International Journal of Performability Engineering Vol. 11, No. 4, July 2015, pp. 369-379.

Printed in India

# M-SRAT: Metrics-based Software Reliability Assessment Tool

KAZUYA SHIBATA[1], KOICHIRO RINSAKA[2], and TADASHI DOHI*[1]

[1]*Department of Information Engineering, Hiroshima University*

*1-4-1 Kagamiyama, Higashi-Hiroshima 739–8527, JAPAN*

[2]*Faculty of Business Administration, Kobe Gakuin University*

*1-1-3 Minatojima, Chuo-ku, Kobe 658-8586, JAPAN*

*(Received on August 31, 2014, revised on January 09, 2015)*

**Abstract:** In this paper we develop a software reliability assessment tool, called M-SRAT: Metrics-based Software Reliability Assessment Tool, by using several testing metrics data as well as software fault data observed in the testing phase. The fundamental idea is to use the metrics-based software reliability models proposed by the same authors. M-SRAT is written in Java language with 54 classes and 8.0 KLOC, where JDK1.5.0 9 and JFreeChart are used as the development kit and the chart library, respectively. This tool can support (i) the parameter estimation of software reliability models via the method of maximum likelihood, (ii) the goodness-of-fit test under several optimization criteria, (iii) the assessment of quantitative software reliability and prediction performance.

## 1. Introduction

During the last three decades, the stochastic models, called *software reliability models* (SRMs) that analyze and explain software fault-detection phenomena, have been extensively developed in the literature [7],[8]. In fact, till now, over 200 SRMs have been proposed from various mathematical points of view. The classical and the most important SRMs may be non-homogeneous Poisson process (NHPP) based SRMs that have gained the popularity for describing the stochastic behavior of the number of software faults detected in the testing phase. On the other hand, it has been pointed out that these SRMs might fail to incorporate the software development metrics apply regression models, where the probability distribution denoting the time to software failure can be represented by the environmental factors characterizing probabilistic

*Corresponding author's email: dohi@rel.hiroshima-u.ac.jp

(multi-variate) events as *covariates*. However, the above work concerns only the static models on time, and cannot deal with the dynamic behavior on software debugging process, called *reliability growth phenomenon*, in their framework. Evanco [2] and Ikemoto and Dohi [3] use exponential regression models to represent the Poisson intensity function and derive different multifactor SRMs from the Cox proportional hazards regression based approach. Ray *et al*. [14], Rinsaka *et al*. [15], Shibata *et al*. [18] develop novel approaches to handle both software fault count data and software metrics data in the NHPP based modeling framework, by means of the proportional intensity model (PIM). Their modeling approach is interesting, but does not unify the existing SRMs in the literature [7],[8]. Shibata *et al*. [17],[19] propose a consistent modeling framework for the multifactor NHPP-based SRMs by introducing the *cumulative Bernoulli trial process* (CBTP) and the discrete Cox proportional hazards regression. Okamura *et al.* [10],[11] also develop another multifactor NHPP based SRMs based on the logistic regression. Recently, both of the Cox proportional hazards regression-based SRM and the logistic regression-based SRM are extended in [6] and [5], respectively. Okamura and Dohi [13] discuss a multifactor NHPP-based SRM for component-based software and propose a Poisson regression-based SRM to assess the component reliability simultaneously. Ikemoto *et al*. [4] consider a generalized multifactor NHPP-based SRM, where the underlying testing environment is modulated by a discrete-time Markov chain.

In this way, considerable attentions have been paid to multifactor SRMs by utilizing the software metrics. The best way for progressive utilization of software reliability assessment methods in practice would be to provide a useful reliability assessment tool. Okamura and Dohi [12] develop a useful software reliability assessment tool on spreadsheet (SRATS) under the assumption that the fault detection time and/or the fault count data are available. The use the well-known 11 SRMs and implement the effective maximum likelihood estimation algorithms based on the EM (expectation and maximization) principle. To our best knowledge, there are only one software reliability assessment tool to handle both the software fault count data and the software metrics data. Shibata *et al.* [18] develop the so-called PISRAT to assess the software reliability with the PIM [15]. However, since it limits the number of available SRMs, it is appropriate to develop the software reliability assessment tool with the other modeling approaches. In

this paper we develop a software reliability assessment tool, called M-SRAT: Metrics-based Software Reliability Assessment Tool, by using several testing metrics data as well as software fault data observed in the testing phase. The fundamental idea is to use PIM, but some useful functions are taken into account, due to the more complexity of PIM than the NHPP-based SRMs. For instance, since PIM includes a larger number of parameters than the corresponding NHPP, and strongly depends on the testing metrics data, the optimization techniques to get the maximum likelihood estimates should be turned up. In order to implement M-SRAT as a web application, the graphical user interface (GUI) is also improved comparing with the existing software reliability assessment tools. M-SRAT is written in Java language with 54 classes and 8.0 KLOC, where JDK1.5.0 9 and JFreeChart are used as the development kit and the chart library, respectively. This tool can support (i) the parameter estimation of software reliability models via the method of maximum likelihood, (ii) the goodness-of-fit test under several optimization criteria, (iii) the assessment of quantitative software reliability and prediction performance. To our best knowledge, M-SRAT is the first freeware for dynamic software reliability modeling and measurement with time-dependent testing metrics. So, M-SRAT can be regarded as a computer-aided software reliability modeling and measurement tool from the user-perspective standpoint by combining the available testing metrics data with the common modeling and analysis techniques.

## 2. Software Reliability Modeling

### 2.1 Binomial Process Model

Let $M\ (>0)$ denote the initial number of faults contained in the software program. Suppose that each software fault is detected at independent and identically distributed *(i.i.d.)* discrete random time $T_j\ (j = 1, 2, \ldots)$, and the probability that one software fault is detected at the $i$-th time is given by $p_i \in (0, 1)$ which is mutually independent. Then, the conditional probability that the number of software faults detected at the $n$-th time, $Y_n$, is given by the binomial distribution [15]:

$$\Pr\{Y_n = y_n | Y_1 = y_1, \ldots, Y_{n-1} = y_{n-1}\} = B\left(y_n; M - \sum_{j=1}^{n-1} y_i, p_n\right), \quad (1)$$

where,
$$B(y;M,p)=\binom{M}{y}p^{y}(1-p)^{M-y} \tag{2}$$

denotes the binomial probability mass function (*p.m.f.*). In this paper, the probability $\mathbf{P}_n = (p_1,\ p_2,\dots,p_n)$ is called *the fault-detection probability*. Next we consider the non-conditional behavior of the number of software faults detected by $n$-th time. Suppose that $Y_1 = y_1$ faults are detected at the first time with probability $p_1$. Define the binomial $i$-fold convolution by $B^i(y;a,\mathbf{P}_i)=\sum_{k=0}^{y}B^{i-1}(k;a,\mathbf{P}_{i-1})B(y-k;M-k,p_i)$. Then the non-conditional probability that $y$ software faults are detected by $n$-th time is given by

$$\Pr\left\{\sum_{i=1}^{n}Y_i=y\right\}=B^n(y;a,\mathbf{P}_n)=\sum_{k=0}^{y}B^{(n-1)}(k;\ M,\mathbf{P}_{n-1})B(y-k;M-k,p_n)$$
$$=B\left(y;\ M,1-\prod_{i=1}^{n}\bar{p}_i\right), \tag{3}$$

which is due to the elementary binomial argument. In Eq.(3) $\bar{p}_i = 1 - p_i$. This is known as the *cumulative Bernoulli trial process* (CBTP).

**2.2 Discrete NHPP-based SRMs**

Since the initial fault contents $M$ cannot be known in general, it is appropriate to assume that $M$ is given by any discrete random variable. In Eqs.(1) and (3), if the prior distribution for $M$ is the Poisson distribution with mean ω ($>0$), then we have

$$\Pr\{Y_n=y_n|Y_1=y_1,\dots,Y_{n-1}=y_{n-1}\}=\frac{\{\omega p_n\}^y}{y!}\exp\{-\omega p_n\},$$

$$\Pr\{\textstyle\sum_{i=1}^{n}Y_i=y\}=\frac{\{\omega(1-\prod_{i=1}^{n}\bar{p}_i)\}^y}{y!}\exp\{-\omega(1-\prod_{i=1}^{n}\bar{p}_i)\}. \tag{5}$$

This is an NHPP with discrete time.

**2.3 Proportional Hazard Modeling**

Suppose that $l\,(\geq 1)$ kinds of software-test metrics data $\mathbf{x_i} = (x_{i1},\dots,x_{il})\ (i = 1,2,\dots,n)$ are available at $i$-th testing time, where each metrics $x_i$ is a function of time $i$ and is called the *time-dependent covariate*. In the discrete PHM, suppose that the fault-detection time $T$ given basic covariates $x_i$ has a discrete probability with mass points at $i = 1,2,\dots,n$. Let $F_n^0 = F_n^0;\theta$ represent the baseline c.d.f. of the random variable $T_j\ (=1,2,\dots)$ for $x_i = O$, where $\theta$ is the parameter in $F_n^0$ and $\lambda_{n,\theta}^0$ is its hazard rate. Then the discrete Cox regression can be represented by

$$\lambda_{n,\mathrm{X}_n;\theta,\beta} = 1 - \left(1 - \lambda_{n;\theta}^{0}\right)^{g(x_n;\beta)}, \tag{6}$$

where $\lambda_{n,\mathrm{X}_n;\theta,\beta}$ is the hazard rate of the random variable $T$ taking account of basic covariates $x_n$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_l)^t$ is the coefficient parameter. Similar to the usual Cox's PHM, an appropriate choice of the covariate function would be given by the following exponential form:

$$g(\mathrm{X}_n;\beta) = \exp(\mathrm{X}_n\beta). \tag{7}$$

Actually this form is well known to be convenient for analysis and to be rather flexible to express the covariate structure in many applications.

Shibata *et al.* [17] consider the following two fault detection probabilities:

**Model A**

$$p_{i,x_i;\theta,\beta} = 1 - \left(1 - \lambda_{i;\theta}^{0}\right)^{\exp(x_i\beta)}. \tag{8}$$

**Model B**

$$p_{i,x_i;\theta,\beta} = \left[1 - \left(1 - \lambda_{i;\theta}^{0}\right)^{\exp(x_i\beta)}\right] \prod_{v=1}^{i-1} \left(1 - \lambda_{v;\theta}^{0}\right)^{\exp(x_v\beta)}. \tag{9}$$

## 3. Development of M-SRAT

The system user can be supported for (i) model selection/ parameter estimation, (ii) the goodness-of-fit test, (iii) software reliability assessment/prediction. In (i), we call a data file of software fault data and metrics data (group data), and estimate the model parameters for selected SRMs. In M-SRAT, totally 20 SRMs (10 binomial SRMs and 10 NHPP-based SRMs) are involved, where the system user can choose a suitable SRM by comparing with the other SRMs from the various points of view. The maximum likelihood estimation is applied to estimate the model parameters. Since the maximum likelihood estimates cannot be obtained analytically in almost all cases, the numerical optimization technique for maximizing the log-likelihood function is implemented. In (ii), it is possible to derive the maximum log-likelihood function (LLF), Akaike information criterion (AIC), Bayesian information criterion (BIC) and mean squared error (MSE). Also, the Kolmogorov-Smirnov (K-S) test can be performed, where the K-S statistics is given by $\max 1 \leq \mathrm{k} \leq \mathrm{n} - 1\ D_k$, where

$$D_k = \max\left\{\left|\frac{H_{k,\omega;\theta,\beta}}{H_{n,\omega;\theta,\beta}} - \frac{y_k}{y_n}\right|, \left|\frac{H_{k,\omega;\theta,\beta}}{H_{n,\omega;\theta,\beta}} - \frac{y_{k-1}}{y_n}\right|\right\}. \tag{10}$$

In (iii), the estimates of the number of faults per unit testing time and its cumulative value are graphically plotted. Also, one assesses the quantitative software reliability which is the

probability that the software product does not fail during a specified time interval after release. In general, it is well known that the better goodness-of fit performance with the past observation data does not lead to the better prediction performance of the SRM. By dividing the observation data into two parts; training data and prediction data, the prediction ability of the selected SRM can be examined by using the prediction log-likelihood (PLL) and the prediction mean squared error (PSE). More specifically, we describe the fundamental functions of M-SRAT. Figure 1 shows the main window of M-SRAT, where the interface architecture is based on a tabbed pane with some tabs.

When the system user selects *Open* button in the menu bar labeled as *File*, a file chooser for navigating the file system appears. The data files are recorded in the CSV format that contains testing time, cumulative number of software faults and time-series metrics data in the first, second and other columns, respectively. By selecting and loading one data file (.csv), the system operation is ready.

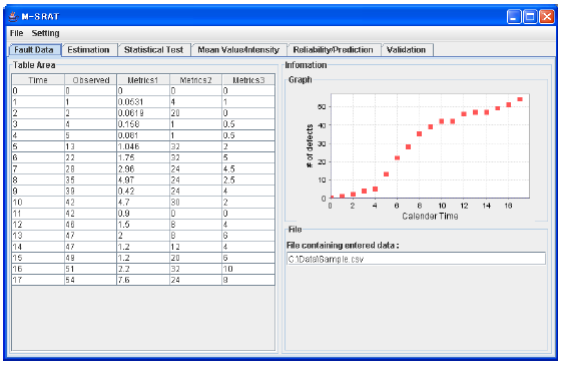


**Figure 1:** Main Window (M-SRAT).

The next step is the model selection and parameter estimation. In M-SRAT, 20 SRMs (5 binomial SRMs and 5 NHPP-based SRMs with 2 fault-detection probabilities) are prepared, where each SRM is classified by the baseline hazard function in the following:

**Geometric:**

$$\lambda_{i;b}^{0} = b, \tag{11}$$

**Negative Binomial (order 2):**

$$\lambda_{i;b,\gamma}^{0} = \frac{ib^2}{1 + b(i-1)}, \tag{12}$$

**Discrete Weibull (order 2):**

$$\lambda^0_{i;b,\gamma} = 1 - b^{i^2-(i-1)^2}, \tag{13}$$

**Negative Binomial:**

$$\lambda^0_{i;b,\gamma} = \frac{\binom{\gamma+i-2}{\gamma-1} b^\gamma (1-b)^{i-1}}{1 - \sum_{k=0}^{i-1} \binom{\gamma+k-2}{\gamma-1} b^\gamma (1-b)^{k-1}}, \gamma > 0, \tag{14}$$

**Discrete Weibull:**

$$\lambda^0_{i;b,\gamma} = 1 - b^{i^\gamma-(i-1)^\gamma}, \gamma > 0. \tag{15}$$

After clicking on *Estimation* button on the right-hand side, the estimation window appears as shown in Fig. 2.

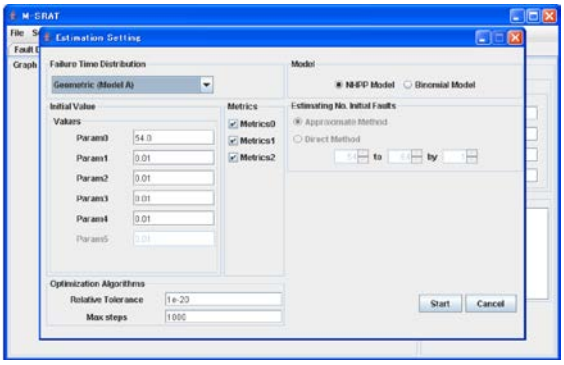


**Figure 2:** Estimation Window (M-SRAT).

First, one SRM, *i.e.*, one software fault-detection time distribution and fault-detection probability, are selected by using the combo box. The next time the user selects a binomial SRM or an NHPP-based SRM in *Model*. For estimating the binomial SRM, it requires to estimate initial number of faults, $M$ to be integer number, since $M$ takes only integer number in the binomial SRM. In M-SRAT, we apply two methods for estimating: (i) approximate method, (ii) direct method. In (i), we estimate $M$ as a real number and round it as integer. In (ii), we estimate candidates of integer $M$ in specified area. Here the user can adjust this area. Second, the kind of testing metrics data is selected with the check boxes in *Metrics*. Even if no check box is selected or there is no metrics data in the CSV file, the estimation procedure is continued but the corresponding SRM is reduced to the common binomial or the common NHPP-based SRM by ignoring the covariate structure. Third, the user can utilize an optimization technique to solve the maximum likelihood equations. In M-SRAT, we apply the Nelder-Mead simplex method [9].

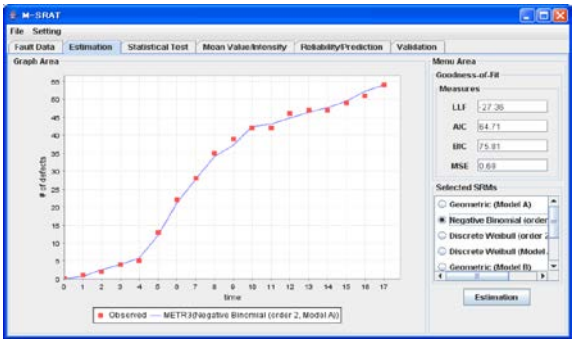


**Figure 3:** Goodness-of-fit Evaluation (MSRAT).

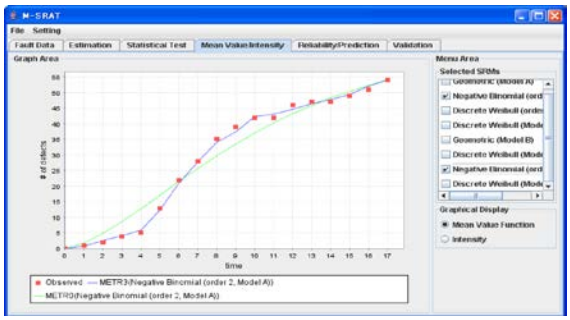


**Figure 4:** Comparison of Mean Value Functions (M-SRAT).

The user gives an initial value of each model parameter which is involved in the selected SRM, a relative tolerance level and the maximum number of steps in computation. Finally, the parameter estimation is completed by clicking on the *Start* button at the bottom of window.

The goodness-of-fit result is automatically displayed after the estimation procedure, if it was successful. Figure 3 is a snapshot of the goodness-of-fit test, where the number of software faults experienced before and the mean value function are both plotted with respect to the calendar time. At the same time, several goodness-of-fit performance measures, like LLF, AIC, BIC and MSE, are calculated. If the system user wants to know the estimated model parameters, they may appear by moving the cursor on each radio button named as the *Probability Distribution*. On the other hand, when the user estimates the parameters based on the other SRMs, it is possible to do it on the same window by clicking on the *Estimation* button again, so that we can return freely to the estimation window in Fig. 2. Once the SRM was selected, it is listed up in the panel referred to as *Selected SRMs*.

The graphical comparison among multiple SRMs is also possible.


**Figure 5:** Comparison of Mean Value Functions (M-SRAT).

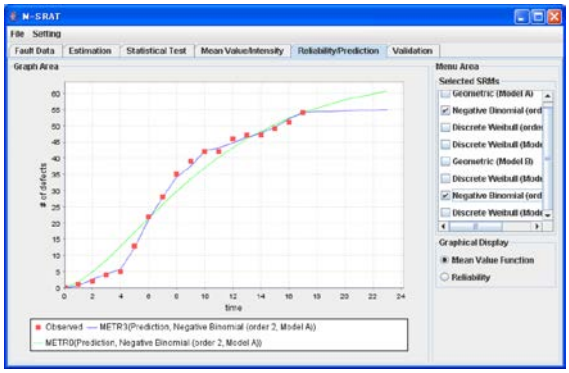


**Figure 6:** Prediction of Cumulative Number of Faults (M-SRAT).

By clicking on *Mean Value/Intensity* tab, as shown in Fig. 4, the system user can compare the multiple mean value functions on the same window. Even in this page the system user can return to the estimation phase in Fig. 2 for the comparison of mean value functions. If one also clicks on *Intensity* button, the number of software faults detected in each observation point and the baseline intensity function for the selected metrics-based SRM are plotted, where *Empirical intensity* corresponds to the number of faults per unit testing time. In the tab labeled as *Statistical Test*, the user can carry out the Kolmogorov-Smirnov test (K-S test), where the number of software faults experienced before and the mean value function are both plotted with respect to the calendar time. Also, the K-S distance, the 1% and 5% critical values, the message whether the selected SRM fits the given data set or not, are output. By cricking on *Reliability/ Prediction* tab, the system user can estimate (i) the future behavior of cumulative number of detected faults, (ii) the software reliability, from the current time. Figures 6 shows the future prediction of cumulative number of detected faults, given that the SRMs are selected. If the user changes the

prediction time length and the kind of testing metrics for analysis, the right-clicking on the check box and selecting the *Edit* menu go to the adjustment function of prediction circumstance.

## 4. Concluding Remarks

In this paper we have developed a software reliability assessment tool, called M-SRAT, by using several testing metrics data as well as software fault data observed in the testing phase. Although the theoretical framework has been introduced in the previous work [17], we have demonstrated how metrics-based SRMs could be implemented in the software reliability modeling and measurement tool. In MSRAT we implicitly assume that the number of testing metrics used for analysis is 5 or 6. However, if the number increases extremely, the parameter estimation will become much harder from view point of real-time computation. Although the Nelder-Mead simplex method is implemented in M-SRAT, this algorithm does not guarantee the global convergence property in spite of their computation efficiency. In the future, the parameter estimation module may be improved by introducing the EM algorithm.

## References

[1] Cox, D. R. *Regression Models and Life-tables*. Journal of the Royal Statistical Society, Series B, 1972; 34(2): 187–220.

[2] Evanco,W. M. *Poisson Analyses of Defects for Small Software Components*. Journal of Systems and Software, 1997; 38(1): 27–35.

[3] Ikemoto, S., and T. Dohi, *Exponential regression-based software reliability model and its computational aspect*, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences (A), 2012; E95-A (9): 1461–1468.

[4] Ikemoto, S., T. Dohi and H. Okamura. *Quantifying Software Test Process and Product Reliability Simultaneously*. Proceedings of The 24th International Symposium on Software Reliability Engineering (ISSRE'13), IEEE CPS, 2013: 108–117.

[5] Kuwa, D., and T. Dohi. *Generalized Logit-based Software Reliability Modeling with Metrics Data*. Proceedings of The 37th Annual International Computer Software and Applications Conference (COMPSAC'13), IEEE CPS, 2013: 53-58.

[6] Kuwa, D.,and T. Dohi. *Generalized Cox Proportional Hazards Regression-based Software Reliability Modeling with Metrics Data*. Proceedings of The 19th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'13), IEEE CPS, 2013: 328–337.

[7] Lyu, M. R. (ed.), *Handbook of Software Reliability Engineering*, 1996, McGraw-Hill, New York.

[8] Musa, J. D., A. Iannino and K. Okumoto. *Software Reliability, Measurement, Prediction, Application*, 1987, McGraw-Hill, New York.

[9] Nelder, J. A. and R. Mead. *A Simplex Method for Function Minimization*. Computer Journal, 1965; 7(4): 308-313.

[10] Okamura, H., Y. Etani and T. Dohi. *A Multi-factor Software Reliability Model based on Logistic Regression*. Proceedings of The 21st IEEE International Symposium on Software Reliability Engineering (ISSRE'10), IEEE CPS, 2010: 31–40.

[11] Okamura, H., Y. Etani and T. Dohi. *Quantifying the Effectiveness of Testing Efforts on Software Fault Detection with a Logit Software Reliability Growth Model*. Proceedings of

2011 Joint Conference of the 21st International Workshop on Software Measurement (IWSM'11) and The 6th International Conference on Software Process and Product Measurement (MENSURA' 11), IEEE CPS, 2011: 62–68.

[12] Okamura, H. and T. Dohi. *SRATS: Software Reliability Assessment Tool on Spreadsheet*. Proceedings of The 24th International Symposium on Software Reliability Engineering (ISSRE'13), IEEE CPS, 2013: 100–117.

[13] Okamura, H., and T. Dohi. *A Novel Framework of Software Reliability Evaluation with Software Reliability Growth Models and Software Metrics.* Proceedings of The 15th IEEE International Symposium on High Assurance Systems Engineering (HASE'14), IEEE CPS, 2014: 97–104.

[14] Ray, B., Z. Liu and N. Ravishanker. *Dynamic Reliability Models for Software using Time-dependent Covariates*. Technometrics, 2006; 4(1): 1–10.

[15] Rinsaka, K., K. Shibata and T. Dohi. *Proportional Intensity-based Software Reliability Modeling with Time Dependent Metrics.* Proceedings of The 30th Annual International Computer Software and Applications Conference (COMPSAC'06), IEEE CPS, 2006: 405–410.

[16] Shanthikumar, J. G. *Software Reliability Models: A Review*. Microelectronics & Reliability, 1983; 23(5): 903–943.

[17] Shibata, K., K. Rinsaka and T. Dohi. *Metrics-based Software Reliability Models using Non-homogeneous Poisson Processes*. Proceedings of The 17th International Symposium on Software Reliability Engineering (ISSRE'06), IEEE CPS, 2006: 52–61.

[18] Shibata, K., K. Rinsaka and T. Dohi. *PISRAT: Proportional Intensity-based Software Reliability Assessment Tool.* Proceedings of The 13th Pacific Rim International Symposium on Dependable Computing (PRDC'07), IEEE CPS, 2007: 43–52.

[19] Shibata, K., K. Rinsaka and T. Dohi. *Dynamic Software Reliability Modeling with Discrete-test Metrics; How Good is it ?.* International Journal of Industrial Engineering, 2007; 14(4): 332–339.

[20] Wightman, D. W., and A. Bendell. *Proportional Hazards Modelling of Software Failure Data*. Software Reliability; State of the Art Report (A. Bendell and P. Mellor, eds.), Pergamon Infotec, Oxford, 1986: 229–263.

**Kazuya Shibata** received B.S.E. and M.S. from Hiroshima City University and Hiroshima University, Japan, in 2005 and 2007, respectively. Since 2007, he has been working as a technical member in Sony Corporation, Japan.

**Koichiro Rinsaka** received B. of Information & Management Science and M. of Marketing & Distribution Sciences degrees from University of Marketing & Distribution Sciences, Japan, in 1998 and 2000, respectively. He also received Dr. of Engineering degree from Tottori University, Japan, in 2005. From 2002 to 2003, he was an Assistant Professor at Kobe International University. From 2003 to 2007, he was a Research Associate in the Department of Information Engineering, Graduate School of Engineering, Hiroshima University. Since 2007, he has been working as an Associate Professor in the Faculty of Business Administration, Kobe Gakuin University. His research areas include Reliability Engineering, Warranty Theory and Operations Research.

**For Tadashi Dohi's biography, please refer to page 304 of this issue.**