Software Reliability for Multi-Type Defects: Applications to Modern Bug Databases

Vignesh T Subrahmaniam, Anup Dewanji and Bimal K Roy
GE Global Research, Bangalore and Indian Statistical Institute, Kolkata

1 Introduction

Traditionally, software reliability models have focused on estimating the reliability of the software based upon data generated from in-house testing where the testing methodology and reporting of defects is strictly controlled. These reliability models cannot be directly applied to data retrieved from bug databases as they contain defects recorded due to uncontrolled software usage and reporting of software defects. The usage rate of a software is typically a function of time and is in general unknown. The reporting pattern also depends on the type of users. There is a tendency on the part of an average user to discover behaviour defects more often as compared to, say, critical security defects. Such tendencies can complicate the developement of a software reliability model.

In this paper, we use the classification of user-reported software defects into multiple types (based, for example, on their severity) to formulate and estimate a software reliability model that is non-parametric with respect to the usage rate and takes into account differences in reporting rates of different types of defects. A partial likelihood approach is used for model estimation. Based upon the proposed model, reliability metrics are proposed that do not depend upon the usage rate.

Software defects can be classified into multiple types and certain defect types are of great importance to the software community, an example being defects which are related to security loopholes in the software (Musa 2005 pp. 198). Software reliability models can be expected to answer questions such as "Whether newer versions of a software are more reliable with respect to security faults when compared to older versions?" Such inference based on bug databases needs caution as multiple factors can confound the infer-

ence of a statistically significant result. Usage rates can confound statistical inference because a version of software that is used more often is bound to have more defects reported as compared to a version that is used less often. A more subtle confounding effect is due to the differences in the discovery and reporting rates of different types of defects.

Existing software reliability models incorporating time-varying reporting rates of defects either assumed a NHPP model with a parametric form for the defect reporting rate (See Singpurwalla and Wilson 1999 for a survey of such models), or considered a non-parametric method (Wang et. al., 2007). The non parametric method may be more appropriate for user-driven defect discovery when compared to parametric models, but an appropriate reliability metric is difficult.

The approach presented in this article is different from existing models as it uses the classification of defects into multiple types and introduces the usage rate as a nonparametric time dependent component of the model. Also, the notion of reliability that we will develop relates to the incidence of one or more special types of defects (e.g., security or crash related defects) instead of software failures due to defects assumed to be equally important irrespective of their types. Unequal discovery and reporting rates for different defect types are modeled through the use of multivariate counting process with different intensity functions. The effect of time dependent usage rates is incorporated through a proportionality model of the intensity functions.

2 The Modeling and the Method

Consider data related to all new defects reported up to a certain calendar time S from the release date of a software. This may be represented as the sequence of tuples $(T_1, Z_1), \ldots, (T_n, Z_n)$, where T_i is the reporting time of the i^{th} new defect reported since the release of the software and Z_i is the type of the defect, for $i = 1, ..., n$, where n is the number of new defects reported till time S. Suppose there are m special types of defects, denoted by $1, ..., m$ and all other types are pooled into one "baseline" type denoted by 0, then Z_i takes values in $\{0, 1, \cdots, m\}$. Alternatively, such a data set can be represented by the sequence of tuples $(T_1, \mathbf{N}(T_1)) \ldots, (T_n, \mathbf{N}(T_n))$, where the vector $\mathbf{N}(t) = [N_0(t), N_1(t), \ldots, N_m(t)]$ denotes the multivariate counting process with $N_i(t)$ being the number of new defects of type i reported upto and including time t, for $i = 0, 1, \cdots, m$.

The intensity $\lambda_i(t)$ of discovering a new software defect at time t is a function of the number of defects already discovered up to time t along with the type of defect and usage rate of the software. Suppose, for each $N_i(t)$, there is an underlying intensity function $\gamma_i(t)$ depending only on the usage rate and reporting rate of the ith type of defect. The intensity $\gamma_i(t)$ may be calibrated to model the effect of the history of the process $\mathbf{N}(t)$ on the intensity process $\lambda_i(t)$ through a proportional intensity model as given by

$$\lambda_i(t) \quad = \quad \gamma_i(t) f_i(\mathbf{N}(t-)), \text{ for } i = 0, 1, \cdots, m, \tag{1}$$

where $\gamma_i(t)$ is considered to be unknown and arbitrary and $f_i() \geq 0$ with $f_i(\mathbf{0}) = 1$. We intend to model $f_i(\mathbf{N}(t))$ parametrically leading to a semi-parametric model for $\lambda_i(t)$. In particular if $f_i(\mathbf{N}(t))$ depends on $\mathbf{N}(t)$ only through $N_i(t)$, the component processes $N_i(t)$'s are independent. Note that the set of functions $\{f_i(\mathbf{N}(t)), i = 0, 1, \cdots, m\}$ can be used to obtain reliability metrics that are agnostic to the usage and reporting rates of the defects. A further simplification can be achieved by assuming that each $\gamma_i(t)$ is proportional to a common rate parameter $\gamma(t)$ across all types of defects. This assumption greatly simplifies the estimation of model parameters apart from enabling inference on the reliability with respect to a given type of defect. Therefore, with $\gamma_0(t) = \gamma(t)$, we may assume

$$\gamma_i(t) = \gamma(t)e^{\alpha_i}, \text{ for } i = 1, \cdots, m. \tag{2}$$

There may be several choices for $f_i(\mathbf{N}(t))$. In this context, one may recall the Jelinski and Moranda (1972) model for software testing data, which postulates a linear decrease in the intensity of reporting a new defect with the number of already detected defects, and consider a linear decrease model as given by

$$f_i(\mathbf{N}(t)) = 1 - \sum_{j=0}^{m} \beta_{ji} N_j(t). \tag{3}$$

.

Alternatively, a non-linear decrease model in the spirit of the logarithmic Poisson model (Musa and Okumoto, 1975), as given by

$$f_i(\mathbf{N}(t)) = exp\left(-\sum_{j=0}^{m} \beta_{ji} N_j(t)\right) \tag{4}$$

may also be considered. In the special case, when the component processes $N_i(T)$ are indepedent, it would be appropriate to consider $f_i(\mathbf{N}(t)) = exp(-\beta_i N_i(t))$ with a scalar β_i, for $i = 0, 1, \ldots, m$.

Let us write $T^{(j)} = (T_j, T_{j-1}, \ldots, T_1)$ and $Z^{(j)} = (Z_j, Z_{j-1}, \ldots, Z_1)$. The likelihood of the data sequence $(T_1, Z_1), \ldots, (T_n, Z_n)$ is proportional to

$$\prod_{j=1}^{n} P_{T_j|T^{(j-1)},Z^{(j-1)}}(.|\alpha, \beta, \gamma(.)) \times \prod_{j=1}^{n} P_{Z_j|T^{(j)},Z^{(j-1)}}(.|\alpha, \beta). \tag{5}$$

The second product in the above equation, under the modeling assumptions (1) and (2), does not depend on $\gamma(t)$, and is the partial likelihood for estimating α and β (Cox 1975). Note that the conditional probability $P_{Z_j|T^{(j)},Z^{(j-1)}}(.|\alpha, \beta)$ in the second product can be derived as the following multinomial probability under assumptions (1) and (2):

$$P(Z_j = i|T^{(j)}, Z^{(j-1)}) = \frac{e^{\alpha_i} f_i(\mathbf{N}(T_j-))}{\sum_{k=0}^{m} e^{\alpha_k} f_k(\mathbf{N}(T_j-))}, \tag{6}$$

for $i = 0, 1, \ldots, m$. In the special case given by (4), this partial likelihood simplifies to the likelihood of the multinomial logistic regression model as

given by,

$$\prod_{j=1}^{n} P(Z_j = i|T^{(j)}, Z^{(j-1)}) = \prod_{j=1}^{n} \frac{e^{\alpha_i - \beta_i' \mathbf{N}(T_j-)}}{\sum_{k=0}^{m} e^{\alpha_k - \beta_k' \mathbf{N}(T_j-)}}. \quad (7)$$

Maximizing the partial likelihood to estimate α and β can now be performed through a multinomial logistic regression between Z_j and the vector $\mathbf{N}(T_j-)$. A Newton-Raphson procedure can be used to maximize the log partial-likelihood in order to estimate the parameters. The asymptotic normality of the partial likelihood estimates (Wong 1986) can be used to perform tests of significance and obtain confidence intervals for the parameters of interest and functions thereof.

We propose two new reliability metrics that do not depend upon the usage rate. The first metric is the probability of discovering no crash related defect (corresponding to, say, type m) in the next N defects of any type. Let us denote this by $R(N)$. This metric would not depend upon the usage rate of the software. The second metric is the expected number of defects to be observed before observing a new crash related defect, denoted by $MNDF$, and use it as an alternative to mean time to failure ($MTTF$). The metrics $R(N)$ and $MNDF$ can be computed as follows

$$R(N) = \prod_{j=n+1}^{n+N} P\left(Z_j = 0|T^{(j)}, Z^{(j-1)}; \alpha, \beta\right). \quad (8)$$

Note that $Z^{(j-1)}$ in the conditioning event in each term of (10) is given by $(Z_1, \ldots, Z_n, Z_{n+1} = 0, \ldots, Z_{j-1} = 0)$. Also, each such term is independent of $T^{(j)}$, except through $\mathbf{N}(T_j)$ as in (8). In this case, the other reliability measure MNDF can be calculated as the infinite sum

$$MNDF = \sum_{l=n+1}^{\infty} \prod_{j=n+1}^{l} P\left(Z_j = 0|T^{(j)}, Z^{(j-1)}; \alpha, \beta\right). \quad (9)$$

Theoretical derivations of these reliability metrics in general can be a challenge because of the stochastic nature of the model (1). For example, the probability of no crash related defect requires a huge sum of m^N joint probability terms each corresponding to an ordered set of N defects of types other than crash related defect.

3 Analysis of Python Software

Python is a general purpose scripting language that is extensively used in a variety of applications. It is an open source software which is maintained and developed by it's community. The Python project maintains a bug database that records defects in the software reported by its user community. Python's bug database provides a method for querying its database (http://bugs.python.org/issue?@template=search). Only those defects whose resolution was "fixed" or "fixed and accepted" as on 31 January 2012 are retrieved. Table 2 contains a summary of software defects for Python versions 2.7 and 2.6 between their release and 31 January 2012.

Table 1: Summary of defects in the two versions of Python

	Python 2.7		Python 2.6	
Defect Type	Count	Percentage	Count	Percentage
Crash	130	5.7 %	124	6.3 %
Security	19	0.8%	16	0.8%
Others	2124	93.5 %	1835	92.9%
Total Bugs	2273	100%	1975	100 %

For our illustration of a simplified analysis analysis with two types of defects, we consider "Crash" as the defect type of importance ($i = 1$) and pool securiy related defects with other types ($i = 0$). The independent logarithmic Poisson model given by $f_i(\mathbf{N}(t)) = exp(\alpha_i - \beta_i N_i(t))$, for $i = 0, 1$, with $\alpha_0 = 0$, is once again used. As in the three-type analysis, we only need the vector $Z^{(n)} = (Z_1, \ldots, Z_n)$, where each Z_i takes values 0 or 1. The estimates of the parameters and the reliability metrics $R(N)$, for $N = 10$, and $MNDF$, along with their standard errors are presented in the bottom panel of Table 2.

Table 2: Estimates of the parameters and reliability metrics with their standard errors.

		Python 2.7		Python 2.6	
Analysis	Parameters	Estimate	SE	Estimate	SE
Two-type	α_1	3.85	0.37	3.56	0.33
	β_0	0.08	0.02	0.04	0.02
	$\beta_1(\times 10^2)$	0.55	0.18	0.35	0.15
	$R(N)$	0.44	0.07	0.47	0.07
	$MNDF$	11.23	2.09	12.28	2.26

From the reliability metrics $R(N)$ and $MNDF$ values, Python 2.6 seems more reliable with respect to "crashes" when compared to Python 2.7. However, by looking at the standard errors, we cannot conclude that the reliabilities of the two version of the software are significantly different.

4 Concluding Remarks

A limited simulation stidy using the independent logarithmic Poisson model gives evidence in favor of consistency and asymptotic normality of the parameter and reliability estimates. Interestingly the estimated reliability metrics seem to have better convergence than the estimated model parameters. Also, the effect of model mis-specification on the estimates of $R(N)$ and $MNDF$ seems to be minimal.

The proposed reliability metrics are easily interpretable and can be used to compare different software versions. For example, in the analysis of Python software, we gather from the estimate of $MNDF$ for Python 2.7

that, on an average, approximately, 12 non-crash related defects will be discovered before the discovery of a crash related defect, which is about the same when compared to Python 2.6.

The integrated intensity function $\Gamma(t) = \int_0^t \gamma(u)du$ (See Section 2) may be estimated using a Breslow type estimator (Breslow 1972), as given by, for the logarithmic Poisson model,

$$\hat{\Gamma}(s) = \int_0^s \frac{dN_{\cdot}(t)}{\sum_{i=0}^m exp(\hat{\alpha}_i - \hat{\beta}_i N_i(t-))} dt, 0 < s \leq S, \tag{10}$$

where $N_{\cdot}(t) = \sum_{i=0}^m N_i(t)$. Extrapolation of the estimated $\gamma(t)$ through a parametric model may be used to compute time based reliability metrics (Wang et al. 2007). This type of estimator may also be used for testing the proportionality assumption (2). For example, the individual integrated intensity $\Gamma_i(t) = \int_0^t \gamma_i(u)du$ may be estimated by

$$\hat{\Gamma}_i(s) = \int_0^s \frac{dN_i(t)}{exp(-\hat{\beta}_i N_i(t-))} dt, 0 < s \leq S, \tag{11}$$

for $i = 0, \ldots, m$. Plots of $log(\hat{\Gamma}_i(t))$ for different i on the same graph should be near parallel if (2) is true.

References

Breslow, N. (1972), Comment on D. R. Cox (1972) paper, Journal of the Royal Statistical Society: Series B, 34, 216-217.

Cox, D. R. (1975). Partial Likelihood, Biometrika, Vol. 62, No. 2, 269-276.

Goel, A.L. and Okumoto, K. (1978). An Analysis of Recurrent Software Failures on a Real-time Control System, . Proc. ACM Annu. Tech. Conf., Washington D.C., 496-500.

Jelinsky, Z. and Moranda P. B. (1972). Software Reliability Research. In Statistical Computer Performance Evaluation, ed. Freiberger W., 465-484.

Singpurwalla, N. D. and Wilson, S. P. (1994). Software Reliability Modeling, International Statistical Review, Vol. 62, No. 3, 289-317.

Wang, Z., Wang, J. and Liang, X. (2007). Non-parametric Estimation for NHPP Software Reliability Models, Journal of Applied Statistics Vol. 34, No. 1, 107–119.

Wong, W. H. (1986). Theory of Partial Likelihood, Annals of Statistics, Vol. 14, No. 1, 88-123.