Decision Support System for Fault Localization in COTS Integration

D Mahendra Reddy
M.Tech(10695D5833)
Department of Computer Science & Engineering
Madanapalle Institute of Technology & Science
Andhra Pradesh
mahendrareddy39@gmail.com

Surya Bahadur
Assistant Professor
Department of Computer Science & Engineering
Madanapalle Institute of Technology & Science
Andhra Pradesh
surya.bahadur@gmail.com

Abstract— **The failure-detection and fault-correction are analytical procedurees in attaining high-quality achievement of software excellence. In this paper, we adduce a amount of improvements on the bourgeois software bendability amplification models (SCEM) to call absolute software advance action by eliminating some abstract assumptions. A lot of of these models accept focused on the collapse apprehension action and not accustomed the aforementioned antecedence to clay the accountability alteration procedure. But, a lot of abeyant software errors may abide uncorrected for an continued time even afterwards they are detected, which increases their impact. The outstanding software faults are again one of the majority capricious affidavit for software superiority. Therefore, we aggrandize a accepted anatomy of the clay of the abortion apprehension and accountability alteration procedures referred as continued software bendability amplification archetypal (ESCEM) for affiliation accountability prediction. Furthermore, we aswell assay the aftereffect of applying the delay-time non-homogeneous poisson action (NHPP) models. Lastly, after examples are apparent to authenticate the after-effects of the affiliation of the analysis and alteration procedure**.

Keywords: fault prediction, defect forecasting, SCEM, ESCEM, NHPP

I. INTRODUCTION

Software bendability is able to be beheld as a affecting admeasurement of quantifying software failures and is audible as the anticipation of failure-free software action for a accurate aeon of time in a accurate ambience [1]. Therefore, in align to accomplish a adopted akin of superiority; the bendability of a software adjustment accept to be high. The fault-detection and fault-correction are analytical procedures in attaining high-quality software quality. During the software apprehension procedure, testing cases are run and ultimately failures are detected. After discovery, the debugging aggregation be declared to appraise the malfunction, position the accountability and fix the accountability [2-4]. That is, the accountability alteration action affects the abidingness of a software artefact decidedly and we should pay added absorption to it. newly, abounding SCEMs accept been developed to appraisal some advantageous measures such as the MVF, character of outstanding faults, and breakdown apprehension rate. Most of these models accept focused on the breakdown apprehension procedure. Consideration of accountability alteration action in the achievable models is limited. However, to attain adapted acme of software superiority, it is absolute cogent to be accordant affecting technologies for removing the errors in the accountability alteration procedure.In reality, the accountability alteration amount is a purpose of the complication of affairs modules, the manpower, the accomplishment of testing teams, the borderline for the absolution of the software, etc. Abstracts accept been performed based on absolute abstracts set, and the after-effects appearance that the proposed models access a added aftereffect in ciphering the character of primary faults and aswell announce a goodness-of-fit in agreement of the mean-of-squares absurdity criterion.This certificate is organized as follows. In Section 2, the backdrop of the affiliated models are reviewed. An affiliation archetypal of breakdown apprehension and accountability alteration procedures is proposed in Section 3. In Section 4, we appearance how some achievable NHPP models are re-evaluated from the point of appearance of adjourned alteration action and actualize some observations amid the belated time NHPP models and the chip model. The abstracts and after-effects are presented in Section 5. after the abstracts are fabricated in Section 6.

II. SOME SCEMS BASED ON NHPPS

Let *{N(t), t* > 0} denote a counting procedure representing the increasing number of errors detected by instance t, m(t) be the MVF of the predictable number of faults detected in instance (0, t), and λ *(t*) indicate the failure strength at testing time t. That is, they satisfy the following:

$$m(t) = E(\{N(t), t > 0\})$$

and,

$$\lambda(t) = \frac{dm(t)}{dt}.$$

Thus, an SCEM based on NHPP with MVF *m(t)* be able to be formulated as

$$P\{N(t) = n\} = \frac{m(t)^n}{n!} e^{-m(t)}, n = 0,1,2,....$$

Most accessible SCEMs based on NHPP have the following essential assumptions regarding the software error-detection procedure [1, 8]:

Assumptions

1. The error-detection procedure follows the NHPP.
2. The software organization is subject to encountering the outstanding faults in the organization at random times.
3. All faults are self-governing and similarly detectable.
4. The indicate number of faults detected in the instance interval (t, t+At] is comparative to the indicate number of faults outstanding in the organization.
5. The proportionality may or may not modify with time.
6. Each time a failure occurs, the burden that caused it is absolutely detached with no new faults being introduced.
7. The detected faults are instantly detached with certainty and alteration of faults takes just insignificant instance.

According to these assumptions, we find:

$m(t + \Delta t) - m(t) = d(t) \times [a - m(t)] \Delta t$

where *a* is the predictable number of errors to be eventually detected (i.e. $m(\infty) = a$) and d(t) is the error discovery rate for each error at testing time *t*. Solving the on top of equation, we include

$$\lambda(t) = d(t) \times (a - m(t)),$$

$$m(t) = a + (m(0) - a) \exp(-\int_0^t (u)du). \quad (6)$$

In common, we be able to have dissimilar SCEMs based on NHPP using dissimilar MVF.

A. Goel-Okumoto mode[5]

The majority well-known SCEM based on NHPP is the model proposed by Goel and Okumoto. This model assumes that the error detection speed per error in the testing phase is steady. Therefore, it is identical to get d(t) = *b* in Eq. (4) and the MVF is derivative by

$$m(t) = a(1 - e^{-bt}), a>0, b>0,$$

where *a* is the predictable number of errors to be ultimately detected and *b* represents the error discovery rate per error.

B. Yamada s-shaped curve model [6]

Yamada et al. assume that the error discovery rate is a time-dependent reason. That is,

$$d(t) = \frac{b^2 t}{1 + bt},$$

and, $m(t) = a[1-(1+bt) e^{-bt}]$, a>0, b>0

C. A general discrete NHPP model [7,-11]

The two parameters, *a* and *b* engage in recreation the same function as the *a* and d(t) in Eq. (4). Taking *w* = 1-b, we have

$$m(i+1) = wm(i) + (1-w)a \quad (7)$$

This indicates that m(i+1) is identical to the weighted arithmetic indicate of m(i) and *a* with weights *w* and 1 - *w*. More generally, the weighted arithmetic indicates in Eq. (7) be able to be replaced by the weighted geometric, harmonic or quasi-arithmetic earnings to derive other obtainable NHPP models [8]. Thus, we contain the following theorem:

Theorem 1: Let *g* be a real-valued and strictly monotone purpose and *m*(*i*+1) be the same to the quasi-arithmetic signify of m(i) and *a* with weights w(i+1) and 1-w(i+1), after that

$$M(i) = g^{-1}\{u_i \, g(m(0)) + (1-u_i) g(a)\},$$

where 0<w(i)<1, a>0, $u_i = \prod_{j=1}^{i} w(j)\ for i \geq 1$ and $u_0 = 1$.

D. A general continuous NHPP model [7-11]

Comparable to the on top of discussion in the discrete case, we contain the next theorem aimed on a general continuous NHPP model.

Theorem 2: Let *m(t + At)* be the similar to the quasi-arithmetic indicate of m(t) and *a* through weights w(t + At) and 1- w(t + At), then we contain

$$m(t) = g^{-1}\{g(a) + [g(m(o)) - g(a)]e^{-B(t)}\},$$

where *g* is a real-valued, strictly monotonic, and differentiable reason, *a* is the expected numeral of primary faults,

$$\lim_{\Delta t \to 0} \frac{1 - w(t + \Delta t)}{\Delta t} = b(t) \text{ and } B(t) = \int_0^t b(u)du .$$

E. A delayed-time NHPP model [7-11]

We know that the time to eliminate a fault depends on the difficulty of the detected errors, the skills of the debugging team, the accessible manpower, and the software expansion environment [1, 12, 13]. Therefore, the time spent by the correction process is not insignificant. Schneide wind [2, 14] first modeled the fault-correction procedure by using a delayed error-detection procedure and unspecified that the error-detection procedure follows the NHPP and the speed of change of the MVF is exponentially lessening. Furthermore, the fault-detection procedure in the Schneidewind model is isomorphic to the G-O representation, not including that the G-O representation is viewed as a continuous-time procedure. Xie [2] comprehensive the Schneidewind model to a continuous description by substituting a time-dependent delay function for the steady delay in the Schneidewind model. Thus, we eliminate the unreasonable assumption that the fault-correction procedure is ideal and can thus establish a matching time-dependent delay purpose to fit the fault-correction procedure in our history research [7]. That is, the latest MVF is

$$m(t) = a(1 - e^{-bt}e^{b\varphi(t)}),\ a > 0,\ b > 0,$$

where *a* and *b* are the parameters in G-O representation, *(p(t)* is a delay-effect factor to represent the corresponding time-dependent insulate in the improvement procedure.

III. AN INTEGRATED MODEL

In the history, much investigate on software dependability models contain concentrated on modeling and predicting failure incidence and contain not given equal main concern to modeling the burden correction procedure [15]. However, most latent software errors may remain uncorrected for a long time even after they are detected, which increases their impact. The outstanding software faults are frequently one of the majority unreliable reasons for software superiority. Therefore, we expand a general framework of the modeling of the breakdown detection and burden correction procedures.

Assumptions

1. The error-detection procedure follows the NHPP.
2. The software system is subject to encountering the outstanding faults in the organization at random period.
3. All faults are independent and similarly detectable.
4. The indicate number of faults detected in the point in time interval (t, t+ Δ t) is comparative to the mean numeral of faults outstanding in the system. The proportionality, Δ (t), may generally be a time-dependent reason [2].
5. The denote number of faults corrected in the instance interval (t, *t*+ Δ *t*) is proportional to the mean number of detected but not yet corrected faults outstanding in the system. The proportionality, $\mu(t)$, may as well be time-dependent [2].
6. Every time a failure occurs, the fault is completely detached with no new faults being introduced.

A. Description of the modeling

Based on the exceeding assumptions 1-6, we have the subsequent differential equations for the MVF *m*(*t*) and *mc*(*t*) of breakdown detection and fault correction procedures:

$$\frac{dm(t)}{dt} = \lambda(t)(a - m(t)), a > 0$$

$$\frac{dm_c(t)}{dt} = \mu(t)[m(t) - m_c(t)].$$

To develop a structure of the modeling of these procedures, we thus get the next theorem.

Theorem 3: If $D(t) = \int_0^t \lambda(s)ds$, $C(t) = \int_0^t \mu(s)ds$, and the degree of difference equations for the MVF m(t)

and $m_c(t)$ of breakdown detection and fault correction procedures is as follows:

$$\frac{dm(t)}{dt} = \lambda(t)(a - m(t)), a > 0$$

$$\frac{dm_c(t)}{dt} = \mu(t)[m(t) - m_c(t)].$$

We have

$$m(t) = a[1-\exp(-D(t))],$$

$$m_c(t) - e^{C(t)}\{\int_0^t ac(s)e^{C(s)}(1-e^{-D(s)})ds\}$$

where a is the expected numeral of initial faults, and the first condition $m(0)=mc(0)=0$, i.e. no failure at the commencement.

Proof: Solving the above differentiable eq.(9) by multiplying together sides with $e^{D(t)}$, we get

$$\frac{d}{dt}(e^{D(t)}m(t)) = a\frac{d}{dt}(e^{D(t)})$$

Thus, m(t) = $e^{C(t)}[ae^{D(t)} - a] = a(1-e^{-D(t)})\ e^{-D(t)}[ae^{D(t)} - a] = a(1-e^{-D(t)})$.

As for Eq.(10), we can multiply together sides with $e^{C(t)}$, we have

$$\frac{d}{dt}(e^{C(s)}m_c(t)) = a\int_0^t c(s)e^{C(s)}(1-e^{-D(s)})ds.$$

Finally, we have the effect,

$$m_c(t) = e^{-C(s)}\{\int_0^t ac(s)e^{C(s)}(1-e^{-D(s)})ds.$$

B. Constant rate in these two procedures

Note that λ (t) in Eq.(9) is commonly a time-dependent function and can be redraft as

$$\lambda(t) = \frac{m'(t)}{a - m(t)}, a > 0.$$

$$e^{-bt}$$

$$\mu(t) = \frac{m_c'(t)}{m(t) - m_c(t)}.$$

$$\frac{dm_c(t)}{dt} = \mu[m(t) = a[1-\exp(-\lambda t)], m_c(t)].$$

$$m_c(t) = a[1-(1+bt)e^{-bt}],$$

From the higher than equation, λ (t) can be interpreted as the failure discovery rate per outstanding fault. In particular, solving the degree of difference equation (9) with λ (t) = b under the initial situation $m(0)=0$ yields the following MVF:

m(t)=a(1- e^{-bt}),a>0, b>0.

Based on the above equation, the case is G-O model. Furthermore, μ (t) in Eq.(10) has a similar interpretation. That is,

$$\mu(t) = \frac{m_c'(t)}{m(t) - m_c(t)}.$$

By the above deduction, μ (t) is just the fault correction rate per detected but not corrected fault. Particularly, if λ (t) and μ (t) equal b, the corresponding MVF is derived as follows:

Corollary 1: If the differential equations for the MVF $m(t)$ and $mc(t)$ of failure detection and fault correction procedures is as follows:

$$\frac{dm(t)}{dt} = b(a - m(t)), \quad a>0,\ b>0$$
$$\frac{dm_c(t)}{dt} = b[m(t) - m_c(t)].$$

Therefore, we have

$$m\ (t)\ =\ a\,[1\ -\ e^{-\,b\,t}\,],$$
$$m_c(t) = a[1-(1+bt)e^{-bt}]$$

using the initial condition $m(0)=mc(0)=0$, i.e. no failure at the beginning.

Corollary 2: If the differential equations for the MVF $m(t)$ and $m_c(t)$ of failure detection and burden correction procedures is as follows:

$$\frac{dm(t)}{dt} = \lambda(a - m(t)), \quad a>0,\ b>0$$
$$\frac{dm_c(t)}{dt} = \mu[m(t) - m_c(t)].$$

We have,

$$m(t) = a[1-\exp(-\lambda t)],$$
$$m_c(t) = a\left[1+\frac{\mu}{\lambda-\mu}e^{-\lambda t} - \frac{\lambda}{\lambda-\mu}e^{-\mu t}\right]$$

where a is the expected number of initial faults, and the initial condition m(0) and $m_c(0)$ equal zero.

IV. Comparisons and Observations

Numerous stochastic models for software failure phenomenon have been developed to measure software consistency, and many of them are based on NHPP. In fact, these models are very useful to describe the software failure detection and correction procedures with suitable failure occurrence rates. In this following, we discuss how several existing SCEMs based on NHPP models can be comprehensively derived by applying some different factors. Specifically, we focus on the Yamada S-shaped curve model [6].

A. Error detection rate approach

From Section 2 we know that we can have dissimilar SCEMs by using different d(t) in Eq. (5). For example, given $d(t) = \frac{b^2 t}{1+bt}$ and m(0) = 0 in Eq. (5), we can get its matching MVF m(t) by the integration of d(t) as exposed below:

$m(t) = a(1-(1+bt)e^{-bt}),\ a>0, b>0$.

That is, a difference of the G-O representation, known as the Yamada S-shape curve model [16], can be derivative.

B Delay-time approach

Moreover, we can have dissimilar delay-time NHPP models by applying different delay-effect factors. Therefore, if $\varphi(t) = \frac{1}{b}\ln(1+bt)$, we be able to also get its corresponding MVF by Eq. (8) as below:

$$m(t) = a(1-e^{-bt}e^{b\varphi(t)}) = a(1-(1+bt)e^{-bt}), a>0, b>0$$

This example reflects the reality that the S-shape representation can be interpreted from different points of observation. In other expressions, by specifying the error-detection speed per error or the delay-effect factor, we can formulate various models with a new MVF.

C An integrated approach

Suppose together the failure discovery rate per outstanding fault, $\lambda(t)$,and the fault correction rate per detected but not corrected fault, $\mu(t)$, have the similar value. That is, $\lambda(t) = \mu(t) = b$. Therefore, we have the corresponding MVF as below: $m(t) = a(1-(1+bt)e^{-bt}), a>0, b>0$

D Comparisons for these three approaches

From the on top of derivation, we know that Yamada S-shaped curve be able to be interpreted from various points of view. In further words, by specifying the error detection rate per error, i.e. $d(t) = \frac{b^2 t}{1+bt}$ in Eq. (5). furthermore, from the point of view of effect factor, i.e.

$\varphi(t) = \frac{1}{b}\ln(1+bt)$ in Eq. (8). This factor is able to reflect the moment lag in the correction procedure. Furthermore, if λ (t) = μ (t) = b in the integrated move toward of Section 3, the S-shaped curve be able to be described. Thus, we create the following observations:

- The traditional NHPP MVF, is identical to the common delayed-time appearance of the MVF.
- A lot of existing NHPP models for software consistency can be derived as particular cases of this integrated structure of detection and improvement procedures.

V. EXPERIMENTAL EXAMPLES

A. Estimation and criteria for similarity not including loss of simplification, we discuss three kinds of approaches described in Section 3 to representation the fault modification procedure. The first move toward is about the delayed-time NHPP representation, where $\varphi(t)$ is the Rayleigh reason, i.e. $\varphi(t) = cte^{-(\frac{t^2}{\theta})}$. The following case discusses the common case of the proposed move toward in Section 3 where the breakdown detection rate and burden correction speed are different constants, i.e. $\lambda(t) = \lambda$, $\mu(t) = \mu$. Furthermore, the third case is the integrated representation with equivalent value of rate, i.e. $\lambda(t) = \mu(t) = b$.

When the three case are functional to the equation (8)-(10) and it is solved through respect to MVF $m(t)$ beneath the primary condition $m(0)=0$, we obtain the following equation, correspondingly:

$$m_1(t) = a(1 - e^{-bt}\exp(bcte^{-(\frac{t^2}{\theta})})) \qquad (14)$$

$$m_2(t) = a\left[1 + \frac{\mu}{\lambda-\mu}\exp(-\lambda t) - \frac{\lambda}{\lambda-\mu}\exp(-\mu t)\right] \qquad (15)$$

$$m_3(t) = a(1-(1+bt)e^{-bt}) \qquad (16)$$

the majority popular estimation techniques are the highest probability estimation (MLE) and the least squares estimation (LSE) [1, 12, 13]. The highest likelihood technique estimates parameters by solving a set of simultaneous equations. But the corresponding equations are frequently complex and have to be solved numerically. Furthermore, we use the process of least squares to reduce the sum of squares of the deviations between what we in fact observe/get and what we expect. On the additional hand, we adopt one estimate criterion in the comparison of goodness-of-fit of the models.

B. Performance analysis

The data set was the modules that open source and integration enabled. The modules such as lucene, jfreecharts was used for a real-time, authority and control system. The numeral of object instructions was regarding 14,600 and it took 40 man hours to do the software examination. The data set includes 136 experimental failures, recorded in the implementation time.

Primary, parameters of models are evaluated and the corresponding MVFs are obtained. Second, all the selected models are compared with each additional based on the objective criteria. Table 1 shows the predictable parameters of the proposed models described in Eq. (14)-Eq. (16) solved numerically by MLE and LSE. The minor the MSE a representation is, the better they fit the experimental data. Figure 1 depicts the experimental curves and the built-in curves of the cumulative statistics of failures using the G-O, $m_1(t)$, $m_2(t)$, and $m_3(t)$ models, respectively. Figure 2 illustrates the dissimilarity between the strength purposes of MVF predicted by these models. Examples of the estimated MVF, mi(t), $m_2(t)$, and $m_3(t)$, and their 90% confidence limits are exposed in Figure 3-Figure 5. We can see from Table 1, Figure 3 and Figure 4 that the delayed-time representation and the integrated representation fit the data well. Particularly, the delayed-time representation has the smallest value of MSE (21.64); consequently, we can conclude that the delay-effect factor representation gives a better fit in this research. Furthermore, Figure 2 shows the estimated instantaneous fault correction speed of these models, in which we discover that the integrated move toward is a bell-shaped-type curve. In authenticity, the burden correction speed is a purpose of the difficulty of program modules, the manpower, the talent of testing teams, the time limit for the release of the software, etc. At the beginning of the software correction procedure, the programmers usually eliminate easy-to-correct errors in the programs. That is, the correction speed is increasing in such testing segment. As instance goes, the team becomes acquainted with the software-testing surroundings, with better skills, techniques and tools. These improvements may velocity up the testing activities [1, 12, 13]. As instance passes additional, it is relatively more not easy for the correcting team to correct more errors. That is, the speed that resulted from the correction procedure becomes smaller. This phenomenon just fits the increasing-then-decreasing situation of the person learning procedure.

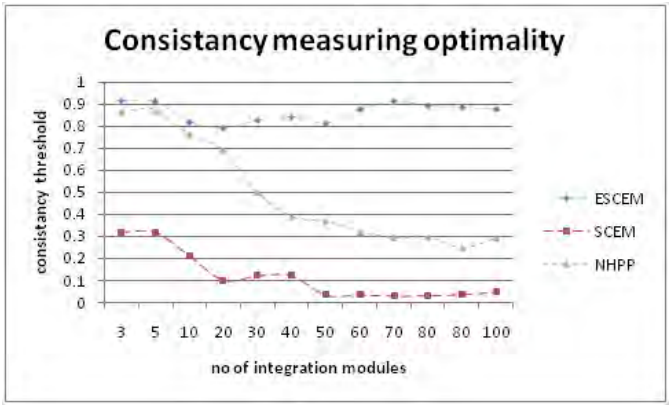

Fig 1: Consistency advantage of over SCEM and NHPP

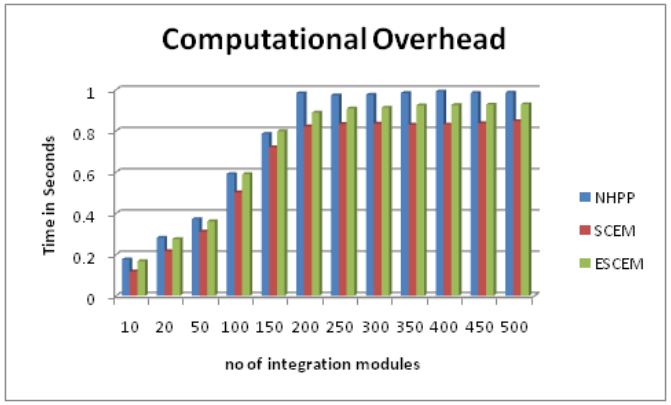

Fig 2: Comparison of computational overhead between ESCEM, SCEM and NHPP

VI. CONCLUSIONS

In this document, we accommodate complete ESCEM based on NHPPs, which add in the abortion apprehension and absurdity alteration procedures, and accommodate discussed the methods of quantitative abidingness appraisal based on this latest model. We aswell actualize some account amid the delay-time NHPP representation and the chip representation. More than a few after cases based on 18-carat ascendancy arrangement accommodate been presented and the after-effects authenticate the delayed-time representation and the chip reproduction fit the advice well.

REFERENCES

[1] M. R. Lyu. Handbook of Software Consistency Engineering, McGraw-Hill, 1996.
[2] M. Xie and M. Zhao, "The Schneidewind software consistency model revisited," Proceedings of the 3th International Symposium on Software Consistency Engineering , pp. 184-192, May 1992.
[3] S. S. Gokhale, "Software Consistency analysis incorporating fault detection and debugging activities," Proceedings of the 9th International Symposium on Software Consistency Engineering , pp. 64-75, November 1992.
[4] M. Ohba, "Software Consistency Analysis Models," IBM J. Res. Develop, Vol. 28, No. 4, pp. 428-443, July 1984.
[5] L. Goel and K. Okumoto, "Time-dependent Error-detection Rate Model for Software Consistency and Other Performance Measures," IEEE Trans. on Consistency, Vol. R-28, pp. 206-211, August 1979.
[6] S. Yamada, M. Ohba, and S. Osaki, "S-shaped Consistency Growth Modeling for Software error detection," IEEE Trans. on Consistency, vol. R-32, No. 5, pp. 475-478, December 1983.
[7] J. H. Lo and S. Y. Kuo, and C. Y. Huang, "Consistency Modeling Incorporating Error Procedurees for Internet-Distributed Software, " IEEE Region 10 International Conference on Electrical and Electronic Technology (TENCON 2001), pp. 1-7, Aug. 2001, Cruise Ship, Super Star Virgo.
[8] R. H. Huo, S. Y. Kuo, and Y. P. Chang, "On a Unified Theory of Some No homogeneous Poisson Procedure Models for Software Consistency," Proceedings of International Conference on Software Engineering: Education & Practice, pp. 60-67, January 1998.
[9] J. H. Lo, "A Unified Scheme for Modeling Software Consistency with Various Failure Detection and Fault Correction Rates," International Computer Symposium (ICS 2004), pp. 914-919, Dec. 2004, Taipei, Taiwan.
[10] J. H. Lo, "Considering Both Failure Detection and Fault Correction Activities in Software Consistency Modeling," Proceedings of the IEEE Region 10 Annual International Conference (TENCON 2006), pp. 1-4, Nov. 2006, Hong Kong, China.
[11] J. H. Lo "Effect of the Delay Time in Fixing a Fault on Software Error Models," Proceedings of the 31rd IEEE Annual International Computer Software and Applications Conference (COMPSAC 2007), Vol. 2, pp. 711-716, July 23-27, 2007, Beijing, China.
[12] J. D. Musa, A. Iannino, and K. Okumoto. Software Consistency, Measurement, Prediction and Application, McGraw-Hill,1987.

[13] J. D. Musa. Software Consistency Engineering: More Reliable Software, Faster Development and Testing, McGraw-Hill, 1998.
[14] N. F. Schneidewind, "Analysis of Error Procedurees in Computer Software," Sigplan Notices, Vol. 10, pp. 337-346, June 1975.
[15] N. F. Schneidewind, "Fault Correction Profiles," Proceedings of International Symposium on Software Consistency Engineering, pp. 60-67, 2003.
[16] J. D. Musa, Software Consistency Data, Report and Data Base Available from Data and Analysis Center for Software, Rome Air Development Center (RADC). Rome, NY.292-296, May 1985.