

Bayesian Estimation of Defects based on Defect Decay Model: *BayesED³M*

Syed Waseem Haider
Department of Computer Science
University of Texas at Dallas
waseem@utdallas.edu

João W. Cangussu
Department of Computer Science
University of Texas at Dallas
cangussu@utdallas.edu

Abstract

The estimation of the total number of defects at early stages of the testing process helps managers to make resource allocation and deadline decisions. The use of non-bayesian approaches has proven to be accurate but presents a certain latency to achieve a reasonable accuracy. Here we describe BayesED³M, a bayesian estimator construct upon a existing MLE (Maximum Likelihood Estimator) named as ED³M. The case studies presented in this paper show that BayesED³M outperforms ED³M whenever reasonable historical data is available.

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. Unfortunately, many companies do not use any prediction technique and the release date is solely based on a given deadline and not on the status of the product based on the results of the testing process. Under such circumstances a product with a high number of defects may be released incurring high customer support and dissatisfaction. 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 techniques, such as software reliability models [9] and coverage analysis [8], can be used to estimate the status of a testing process. Assume the goal of a testing process is to achieve 100% decision coverage. It is easy to see that a product with only 60% of coverage is far from achieving its goal. However, it does not help to determine the time required to achieve such a goal

much less what would be the consequences of releasing a product with a lower coverage. Reliability models improve on coverage analysis as time to completion can be estimated as well as side effects of releasing a product with a lower reliability. However, reliability models may be very sensitive to operational profile which are not easily estimated.

An alternative measure to compute the status of a testing process is the number of remaining defects in a software product which clearly depends on the estimation of the total number of defects. 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. This paper focuses on the prediction of the total number of defects using Bayesian Estimation. The proposed approach is an extension of ED^3M^1 [6, 7] and addresses the limitations discussed in [6, 7]. The limitations were related mainly to the presence of high noise during the initial phase of system testing. The estimator described here is based upon two assumptions. First assumption is that system testing presents an exponential or s-shaped decay in the number of defects. This assumption is confirmed by the data sets used in this study as well as others available in the literature [3, 4]. Second assumption is that prior knowledge about the parameters which represents the total number of defects in the software is available. This assumption mainly addresses the issue of noise in the early phase of system testing.

The remainder of the paper is organized as follows. A description of the approach proposed here is the subject of Section 2. Case studies with industrial data are presented in Section 3. Related work, and conclusion and future work are presented in Sections 4 and 5 respectively.

¹ ED^3M stands for Estimation of Defects based on the Defect Decay Model

2. The *BayesED³M* Approach

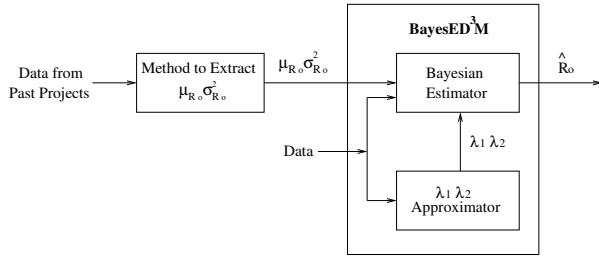


Figure 1. *BayesED³M* is composed of Bayesian Estimator and λ_1 and λ_2 Approximator.

We will now describe the Bayesian Estimation of Defects based on the Defect Decay Model (*BayesED³M*) [2, 6, 7]. As shown in Figure 1 *BayesED³M* is functionally divided in two blocks viz., the “Bayesian Estimator”, and the “ $\lambda_1 \lambda_2$ Approximator”. The output is an estimate \hat{R}_0 of the total number of defects in the software product under test. Both blocks take current data (the data from the project for which we are interested in finding R_0 or the actual number of defects) as input. The data contains records of defects removed, from the software under test, on either daily basis or weekly basis or any other given fixed time unit. We more formally refer to this data as data set. The estimations \hat{R}_0 are evaluated incrementally as more data points become available. We provide another input to the “Bayesian Estimator” block called prior knowledge or prior information in the form of prior mean and prior variance of the parameter R_0 which we are estimating. This parameter represents the estimate of the initial number of defects present in the software product before the start of testing. The prior knowledge extracted from past similar projects supplements the data set during the initial phase of testing in the estimation of R_0 . During the initial phase of the testing the data set does not contain enough information, and the situation is further aggravated by the noise in the testing process. These limitations have been discussed elsewhere [6, 7].

Here, we use a divide and conquer approach by separating the acquisition of prior knowledge from its use and by standardizing it in the form of prior mean and prior variance of R_0 . The acquisition of prior knowledge from past similar projects is under study and more is discussed in Section 5. In this section we propose a Bayesian Estimator which uses the prior knowledge. We discuss the behavior of *BayesED³M* for various values of prior knowledge in Section 3 for a data set acquired from an industrial project.

As stated in [2, 6, 7], the overall behavior of the decay of defects in the system testing phase is assumed to have a

double exponential format as given by Eq. 1.

$$R(n) = R_0 \left(\frac{\lambda_2}{\lambda_2 - \lambda_1} e^{-\lambda_1 n} - \frac{\lambda_1}{\lambda_2 - \lambda_1} e^{-\lambda_2 n} \right) \quad (1)$$

$R(n)$ is the number of remaining defects at day (or any other fixed time interval) n , after removing defects at day $n - 1$. R_0 is the total or initial number of defects present in the software product. We now define in Eq. 2 the random version of Eq. 1 in the form of total number of defects removed from the product under test in n days.

$$D[n] = R_0 \left(1 - \frac{\lambda_2}{\lambda_2 - \lambda_1} e^{-\lambda_1 n} + \frac{\lambda_1}{\lambda_2 - \lambda_1} e^{-\lambda_2 n} \right) + w[n] \quad (2)$$

$D[n]$ is total number of defects removed from the product in n days. Square brackets in $D[n]$ emphasizes that its a random quantity as compared to $R(n)$ which is a deterministic quantity. $w[n]$ is the n th noise component, distributed as Gaussian with zero mean and σ^2 variance, $w[n] \sim \mathcal{N}[0, \sigma^2]$.

The motivation behind the use of a stochastic model is that a deterministic model as defined by Eq. 1 does not account for random variations which are inbred in the testing process. Nonetheless, noise and unforeseen perturbations are common place in the software testing process, there are multiple sources for these variations such as work force relocation, noise in the data collection process, test of “complex” parts of the system, multiple reports of the same error, among others. An effort to individually account for each such factor, will result in a large number of random variables and an inherently very complex mathematical model. The resulting model would get even more complex when the interference of these random variables with each other is incorporated. And still more complicated when samples taken at different instants of time (dataset) interfere with each other as well. A random variable that consists of sum of a large number of small random variables (various sources of noise), under general conditions can be approximated by a Gaussian random variable [6, 7]. Because of this very reason Gaussian distribution finds its use in a wide variety of man made and natural phenomena. Hence we simplify the model by first assuming that, a Gaussian noise identically affects the whole testing process. Moreover, the samples taken at different instants are independent to each other [6, 7].

The derivation of an estimator for Eq. 2 would incur the computation of three parameters R_0 , λ_1 , and λ_2 . The complexity of the solution has led us to simplify the problem by first computing an approximation for λ_1 and λ_2 in the “ $\lambda_1 \lambda_2$ Approximator” block in Figure 1. We summarize here the description of this block, a detailed account is given in [6, 7]. The block approximates the values of λ_1 and λ_2

in a two-step process. The first step is to find the initial values λ_{1i} and λ_{2i} using a technique called Exponential Peeling [6, 7]. The initial values are then input to a nonlinear regression method in the second step. The nonlinear regression, which is implemented using Gauss–Newton method with Levenberg–Marquardt modification [6, 7], results in the values of λ_1 and λ_2 which are fed to the “Estimator” block. The approximated values are then used to compute $h(n)$ as given in Eq. 3.

$$h(n) = 1 - \frac{\lambda_2}{\lambda_2 - \lambda_1} e^{-\lambda_1 n} + \frac{\lambda_1}{\lambda_2 - \lambda_1} e^{-\lambda_2 n} \quad (3)$$

After substituting the value of $h(n)$ in Eq. 2, we write it in the vector form where \mathbf{D} , \mathbf{h} and \mathbf{w} are vectors of dimensions $N \times 1$

$$\mathbf{D} = R_0 \mathbf{h} + \mathbf{w} \quad (4)$$

Now if we consider R_0 to be deterministic or in other words we do not have any prior knowledge about R_0 , then we get the MLE (Maximum Likelihood Estimator) version of \hat{R}_0 or simply ED^3M , as given by Eq. 5. For more details on ED^3M refer to [6, 7].

$$\hat{R}_0 = (\mathbf{h}^T \mathbf{h})^{-1} \mathbf{h}^T \mathbf{D} \quad (5)$$

On the other hand if we do have some prior knowledge about R_0 then we can define a Bayesian version of \hat{R}_0 or simply $BayesED^3M$. We now name the two pieces of prior knowledge, i.e. prior mean and prior variance as μ_{R_0} and $\sigma_{R_0}^2$. Assuming that R_0 is also Gaussian distributed we can write more formally as $R_0 \sim \mathcal{N}[\mu_{R_0}, \sigma_{R_0}^2]$. The Bayesian Estimator of \hat{R}_0 is given by Eq. 6

$$\hat{R}_0 = \mu_{R_0} + \sigma_{R_0}^2 \mathbf{h}^T (\sigma_{R_0}^2 \mathbf{h} \mathbf{h}^T + \sigma_n^2 \mathbf{I})^{-1} (\mathbf{D} - \mu_{R_0} \mathbf{h}) \quad (6)$$

where $\sigma_n^2 = \frac{\sigma^2}{n}$.

3. Case Studies

In this section we explain how the Bayesian estimator behaves and its dependency on the prior knowledge of the parameter to be estimated. Note that prior knowledge is required in the form of prior mean μ_{R_0} and prior variance $\sigma_{R_0}^2$. In the following paragraphs we will discuss the relationship of prior mean μ_{R_0} and prior variance $\sigma_{R_0}^2$, we will analyze the behavior of $BayesED^3M$ for different values of μ_{R_0} over a large fixed interval of $\sigma_{R_0}^2$, such that the minimum value of $\sigma_{R_0}^2 \ll \sigma_N^2$ and maximum value of $\sigma_{R_0}^2 \gg \sigma_N^2$. Note that σ_N^2 is $\frac{\sigma^2}{N}$ at $n = N$, where N is the total size of the data set. The data set used in this section has been acquired from a large industrial project and due to

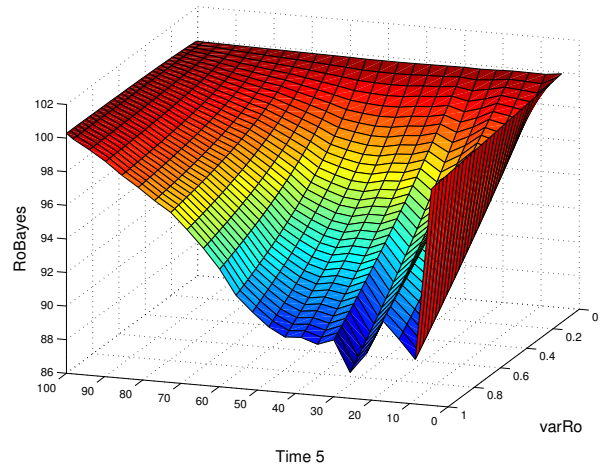


Figure 2. Results of applying $BayesED^3M$ with prior mean μ_{R_0} set to actual number of defects.

proprietary reasons the actual number of defects has been normalized to 100.

Figure 2 shows that if the prior knowledge or the prior mean μ_{R_0} is very close to the actual number of total defects R_0 , then it means that the historical data is gathered from past projects using a very efficient method. Naturally our confidence on such method, historical data, and the resulting knowledge shall be very high. This confidence is shown by the low prior variance $\sigma_{R_0}^2$. In Figure 2 we have kept the prior mean μ_{R_0} constant and plotted the behavior of $BayesED^3M$ for a range of $\sigma_{R_0}^2$. The surface on the far side which is constant or does not change over the whole course of time is due to the smallest $\sigma_{R_0}^2$ hence the highest degree of confidence. This behavior is natural. When this project, by all means, has very similar characteristics (same development team, similar problem, etc.) as one or more past projects from which the prior knowledge is acquired, then a proportional number (based, for example on average defect density for the projects under consideration) of defects is expected. As we increase $\sigma_{R_0}^2$, the dependency of the estimator on the current data increases and hence the resultant curve drops below the level of prior mean μ_{R_0} . But soon picks up back because $\sigma_{R_0}^2$ is not much smaller than variance of the data σ_n^2 . The more we increase the $\sigma_{R_0}^2$ the more the value of the estimator drops and the longer it takes to come back to the actual R_0 . This behavior depicts the interplay of the two variances $\sigma_{R_0}^2$ and σ_n^2 for a constant μ_{R_0} . Note that because of the high accuracy of prior knowledge and consequently the high degree of confidence, even at the highest value of $\sigma_{R_0}^2$ the results of the estimator does not drop below 86% of actual R_0 , which is an optimal lower bound. Hence in this case $BayesED^3M$ is operating in

optimal range and outperforms other contemporary estimators as discussed in Section 4.

Now, let us assume a scenario where prior knowledge is not as accurate as before. In Figure 3 we provide $BayesED^3M$ with prior mean μ_{R_0} which is 80% of the actual number of total defects R_0 . In the case of very low prior variance $\sigma_{R_0}^2$ $BayesED^3M$ will simply output the input through out the length of time, by over shadowing the data set from the current project. Which means that we had over confidence (very low prior variance $\sigma_{R_0}^2$) on not so accurate prior knowledge (μ_{R_0}) and $BayesED^3M$ is simply showing this contradiction by not converging to the actual R_0 . This result also depicts the inefficiency of the method used to collect such prior data. Nonetheless, the balance between the accuracy of the prior knowledge and the confidence in the historical data and the method is achieved when we increase the $\sigma_{R_0}^2$. In this case $BayesED^3M$ starts with reasonable estimate and then quickly adapts to the current data to produce the accurate estimate.

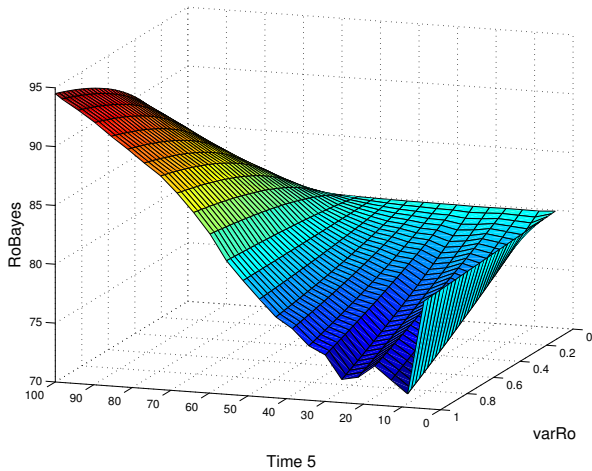


Figure 3. Results of applying $BayesED^3M$ with prior mean μ_{R_0} set to 80% of the actual number of defects.

In general we can conclude that when an initial estimate of μ_{R_0} is within the range of $\pm 20\%$ of the actual number of defects, $BayesED^3M$ converges to a reasonable estimation in a fast manner. However, the value of the variance should be inversely proportional to the confidence in the initial estimate. That is, the higher the confidence the smaller should be the variance. Conflicting data will produce misleading results. That is, having a high confidence and consequently a small variance on prior knowledge that is not accurate, results in over shadowing the data from the current project and consequently producing inaccurate results.

4. Related Work

Estimation theory and other techniques have already been applied to predict the number of software defects present in a software product. We will provide a quantitative comparison between $BayesED^3M$ and ED^3M , other techniques [10, 11] have already been quantitatively compared with ED^3M in [6, 7] therefore a transitive comparison can be drawn between $BayesED^3M$ and these other techniques. Fenton and Neil [5] have also used Bayesian approach to predict the number of defects. However, since their technique make use of data that is not available to us, we restrict the comparison between $BayesED^3M$ and their approach to a qualitative level.

ED^3M is a Maximum Likelihood Estimator (MLE) also based on Defect Decay Model of system testing process proposed in [2]. The main advantage of ED^3M is that it converges fast to the actual R_0 using information only from the current project (the project for which we are interested in finding R_0 or the actual number of defects). Whereas other estimators, discussed in [10, 11], use more information such as historical knowledge from past projects. It is shown in [6, 7] that ED^3M performs better or as good as these estimators. In many cases where no prior knowledge is available, and so the other estimators cannot be applied, ED^3M produces effective results. As with any other approach ED^3M also has limitation, as discussed in [6, 7]. The limitation of ED^3M is that when the system testing process is poor it causes high corruption of the data (the addition of very high noise to data) which is collected from the ongoing project. Therefore such data severely degrades the performance of ED^3M . To overcome this limitation other sources of information are necessary. Assuming good prior knowledge is available, we need an efficient estimator to use this knowledge in conjunction with the data available from the current project to estimate the actual number of defects R_0 in the software. This goal is achieved using the $BayesED^3M$ estimator described in Section 2.

We now quantitatively compare $BayesED^3M$ and ED^3M using data acquired from an industrial project. We have successfully run $BayesED^3M$ on other data sets from other industrial projects of varying sizes, but we will use one to show a quantitative comparison due to brevity of space. The results of applying both ED^3M (drawn with ‘-x-’ line) and $BayesED^3M$ are plotted in Figure 4. We have drawn a thick constant solid line of actual R_0 for reference. Note that the graph is normalized along both axes for the protection of the privacy of data. There are two sets of $BayesED^3M$ curves on two ends of reasonably accurate prior knowledge ($\pm 10\%$ of actual R_0) range. Each set has three $BayesED^3M$ curves of varying prior variances $\sigma_{R_0}^2$ for fixed prior mean μ_{R_0} to show how prior knowledge affects the performance of $BayesED^3M$ when compared

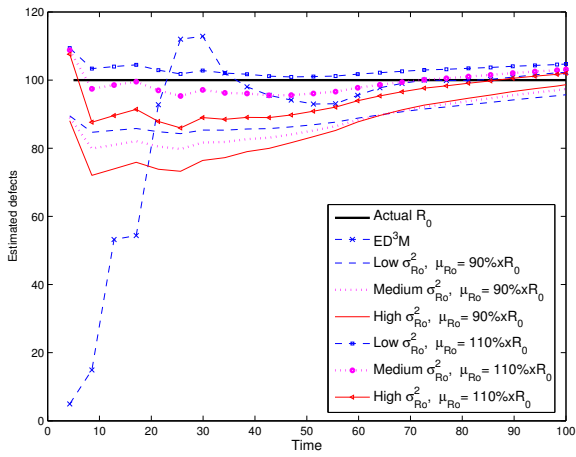


Figure 4. Comparison of $BayesED^3M$ and ED^3M . $BayesED^3M$ is evaluated with $\mu_{R_0} = 100 \pm 10\%R_0$ and also with low, medium, and high values for $\sigma_{R_0}^2$.

to ED^3M . The three variances low, medium and high are all greater than that of the variance of the data σ_N^2 and they reflect different levels of confidence in prior mean μ_{R_0} from high to low respectively. The reason for keeping $\sigma_{R_0}^2 > \sigma_N^2$ is that when the prior variance $\sigma_{R_0}^2$ is lower $BayesED^3M$ will produce a constant line of prior mean over shadowing the effect of current data as discussed in Section 2. Hence this case does not contribute much to the behavioral analysis and comparison of $BayesED^3M$. Note that the value of σ_n^2 for initial data points will be greater than $\sigma_{R_0}^2$. The value of σ_n^2 will decrease as the data set will grow and it will become smaller than $\sigma_{R_0}^2$.

Let us first discuss the set with prior mean $\mu_{R_0} = 110\%R_0$ as seen in Figure 4. In this case low prior variance $\sigma_{R_0}^2$ or high confidence curve given by ‘ $-\square-$ ’ and medium variance or medium confidence curve given by ‘ \dots ’ performs very well compared to ED^3M . Which is again a valid expected behavior when we have reasonably correct information and we are confident on it. In this case $BayesED^3M$ should take this prior knowledge more into account than the data from the current project and produce estimate accordingly. Even the weak confidence curve given by ‘ $-\triangleleft-$ ’ for the same prior mean μ_{R_0} is almost as good as ED^3M . Strong and medium confidence curves (low and medium $\sigma_{R_0}^2$ respectively) in the second set of curves with prior mean $\mu_{R_0} = 90\%R_0$ gives acceptable estimates. The high prior variance $\sigma_{R_0}^2$ or weak confidence curve (‘ $-\square-$ ’) does not perform as well as ED^3M but definitely performs better than ED^3M without correction (as shown by curve ‘ $-\dots-$ ’ in Figure 5) by providing better initial estimates. Figure 5 shows plots of ED^3M for the

same data set which is used in Figure 4. The curve (‘ $-\dots-$ ’) in Figure 5 is ED^3M without correction. The curve (‘ $-\triangleleft-$ ’) depicts the expected exponential behavior but it is not converging fast enough. In order to compensate for the convergence rate we developed a correction technique defined in [6, 7]. The other curve (‘ $-\square-$ ’) in Figure 5 is ED^3M with correction. Whenever we mention ED^3M we mean ED^3M with correction, otherwise we will state explicitly.

Now we will briefly compare $BayesED^3M$ with the other techniques [10, 11]. Note that since these techniques have already been compared in detail with ED^3M in [6, 7], we will draw conclusions based on transitive comparison. Two of these techniques, Padberg’s approach [10] and Software Reliability Growth Model (SRGM) based on Gompertz curve [11], have been quantitatively compared with ED^3M using the data set available in the literature [10]. ED^3M has demonstrated better than Padberg’s approach [10]. More over, Padberg’s approach is dependent on a number of parameters, as discussed in [6, 7], which are not readily available in the testing process. Hence they limit the applicability of the approach. Note that instead of directly incorporating such parameters which give specific attributes or pieces of knowledge of data from projects (regardless of current or past projects) into the model, $BayesED^3M$ simply relies on the number of defects found in the current project and $\sigma_{R_0}^2$ and μ_{R_0} from past projects as shown in Figure 1. We provide a clean interface between $BayesED^3M$ and the data from past projects in the form of a Method which will extract $\sigma_{R_0}^2$ and μ_{R_0} . Similar issue exists with Gompertz curve model [11]. The initial values of the three parameters R_0 , b and k which characterizes Gompertz curve are not easily estimated.

Fenton and Neil [5] have critiqued software defect prediction models based on different techniques. They have proposed an alternate approach using Bayesian Belief Network (BBN). We would like to address some of the issues that Fenton and Neil have raised about techniques of defect prediction in [5]. They have made an argument based on the work of Adams [1] that only 20% of total defects cause 80% of faults or failures (Pareto Principle). They argue that predicting the count of defects in a product may not contribute to improved reliability. However, reliability is just one of many dimensions of software quality that also includes number of defects. Therefore, predicting the number of defects helps to estimate the quality of the product released and also to infer effort needed for customer support. In addition to that, remaining number of defects is a good measurement of the status of the testing process and helps a testing manager in making decisions related to resource allocation and scheduling issues. Another benefit is that when we do have reliable historical knowledge μ_{R_0} and if the estimate of the total number of defects differs substantially than μ_{R_0} , then it strongly raises suspicion towards the quality of

testing process and the quality (which implies reliability) of the product. Another issue that Fenton and Neil [5] pointed out is that a very accurate residual defect density prediction may be a very poor predictor of operational reliability. We know that defect data reflects the operational profile used for system testing and when the operational profile is incorrect defect data is inaccurate as well. We have discussed this issue in [6, 7] for ED^3M , we suggested that with the use of historical knowledge this problem can be alleviated. So work on $BayesED^3M$ addresses this issue.

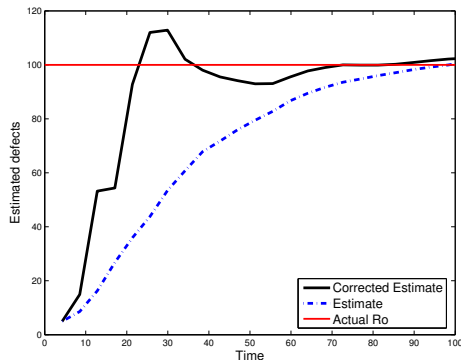


Figure 5. ED^3M with and without correction.

5. Conclusion and Future Work

Results in [6, 7] have shown that ED^3M , a Maximum Likelihood Estimator, in general outperforms other approaches. The same study has shown that high noise in data in the beginning of the testing process greatly impact the outcomes of ED^3M . The higher the noise level, the larger the latency to converge to a reasonable estimation. In order to overcome this problem a Bayesian version ($BayesED^3M$) of the estimator was described here. $BayesED^3M$ uses data from previous similar projects in the form of μ_{R_0} and $\sigma_{R_0}^2$ to compute the estimations. The value of $\sigma_{R_0}^2$ when compared to the variance of the current data dictates the influence of prior knowledge on the estimation.

It is clear from the case studies that $BayesED^3M$ performs better than other techniques whenever reliable and accurate data is available. However, conflicting information such as low variance (high confidence) for inaccurate μ_{R_0} can result in misleading estimations. Also, initial overestimation appear to produce better results than initial underestimation.

Currently we are investigating the use of Clustering techniques to select similar past project. We are studying different attributes of knowledge in projects which can be used to classify projects. These attributes of Projects form a class of similar projects which can be used to extract $\sigma_{R_0}^2$ and μ_{R_0} .

We are also studying relationship of attributes of knowledge to the CMMI levels. For example if an organization is on a certain level then which metrics can be collected from this organization. Our conjecture is that the value of $\sigma_{R_0}^2$ can be associated with the level of capability/maturity in terms of confidence level. In simple words the higher the ranking of the company in CMMI levels, the higher is the confidence (smaller value of $\sigma_{R_0}^2$) in μ_{R_0} .

References

- [1] E. Adams. Optimizing preventive service of software products. *IBM Research Journal*, 28(1):2–14, 1984.
- [2] J. W. Cangussu, R. A. DeCarlo, and A. P. Mathur. A formal model for the software test process. *IEEE Transactions on Software Engineering*, 28(8):782–796, August 2002.
- [3] S. R. Dalal and C. L. Mallows. Some graphical aids for deciding when to stop testing software. *IEEE Journal on Selected Areas in Communications*, 8(2):169–175, February 1990.
- [4] W. K. Ehrlich, J. P. Stampfel, and J. R. Wu. Application of software reliability modeling to product quality and test process. In *12th International Conference on Software Engineering (ICSE'90)*, pages 108–116, Nice, France, March 1990. IEEE.
- [5] N. E. Fenton and M. Neil. A critique of software defect prediction models. *IEEE Transactions on Software Engineering*, 25(5), September-October 1999.
- [6] S. W. Haider and J. W. Cangussu. Estimating defects based on defect decay model: ed^3m . *Submitted to IEEE Transactions on Software Engineering*.
- [7] S. W. Haider and J. W. Cangussu. A novel approach for defect estimation. Technical Report UTDCS-30-05, The University of Texas at Dallas, August 2005.
- [8] B. Marick. *The Craft of Software Testing*. Prentice Hall, Englewood Cliffs, New Jersey, 1995.
- [9] J. Musa. *Software Reliability Engineering*. McGraw-Hill, 1999.
- [10] 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.
- [11] D. Satoh and S. Yamada. Discrete equations and software reliability growth models. In *12th International Symposium on Software Reliability Engineering (IS-SRE)*, pages 176–184. IEEE, November 2001.