamada, the behaviour of the testing resource expenditures over the testing period can be observed as a consumption curve of testing-effort and the increase in reliability is strongly dependent on the allocated testing-effort. Software reliability models are developed by incorporating some testing-effort models.

A model incorporating various factors related to software testing, such as the number of test workers, is proposed in a paper by Tohma. The applicability of this model and various estimation techniques have also been studied in other papers by Tohma and his co-authors.

The paper by Khoshgoftaar and Munson suggests that software metrics are used in reliability prediction and a neural network approach is used. The approach nicely combines the available software metrics and dynamic software reliability growth models. The model can be trained by including more information and the accuracy of prediction is increased.

Musa [7] presented a theory of software reliability analysis based on the execution time. This well-known software reliability model can be considered a model combining static information with dynamic prediction.

Bayesian estimation is a statistical technique for parameter estimation by combining prior information and the data set. Attempts have been made in using Bayesian techniques in software reliability study. Many Bayesian models have been proposed for the analysis of software failure data in combination with previous knowledge in form of a so-called prior distribution of the unknown parameter.

3. MODELS WITH GRAPHICAL INTERPRETATIONS

Usually, we do not know beforehand which model to use in a particular situation. Even if we have selected a model, we have to determine the parameters of the model. All studies on failure time based models focus on these issues. However, most of the results are difficult to apply in practice. In this section we present a graphical approach which makes the model validation and parameter estimation an easy task.

Usually, the graphical method employs some transformations and finds a linear relationship between the variables of interest. If the required transformation exists, the transformed test data tend to be on a straight line. As a straight line is only characterized by its slope and a point which is through, the number of the unknown parameters in the model with graphical interpretation is limited to one or two. The slope and a point on the straight line, usually the intercept on the vertical axis, provide us the estimates of the parameters [8].

One such model is the Duane model which is a well-known model for reliability growth and it has the following mean value function

$$m_{DU}(t) = at^b,\ t \geq 0.$$

The Duane model has widely been discussed and applied by practitioners in analysing failure data of repairable systems. In software reliability, the Duane model has also been discussed from various points of view. However, there is little discussion about its graphical interpretation.

The log-power model is a recently developed model with graphical interpretation [8] and it has the mean value function

$$m_{LP}(t) = a\ln^b(1+t),\ t \geq 0.$$

Our empirical study on the log-power model shows that the log-power model is able to fit different data sets and has a high prediction ability. It is a very promising model in software reliability modelling.

From the relationship

$$\ln m_{LP}(t) = \ln a + b\ln\ln(1+t),\ t \geq 0$$

we see that if we plot $\ln m(t)$ versus $\ln\ln(1+t)$, the plot should be a straight line. The parameter b is the slope of the straight line and the constant term is just the parameter $\ln a$.

In fact, we can validate a specific model by looking at the fitted line before further analysis is made. If the data cannot be fitted by a straight line, there is then no need to analyze the data and other models should be considered. This *FIRST-MODEL-VALIDATION-THEN-PARAMETER-ESTIMATION* technique is then very simple and convenient for practical applications.

REFERENCES

1. Musa, J.D.; Iannino, A. and Okumoto, K.: '*Software Reliability: Measurement, Prediction, Application*', McGraw-Hill, 1987.
2. Xie, M.: '*Software Reliability Modelling*', World Scientific Publisher, 1991.
3. Goel, A.L. and Okumoto, K.: 'Time-dependent error-detection rate model for software reliability and other performance measures', *IEEE Trans. on Reliability*, 1979, R-28, pp.206-211.
4. Yamada, S. and Osaki, S.: 'Reliability growth models for hardware and software systems based on nonhomogeneous Poisson process: A survey', *Microelectronics and Reliability*, 1983, 23, pp.91-112.
5. Fenton, N.E. and Littlewood, B.: '*Software Reliability and Metrics*'. Elsevier, London.
6. Akiyama, F.: 'An example of software system debugging', *Proc. IFIP Congress, Vol.1*, North-Holland, Amsterdam, 1971, pp.353-359.
7. Musa, J.D.: 'A theory of software reliability and its application', *IEEE Trans. on Software Engineering*, 1975, SE-1, pp.312-327.
8. Xie, M. and Zhao, M.: 'On some reliability growth models with simple graphical interpretations', *Microelectronics and Reliability*, 1993, 33, pp.149-167.