

A Study of Estimation Methods for Defect Estimation

Syed Waseem Haider , João W. Cangussu
Department of Computer Science
The University of Texas at Dallas
{waseem,cangussu}@utdallas.edu

Abstract

Accurate defect prediction is useful for planning and management of software testing. We discuss here the statistical characteristics and requirements of estimation methods available for developing the defect estimators. We have also analyzed the models, assumptions and statistical performance of three defect estimators available in the field using data from the literature. The study of the estimation methods here is general enough to be applied to other estimation problems in software engineering or other related field.

1. Introduction

The major goal of a software testing process is to find and fix, during debugging, as many defects as possible and release a product with a reasonable reliability. A trade-off between releasing a product earlier or investing more time on testing is always an issue for the organization. The clear view of the status of the testing process is crucial to compute the pros and cons of possible alternatives. Time to achieve the established goal and percentage of the goal achieved up to the moment are important factors to determine the status of a software testing process. Many defect prediction techniques [1, 2, 3, 6, 11, 14, 15, 16] have addressed this important problem by estimating the total number of defects, which then provide the status of a testing process in terms of remaining number of defects in a software product. The availability of an accurate estimation of the number of defects at early stages of the testing process allows for proper planning of resource usage, estimation of completion time, and current status of the process. Also, an estimate of the number of defects in the product by the time of release, allows the inference of required customer support.

Our goal in this paper is to discuss various estimation methods which are used to develop these defect estimation techniques. We would discuss the assumptions that each method makes about the data model, probability distribu-

tion, and mean and variance of data and estimator. We will also discuss the statistical efficiency of the estimators developed from these estimation methods. There are two broad groups of estimation methods called Classical Approach and Bayesian Approach as shown in Figure 1. If there is prior statistical knowledge about the parameter which we are interested to estimate, then an estimator based on Bayesian Approach can be found. Note that the prior knowledge can be in the form of prior mean and variance and probability distribution of the parameter. There are several Bayesian estimator available [4, 5, 9]. Main advantage of Bayesian estimators is that they provide better early estimates compared to estimators developed from Classical Approach. In the initial phase of software testing not enough test data is available, consequently in the absence of prior knowledge initial performance of estimator suffers.

Organization of the paper is as follows, due to the brevity of space we will limit the discussion of estimation methods to Classical Approaches only in Section 2. We discuss in Section 3 three defect estimators available in the field. We conclude the paper in Section 4.

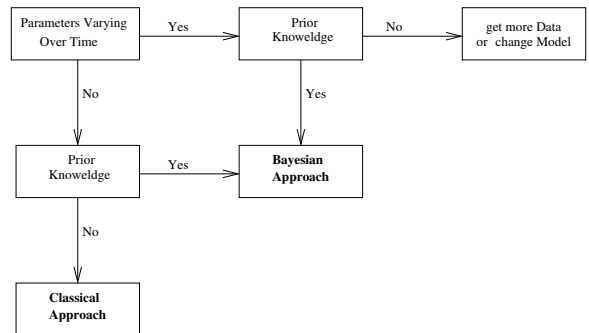


Figure 1. Estimation Approaches.

2. Classical Approach

In this Section we discuss which estimation methods are available and what are the requirements of each method.

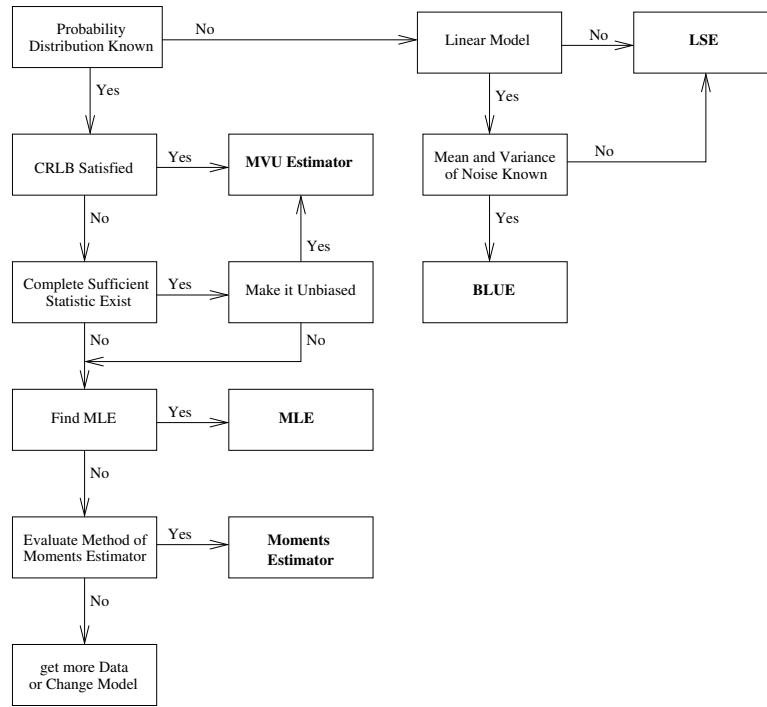


Figure 2. Hierarchical classification of classical approaches for estimators.

When the requirements are met, then the issue to be addressed is how to develop an estimator using this method. We will also compare the performance of each method in order to provide the merits and demerits of the estimators based on these methods. Figure 2 provides a hierarchical organization of these methods. Collectively these estimation methods as shown in Figure 2 are called Classical Approach [7].

During system testing we are interested in finding in advance the total number of defects which will be discovered when the system testing is completed. As pointed out in Section 1 the knowledge of the total number of defects in advance will help us in scheduling testing tasks, management of teams and allocation of resources to meet the target value (total number of defects) in assigned time. Let us use the variable θ to represent the total number of defects to be estimated. Estimation techniques need samples or data from the ongoing system testing process. Sample can be in the form of the number of defects found each day or week or any other time unit. Time unit can be either execution time or calendar time. Sample can also be the total number of defects found by any time instant. In reliability models [2, 6, 11, 15] samples are usually the number of defects discovered per execution time but models for calendar time have also been proposed [6, 12]. Authors have proposed the ED^3M (Estimation of defects based on Defect Decay Model) [3] model which can be used for both execution and

calendar time without any change.

These samples must be related to the parameter θ which is to be estimated. Generally a data model is developed which relates data or samples to θ . Data model must also account for random noise or unforeseen perturbations which naturally occur in any real life phenomenon. In software testing random behavior can be due to work force relocation, noise in the testing data collection process, testing of “complex or poorly implemented” parts of the system, among others. In order to explain the estimation methods given in Figure 2 we provide a simple general data model. Assume that we take an n th sample $x[n]$ which contains the parameter of interest θ corrupted by random noise $w[n]$ as given by

$$x[n] = \theta + w[n] \quad (1)$$

Note that in Eq. 1 $x[n]$ is linearly related to θ . The observations of θ made at N intervals is given by the data set $x[0], x[1], \dots, x[N-1]$. A more generalized linear data model is

$$\mathbf{x} = \mathbf{h}\theta + \mathbf{w} \quad (2)$$

where $\mathbf{x}_{N \times 1}$ is the data vector, $\mathbf{h}_{N \times 1}$ is the observation vector, θ is the parameter of interest, and $\mathbf{w}_{N \times 1}$ is the noise vector. Note that \mathbf{h} can contain information such as number of testers, failure intensity rate, number of rediscovered

faults, etc. for each sample. A linear data model is recommended for two reasons: first it is more likely to provide a closed form solution and second its more efficient as will be discussed later. The joint probability distribution of data is given by $p(x[0], x[1], \dots, x[N-1]; \theta)$ or simply in vector form $p(\mathbf{x}; \theta)$. This PDF (Probability Density Function) is a function of both data \mathbf{x} and the unknown parameter θ . For example if unknown parameter θ changes to θ_1 then the value of $p(\mathbf{x}; \theta_1)$ for the given data set will change. When PDF is seen as the function of unknown parameter θ it is called likelihood function. Intuitively PDF provides how accurately we can estimate θ .

If the probability distribution of the data is known, then the problem of finding an estimator $\hat{\theta}$ is simply finding a function of data (as given by Eq. 6) which gives estimates of the parameter. Many defect estimators [1, 2, 3, 6, 11, 14, 15, 16] assume the probability distribution for the data. The variability of estimates determines the efficiency of the estimator. The higher the variance of the estimator the less effective (or reliable) are the estimates. Hence various estimators can be found for the data but the one with the lowest variance is the best estimator. Another important characteristic of the estimator is that it must be unbiased, i.e. $E[\hat{\theta}] = \theta$.

There are various methods available to determine the lower bound on the variance of the estimators, e.g. [8, 10], but Cramer-Rao Lower Bound (CRLB) [7] is easier to determine. CRLB Theorem states: *it is assumed that the PDF $p(\mathbf{x}; \theta)$ satisfies the regularity condition*

$$E\left[\frac{\partial \ln p(\mathbf{x}; \theta)}{\partial \theta}\right] = 0, \quad \forall \theta \quad (3)$$

where the expectation is taken with respect to $p(\mathbf{x}; \theta)$. Then, the variance of any unbiased estimator $\hat{\theta}$ must satisfy

$$VAR(\hat{\theta}) \geq \frac{1}{-E\left[\frac{\partial^2 \ln p(\mathbf{x}; \theta)}{\partial \theta^2}\right]} \quad (4)$$

□

An estimator which is unbiased, satisfies the CRLB theorem, and is based on a linear model is called an efficient Minimum Variance Unbiased (MVU) estimator [7]. In other words it efficiently uses the data to find the estimates. An estimator whose variance is always minimum when compared to other estimators but is not less than CRLB is simply called MVU estimator. An estimator based on linear data model is more likely to attain CRLB and hence to provide efficient MVU estimator. The efficient MVU estimator and its variance is given by Eqs 6 and 7 respectively.

$$\frac{\partial \ln p(\mathbf{x}; \theta)}{\partial \theta} = I(\theta)(g(\mathbf{x}) - \theta), \quad (5)$$

$$\hat{\theta} = g(\mathbf{x}) \quad (6)$$

$$VAR(\hat{\theta}) = \frac{1}{I(\theta)} \quad (7)$$

A defect estimator which is an efficient MVU estimator will provide very accurate estimates. In other fields such as signal processing and communication systems where the system or model under investigation is well defined in terms of physical constraints, it is possible to find an efficient MVU estimator. But in software engineering no model completely captures all the aspects of a software testing process. Different models [1, 2, 3, 6, 11, 14, 15, 16] are based on different assumptions and this lack of consistency hints towards the absence of a mature testing model. Therefore its unlikely to find an efficient MVU estimator.

It is possible that for a given data model CRLB cannot be achieved. In other words a solution similar to the one given by Eq. 5 may not exist. Another approach to find an MVU estimator is based on finding sufficient statistic such that minimal data is required to make PDF of data $p(\mathbf{x}; \theta)$ independent of unknown parameter θ . As discussed earlier $p(x[n]; \theta)$ is dependent on both data $x[n]$ and θ . For example a simple estimator $\hat{\theta} = x[n]$ will have high variance in estimating θ . On the other hand if sufficient data $x[0], x[1], \dots, x[N-1]$ is available then new data points will not provide additional information about θ . For example if $x[N]$ is a new sample after sufficient statistic has been acquired then the conditional PDF $p(x[N] | x[0], x[1], \dots, x[N-1]; \theta)$ is independent of θ as given by 8.

$$\begin{aligned} p(x[N] | \mathbf{x}[0], \mathbf{x}[1], \dots, \mathbf{x}[N-1]; \theta) = \\ p(x[N] | \mathbf{x}[0], \mathbf{x}[1], \dots, \mathbf{x}[N-1]) \end{aligned} \quad (8)$$

If the sufficient statistic exist then according to Neyman-Fisher Factorization Theorem [7]

$$p(\mathbf{x}; \theta) = g(T(\mathbf{x}), \theta)h(\mathbf{x}) \quad (9)$$

where $T(\mathbf{x})$ is the sufficient statistic. If $p(\mathbf{x}; \theta)$ can be factorized as given by Eq. 9 then sufficient statistic $T(\mathbf{x})$ exists. Then we have to find a function of $T(\mathbf{x})$ which is the estimator $\hat{\theta}$ such that $\hat{\theta}$ is unbiased $E[\hat{\theta}] = \theta$.

$$\hat{\theta} = f(T(\mathbf{x})) \quad (10)$$

The $\hat{\theta}$ is also an MVU estimator. We can intuitively see that $\hat{\theta}$ will have lower bound on the variance as more new data is not contributing any new information about θ . The PDF for ED^3M [3] can be factorized as given by Eq. 9. The estimator of ED^3M is an unbiased function of $T(\mathbf{x})$ as can be seen from Eq. 25, but we do not claim that the estimator of ED^3M is based on sufficient statistic for the following reason. Software testing is a process which produces new defects along with rediscovered defects. An example of sufficient statistic is that we want to estimate the accuracy of a surgical precision laser so samples of the device are taken. These samples will be 'around' the specified desired value. We take sufficient samples to estimate

the average precision achieved. In software testing as Dijkstra noted, testing shows the presence of defects but not their absence. Even though as time elapses rate of finding new defects subsides significantly but there will be new defects now and then. Overall testing process can be considered an increasing function of defects (in one direction, not ‘around’). Therefore we can only forecast the saturation of finding the defects and not the true number of defects. A change in testing strategy can result in a sudden burst of more defects. Hence an estimator based on sufficient statistic must be aware of this particular behavior of testing process. The notion of sufficient statistic in software testing is arguable. Therefore even though ED^3M fulfills the mathematical requirements of a sufficient statistic estimator, we do not claim that its based on this method.

If we cannot find either sufficient statistic $T(\mathbf{x})$ for θ or an unbiased function $f(T(\mathbf{x}))$ then we will have to resort for an estimator called Maximum Likelihood Estimator (MLE). Many practical estimators proposed in the field for defects estimation are based on MLE, e.g. [3, 12, 14]. MLE is asymptotically (as $N \rightarrow \infty$) an efficient MVU estimator. For linear data model MLE achieves CRLB for finite data set [7]. Another important property is that if an efficient estimator exists MLE will produce it [7]. The basic idea is to find the value of θ that maximizes $\ln p(\mathbf{x}; \theta)$ the log-likelihood function for a given \mathbf{x} . If a closed form solution does not exist a numerical method such as Newton–Raphson can be used to approximate the solution. But the approximation may not necessarily converge to maximization of $\ln p(\mathbf{x}; \theta)$ to produce MLE. An example of numerical approximation of MLE is [6]. Authors were able to find a closed form solution of MLE for ED^3M [3].

Another method to find estimator if either MLE can not be found or is computationally intensive is called Method of Moments. Method of moments estimator is generally consistent [7]. Given $p(\mathbf{x}; \theta)$ if we know that the k th moment of $x[n]$, μ_k is a function of θ as given by Eq. 11

$$\mu_k = E(x[n]^k) = f(\theta) \quad (11)$$

$$\theta = f^{-1}(\mu_k) \quad (12)$$

$$\hat{\mu}_k = \frac{1}{N} \sum_{n=0}^{N-1} x^k[n] \quad (13)$$

$$\hat{\theta} = f^{-1} \left(\frac{1}{N} \sum_{n=0}^{N-1} x^k[n] \right) \quad (14)$$

We approximate the k th moment of data \mathbf{x} , $\hat{\mu}_k$, by taking average of $\mathbf{x}^{(k)}$ as given by Eq. 13. If f is an invertible function as given by Eq. 12 then substitution of approximation $\hat{\mu}_k$ into Eq. 12 results in the estimator $\hat{\theta}$ as given by Eq. 14.

If the probability distribution of the data or in other words of the noise is not known, we can look for only mean and variance of the noise. If these two moments of noise are

known then we can develop an estimator called Best Linear Unbiased Estimator (BLUE) based on two linearity conditions viz., we have linear data model and the estimator itself is a linear function of the data \mathbf{x} . These two linearity conditions are given by Eqs. 2 and 15 respectively. Note that the second condition is necessary to make estimator $\hat{\theta}$ unbiased as given by Eq. 16.

$$\hat{\theta} = \sum_{n=0}^{N-1} a_n x[n] \quad (15)$$

$$E[\hat{\theta}] = \sum_{n=0}^{N-1} a_n E(x[n]) = \theta \quad (16)$$

Although BLUE is a suboptimal estimator because the lower bound of its variance is unknown, it can be successfully used if its variance is in acceptable range and if it is producing results with reasonable accuracy. A limitation of this method from a practical point of view is that we have to find the variance of the noise in software testing. In the present day no detailed study has been done which has investigated the statistical characteristics of noise in testing process. A simple way to approximate the variance of noise is to find the variance of data as given in Eqs. 17 and 18. However, the effects of this approximation on the performance of the BLUE estimator are unknown with respect to software testing.

$$VAR[\mathbf{x}] = VAR[\mathbf{h}\theta + \mathbf{w}] \quad (17)$$

$$VAR[\mathbf{x}] = VAR[\mathbf{w}] \quad (18)$$

If either we do not know the mean and variance of the noise or the data model is not linear we can find an estimator based on the Least Square Error (LSE) approach. The geometrical interpretation of LSE is more intuitive. If we have data points in a space then LSE finds the best curve which minimizes the distance from all these points together. But LSE is prone to outlier points which fall outside the group of points. Such points are usually due to error in the process or other reasons. Due to these points the curve may be found away from the vicinity of points. A simple way is to ignore such points when the data set is input to the LSE estimator. Main advantages of LSE is that no statistical information related to the data set or noise is needed and its simple to develop. On the other hand its optimality is questionable in the absence of statistical information. We have used LSE estimator to approximate the values of λ_1 and λ_2 the defect decay parameters in ED^3M [3]. From the application of ED^3M on several industrial data sets and simulation data sets the performance of LSE estimator of λ_1 and λ_2 was concluded acceptable. Two of the results from the application of ED^3M to industrial data sets are shown in Figures 3 and 4. The two industrial data sets are available in literature [12, 13].

3. Defect Estimators

Many estimators exist for the prediction of number of defects and/or failure intensity. However, due to space constraints we are limited here to the brief description of three estimators which are discussed next. Many reliability models have been proposed which predicts the total number of defects in the software. We will discuss two of these models Padberg's approach [13, 14] based on Hypergeometric Distribution Software Reliability Growth Model (HGDM) and Musa–Okumoto logarithmic Poisson model [6, 12]. We will also discuss ED^3M . All of these approaches are based on MLE where MLE requires assumption about the probability distribution. A detailed study is needed about the probability distributions for software testing. Each distribution is based on some assumptions while ignoring other facts. Hence it can be safely deduced that no model will work in all situations.

Padberg developed a defect estimation technique by assuming that software testing is a random experiment with Hypergeometric distribution. Padberg showed that growth quotient $Q(m)$ of the likelihood function $L(m)$ when greater than 1 indicates that likelihood function is indeed increasing and provides maximum likelihood estimates.

$$Q(m) = \frac{L(m)}{L(m-1)} = \frac{(m-w_1) \dots (m-w_n)}{m^{n-1} \cdot (m-c_n)} \quad (19)$$

In Eq. 19 m is the initial number of faults in the software, w_n is the number of newly discovered and rediscovered faults for n th test and c_n is the cumulative number of faults discovered in n tests. We will briefly define the algorithm for finding MLE, more details can be found elsewhere [13, 14]. For given data c_n first find $x = c_n + 1$ then $Q(x)$. If $Q(x) > 1$ then set $x = x + 1$ and find $Q(x)$ again. Keep repeating the steps until $Q(x) \leq 1$ then MLE will be $\hat{m} = x - 1$. Statistical performance of $Q(m)$ is not discussed in [13, 14]. We do not know if the variance of $L(m)$ is asymptotically bounded by CRLB; in other words we do not know if $Q(m)$ is asymptotically an efficient MVU estimator. Even though the underlying data model is not known it can be observed from $Q(m)$ that the model is nonlinear and the MLE which is based on nonlinear data model for finite data records does not achieve CRLB.

Another example of a defect estimator is the Musa–Okumoto logarithmic Poisson model which is briefly described here. Refer to [6, 12] for more details. The model discussed here is the execution-time model but there is another available for calendar-time. Musa and Okumoto proposed a reliability model based on the assumption that the expected number of faults $\mu(t)$ by time t are poisson distributed as given in Eq. 20. The parameters to be estimated are λ_0 the initial failure intensity and θ the rate of reduction in the normalized failure intensity per failure. We can see

that the data model given by Eq. 21 is a nonlinear function of λ_0 and θ . Hence MLE will not achieve CRLB for finite data set.

$$Pr [M(t) = m] = \frac{[\mu(t)]^m}{m!} e^{-\mu(t)} \quad (20)$$

$$\mu(t) = \frac{1}{\theta} \ln(\lambda_0 \theta t + 1) \quad (21)$$

More over, a closed form solution of MLE could not be found for Eqs. 20 and 21. Therefore a numerical approximation of MLE is needed. Hence whether the approximation of MLE will be asymptotically an efficient MVU estimator is not guaranteed.

Now we discuss ED^3M [3]. The data model of ED^3M is given in Eq. 22 where \mathbf{D} is the defect data vector, \mathbf{h} is the observation vector and \mathbf{w} is noise vector. Vectors are of dimensions $N \times 1$. We have assumed that \mathbf{D} is Normally distributed and the PDF of \mathbf{D} is given by Eq. 24. The initial number of defects in software is given by R_0 . We develop an MLE estimator \hat{R}_0 for R_0 in Eq. 25.

$$\mathbf{D} = R_0 \mathbf{h} + \mathbf{w} \quad (22)$$

$$\text{where } h(n) = 1 - \frac{\lambda_2}{\lambda_2 - \lambda_1} e^{-\lambda_1 n} + \frac{\lambda_1}{\lambda_2 - \lambda_1} e^{-\lambda_2 n} \quad (23)$$

$$p(\mathbf{D}; R_0) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} e^{[-\frac{1}{2\sigma^2}(\mathbf{D}-R_0\mathbf{h})^T(\mathbf{D}-R_0\mathbf{h})]} \quad (24)$$

$$\hat{R}_0 = (\mathbf{h}^T \mathbf{h})^{-1} \mathbf{h}^T \mathbf{D} \quad (25)$$

As seen in Eq. 22 the data model is linear, therefore the MLE estimator \hat{R}_0 in Eq. 25 can achieve CRLB for finite data set and will be an efficient MVU estimator. As discussed in Section 2 \hat{R}_0 also qualifies for sufficient statistic estimator which is an MVU estimator. But we do not claim that \hat{R}_0 is MVU based on sufficient statistic for the reason mentioned in Section 2. More detailed discussion for statistical efficiency of ED^3M is discussed in [3].

Figure 3 shows the comparison between ED^3M and Padberg's approach for the data set given in [13]. ED^3M performs slightly better than Padber's approach. Similarly Figure 4 compares ED^3M with Musa–Okumoto Model for the data set given in [12]. ED^3M clearly outperforms Musa–Okumoto Model.

4. Conclusion

An accurate prediction of total number of software defects helps in evaluation of the status of testing process. But the accuracy of the estimator owes to the estimation method which is used to develop the estimator. We have tried to provide a general framework of available estimation methods. Although we have discussed this framework for defect estimation problem, the discussion is general enough to be

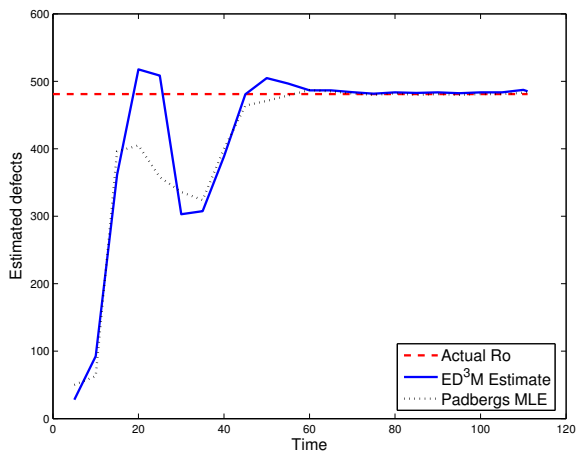


Figure 3. Comparison between ED^3M and Padberg's MLE, where the total number of defects is 481 and the total time length is 111 units.

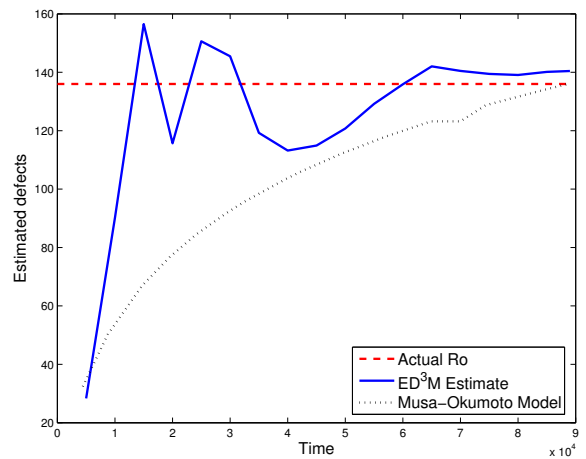


Figure 4. Comparison between ED^3M and Musa-Okumoto model, where the total number of defects is 136 and the total time length is 88.682×10^3 CPU seconds.

used for other estimation problems. We have elicited the requirements of each method. We have also discussed the statistical efficiency that each method offers. Figure 2 summarizes the insightful information in choosing an estimation method for a given estimation problem. Note that even though the discussion is limited to single parameter estimation, it can be easily extended to a vector of parameters to be estimated. In Section 3 we discussed three estimation techniques based on MLE. ED^3M performed better than other two techniques and it is also developed using well defined estimation method as discussed in Section 2. In future we will extend our discussion to Bayesian Approaches and expand the analysis of existing estimators to be more comprehensive.

References

- [1] L. C. Briand, K. E. Emam, B. G. Freimut, and O. Laitenberger. A comprehensive evaluation of capture-recapture models for estimating software defect content. *IEEE Transactions on Software Engineering*, 26(6):518 – 540, June 2000.
- [2] A. L. Goel and K. Okumoto. Time-dependent error-detection rate model for software and other performance measures. *IEEE Transactions on Reliability*, R-28(3):206–211, August 1979.
- [3] S. W. Haider and J. W. Cangussu. Estimating defects based on defect decay model: ed^3m . *IEEE Transactions on Software Engineering*. Accepted.
- [4] S. W. Haider and J. W. Cangussu. Bayesian estimation of defects based on defect decay model: $bayesed^3m$. In *International Conference on Software Engineering and Knowledge Engineering (SEKE)*, July 2006. To appear.
- [5] W. S. Jewell. Bayesian extensions to a basic model of software reliability. *IEEE Transactions on Software Engineering*, SE-11:1465–1471, 1985.
- [6] K. O. John D. Musa. A logarithmic poisson execution time model for software reliability measurement. In *Proceedings of the 7th international conference on Software engineering*, pages 230–238. IEEE, ACM, IEEE Press, March 1984.
- [7] S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall PTR, 1993.
- [8] S. M. Kendall and A. Stuart. *The Advanced Theory of Statistics*, volume 2. Macmillan, New York, 1979.
- [9] B. Littlewood and J. Verrall. A bayesian reliability growth model for computer software. *Journal of the Royal Statistical Society*, 22(3):332–336, 1973.
- [10] R. J. Mcaulay and E. M. Hofstetter. Barankin bounds on parameter estimation. *IEEE Transactions on Information Theory*, 17:669–676, Nov 1971.
- [11] P. Moranda and Z. Jelinski. Final report on software reliability study. Technical Report 63921, McDonnell Douglas Astronautics Company, MADC, 1972.
- [12] Musa, Iannino, and Okumoto. *Software Reliability Measurement, Prediction, Application*. McGraw-Hill, 1987.
- [13] F. Padberg. A fast algorithm to compute maximum likelihood estimates for the hypergeometric software reliability model. In *Second Asia-Pacific Conference on Quality Software*, pages 40–49. IEEE, December 2001.
- [14] F. Padberg. Maximum likelihood estimates for the hypergeometric software reliability model. *International Journal of Reliability, Quality and Safety Engineering*, July 2002.
- [15] S. Yamada and S. Osaki. Software reliability growth modeling: Models and assumptions. *IEEE Transactions on Software Engineering*, SE-11(12):1431–1437, December 1985.
- [16] P. Zhang and A. Mockus. On measurement and understanding of software development processes. Technical Report ALR-2002-048, Avaya Research Labs, November 2002.