Communications for Statistical Applications and Methods
2013, Vol. 20, No. 6, 467–474

DOI: http://dx.doi.org/10.5351/CSAM.2013.20.6.467

Virtual Coverage: A New Approach to Coverage-Based Software Reliability Engineering

Joong-Yang Park[a], Gyemin Lee[1,a]

[a]Department of Information and Statistics, Gyeongsang National University

Abstract

It is common to measure multiple coverage metrics during software testing. Software reliability growth models and coverage growth functions have been applied to each coverage metric to evaluate software reliability; however, analysis results for the individual coverage metrics may conflict with each other. This paper proposes the virtual coverage metric of a normalized first principal component in order to avoid conflicting cases. The use of the virtual coverage metric causes a negligible loss of information.

Keywords: Software reliability, software testing, coverage metric, coverage growth function, principal component

1. Introduction

Software systems represent critical components of computer systems. Failures of a software system can cause severe consequences. Both software developers and users are concerned with the quality of software systems in regards to reliability. The reliability of a software system is improved by the detection and removal of faults resident in the software system. Software systems are tested for fault detection and removal before they are released; consequently, software testing is a key activity to improve software reliability.

Theoretically, it is impossible to execute all the possible inputs of the software system under testing. Consequently, it is nearly impossible to detect and remove all the faults in the software system. However, the developed software should be released or delivered by an appropriate time. This requirement demands software testers to perform the testing activity for a reasonable amount of time. Software developers usually determine the time to stop testing and release the software system based on estimates of reliability measures.

Software reliability growth models(SRGMs) have been proposed and applied in practice to estimate software reliability measures (Musa *et al.*, 1987; Lyu, 1996; Musa, 1999). Most SRGMs describe a relationship between a reliability measure and the testing time. Such a relationship is usually derived by modeling the fault detection and removal process during testing.

However, it was recognized that the testing time was insufficient to represent the fault detection and removal process. Numerous researchers have attempted to integrate coverage information into SRGMs. Gokhale *et al.* (1996), Malaiya *et al.* (2002), Pham and Zhang (2003), Kapur *et al.* (2006) and Park *et al.* (2008b) studied SRGMs taking a coverage growth function(CGF) into account. These SRGMs are called the coverage-based SRGMs. The CGF represents the coverage growth in terms of the testing time. Even though a CGF is embedded in each of the above mentioned coverage-based

[1] Corresponding author: Professor, Department of Information and Statistics, RINS and RIC, Gyeongsang National University, 900 Gazwa-dong Jinju 660-701, Korea. E-mail: gyemin@gnu.ac.kr

SRGMs, the SRGMs have been used without actual coverage measurement. Fujiwara *et al.* (2007) discussed the potential problems of such SRGMs and suggested that the coverage should be measured and explicitly incorporated into SRGMs. In this context Park *et al.* (2003, 2004) and Fujiwara *et al.* (2007) considered SRGMs in terms of coverage and not testing time. In order to relate such SRGMs to the testing time, Park and Fujiwara (2006), Park *et al.* (2007, 2008a) and Park and Lee (2010) studied CGFs. Crespo *et al.* (2008, 2009) recently proposed new SRGMs that simultaneously incorporate testing time and coverage.

Most multiple coverage metrics are measured during testing. Previous studies apply the CGFs and coverage-based SRGMs to each measured coverage metrics. In this approach a problem occurs when results conflict among the coverage metrics. For example, we consider the case where one SRGM fits best to the block coverage and another SRGM does so to the *p*-use coverage. Since no coverage metric is known to be superior to other coverage metrics, one way to solve such a problem is to use an optimal combination of the available coverage metrics instead of an individual coverage metric. An optimal combination of the available coverage metrics is called the virtual coverage. Section 2 empirically illustrates that the coverage metrics are highly correlated. Section 3 defines the virtual coverage. Section 4 illustrates how to use the virtual coverage for software reliability engineering. Section 5 presents the conclusions.

2. Correlation Between Coverage Metrics

Let us begin this section by defining the coverage statistically. A software system can be considered as a collection of constructs, where a construct is a basic building element of a software system. If the statement coverage is measured, constructs are statements; if *p*-use coverage is measured, constructs are *p*-uses. Therefore, the definition of a construct depends on which coverage criterion is employed for the structural testing. Let M_j be the set of all the constructs of the software system under testing for j coverage criterion. The set of constructs executed up to t testing time is denoted by $M_j(t)$. One metric for measuring the thoroughness and/or the progress of the testing is the coverage defined as $C_j(t) = |M_j(t)|/|M_j|$, where $|\cdot|$ is the cardinality of a set. Since test cases are selected randomly from the inputs domain according to the given testing profile, $M_j(t)$ and $C_j(t)$ are stochastic processes.

The increase of a coverage metric is likely to be accompanied with the increase of other coverage metrics. We now show empirically that coverage metrics are highly correlated. Vouk (1992) collected three data sets from a NASA supported project implementing sensor management in an inertial navigating system. Each data set consists of a number of executed test cases, the number of detected faults and 4 coverage metrics. The four coverage metrics are respectively block, branch, *c*-use and *p*-use coverages. One of the three data sets is considered in this paper and called DS1. Pasquini *et al.* (1996) reported a data set collected from a configuration software for an array of antennas developed by European Space Agency. The data set, called DS2, contains the values of block, branch, *c*-use and *p*-use coverages. Next, we consider a recent data set reported by Crespo et al. (2008). This data set (called DS3) was collected from a software system developed by European Space Agency. DS3 consists of the testing time and 5 coverage metrics, which are respectively all-nodes, all-arcs, all-potential-uses, all-potential-uses/du and all-potential-du-paths coverages. Table 1 shows DS3, in which the testing time is the number of executed test cases

Table 2 is the correlation matrix between 4 coverage metrics of DS1. Figure 1 shows scatter plots of 6 different combinations of 4 coverage metrics of DS1. Straight lines were fitted to each scatter plot by the least squares method. Table 2 and Figure 1 indicates that the 4 coverage metrics of DS1 are highly correlated. Table 3 and Table 4 are the correlation matrices of DS2 and DS3. All the three

Table 1: Five coverage metrics of DS3

Time	Nodes	Arcs	pu	pudu	pdu	Virtual
1	0.3068	0.2130	0.1671	0.1518	0.0741	0.2019
2	0.3377	0.2234	0.1701	0.1591	0.0781	0.2151
3	0.3579	0.2358	0.1798	0.1622	0.0815	0.2263
4	0.3893	0.2486	0.1876	0.1671	0.0843	0.2405
5	0.3993	0.2540	0.1952	0.1736	0.0890	0.2476
6	0.4013	0.2601	0.1969	0.1791	0.0923	0.2514
7	0.4174	0.2739	0.1991	0.1852	0.0943	0.2609
8	0.4213	0.2811	0.2068	0.1998	0.0989	0.2683
9	0.4457	0.2884	0.2218	0.2034	0.1024	0.2804
11	0.4674	0.3016	0.2317	0.2199	0.1107	0.2953
12	0.4740	0.3241	0.2403	0.2218	0.1174	0.3054
13	0.4796	0.3287	0.2435	0.2246	0.1188	0.3094
14	0.4817	0.3311	0.2456	0.2268	0.1199	0.3114
15	0.4849	0.3333	0.2479	0.2292	0.1209	0.3138
17	0.4901	0.3398	0.2491	0.2316	0.1236	0.3178
18	0.4977	0.3413	0.2521	0.2369	0.1257	0.3219
23	0.5013	0.3499	0.2599	0.2415	0.1292	0.3277
24	0.5089	0.3505	0.2627	0.2454	0.1327	0.3315
25	0.5123	0.3679	0.2699	0.2567	0.1378	0.3408
26	0.5269	0.3719	0.2719	0.2692	0.1457	0.3491
32	0.5384	0.3836	0.2857	0.2854	0.1529	0.3614
71	0.5747	0.4233	0.3183	0.2953	0.1692	0.3909
91	0.6404	0.4775	0.3268	0.3035	0.1763	0.4259
126	0.6808	0.5159	0.3577	0.3294	0.1931	0.4586
186	0.6836	0.5198	0.4099	0.3645	0.2106	0.4784
439	0.7158	0.5807	0.4116	0.4113	0.2492	0.5155
839	0.7158	0.5807	0.4542	0.4116	0.2501	0.5236
1240	0.7178	0.5847	0.4621	0.4156	0.2522	0.5276

Table 2: Correlation matrix of DS1

	Block	Branch	*p*-use	*c*-use	Virtual
block	1.0000	0.9992	0.9974	0.9724	0.9988
branch	0.9992	1.0000	0.9975	0.9782	0.9995
p-use	0.9974	0.9975	1.0000	0.9826	0.9991
c-use	0.9724	0.9782	0.9826	1.0000	0.9822
Virtual	0.9988	0.9995	0.9991	0.9822	1.0000

data sets support that coverage metrics are highly correlated.

A reasonable approach to this situation to derive a virtual coverage metric from the highly correlated coverage metrics and apply CGFs and coverage-based SRGMs to the virtual coverage metric. The next section proposes a method to derive a virtual coverage metric from the highly correlated coverage metrics.

3. Virtual Coverage Metric

Principal component analysis(PCA) is a statistical method that transforms a number of possibly correlated variables into a number of uncorrelated variables called principal components. The first principal component accounts for as much of the data set variation as possible. Each succeeding principal component accounts for as much of the variation as possible under the constraint that it is orthogonal to the preceding principal components. The dimension of a data set can be reduced by selecting the first few principal components to explain most variations in the data set. The first principal component will

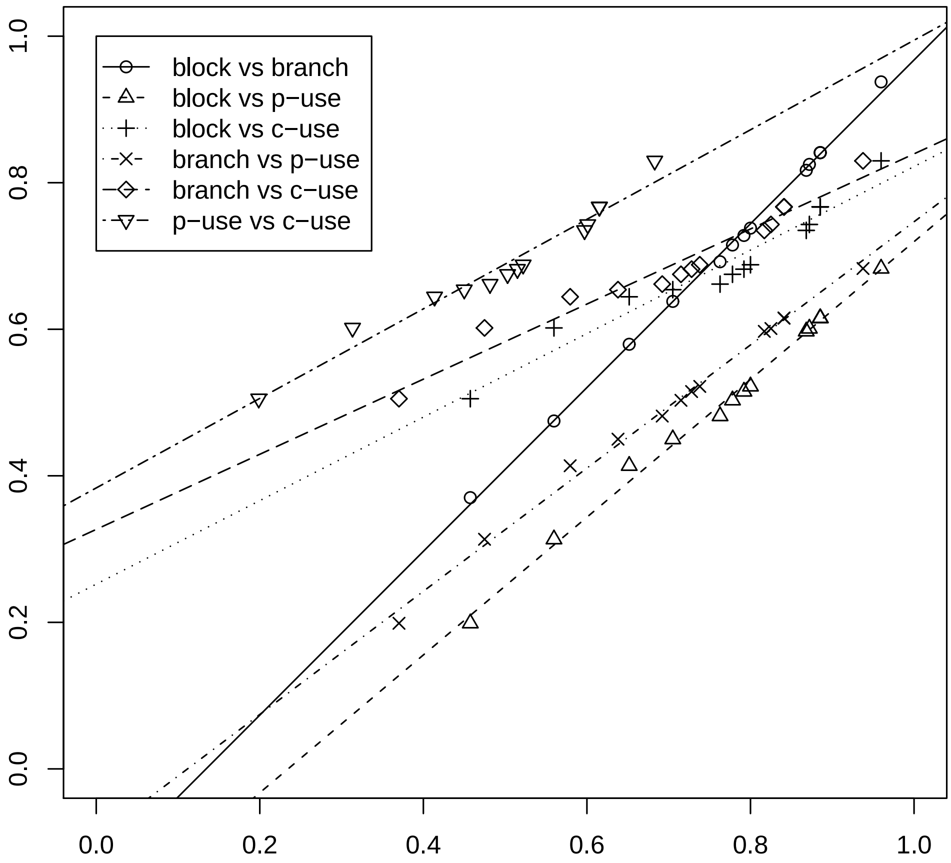

Figure 1: *Scatter plots of 6 combinations of 4 coverage metrics of DS1*

Table 3: Correlation matrix of DS2

	Block	Branch	*p*-use	*c*-use	Virtual
Block	1.0000	0.9891	0.9805	0.9967	0.9954
Branch	0.9891	1.0000	0.9968	0.9905	0.9982
p-use	0.9805	0.9968	1.0000	0.9824	0.9940
c-use	0.9967	0.9905	0.9824	1.0000	0.9963
Virtual	0.9954	0.9982	0.9940	0.9963	1.0000

account for most data set variation when the variables in the data set are highly correlated as in the multiple coverage data set; therefore, we can use the first principal component as the virtual coverage. We can consider the virtual coverage an optimal combination of the available coverage metrics since the first principal component has the maximum variance among all linear combinations of original variables.

A popular method to compute principal components is to compute eigenvectors of the covariance matrix. Eigenvalues and eigenvectors of DS1–DS3 are shown in Table 5–Table 7. The eigenvector that corresponds to the largest eigenvalue is the first principal component.

It is useful to express the first eigenvalue as a percentage of the sum of all eigenvalues. Table 8 shows the percentages that represent the relative amount of the variation explained by the first

Table 4: Correlation matrix of DS3

	Nodes	Arcs	pu	pudu	pdu	Virtual
Node	1.0000	0.9915	0.9755	0.9791	0.9755	0.9937
Arcs	0.9915	1.0000	0.9892	0.9918	0.9932	0.9988
pu	0.9755	0.9892	1.0000	0.9928	0.9936	0.9922
pudu	0.9791	0.9918	0.9926	1.0000	0.9979	0.9943
pdu	0.9755	0.9932	0.9936	0.9979	1.0000	0.9935
Virtual	0.9937	0.9988	0.9922	0.9943	0.9935	1.0000

Table 5: Eigenvalues and eigenvectors of DS1

Coverage	e_1	e_2	e_3	e_4
Block	0.5365	0.3845	0.0170	0.7510
Branch	0.6005	0.1873	0.5614	−0.5376
p-use	0.5059	−0.0798	−0.8040	−0.3023
c-use	0.3092	−0.9004	0.1953	0.2356
Eigenvalue	$6.8543e-2$	$2.9259e-4$	$4.3601e-5$	$6.2407e-6$

Table 6: Eigenvalues and eigenvectors of DS2

Coverage	e_1	e_2	e_3	e_4
Block	−0.4814	0.5144	0.6962	0.1378
Branch	−0.5341	−0.3393	0.0345	−0.7736
p-use	−0.4791	−0.6322	0.0153	0.6088
c-use	−0.5035	0.4698	−0.7168	0.1096
Eigenvalue	$6.0940e-02$	$4.0035e-04$	$4.8708e-05$	$2.7150e-05$

Table 7: Eigenvalues and eigenvectors of DS3

Coverage	e_1	e_2	e_3	e_4	e_5
Node	0.5718	0.7271	−0.2570	−0.2395	0.1451
Arcs	0.5460	0.0053	0.5185	0.6021	−0.2656
pu	0.4065	−0.4883	−0.7308	0.2458	−0.04346
pudu	0.3791	−0.3727	0.2415	−0.7167	−0.3814
pdu	0.2571	−0.3066	0.2698	−0.0780	0.8724
Eigenvalue	$4.1217e-2$	$3.1100e-4$	$5.7657e-5$	$4.1322e-5$	$2.5545e-6$

Table 8: Percentage of the total variation explained by the first principal component

DS1	DS2	DS3
99.0090	99.2246	99.5029

principal component. The first principal components of DS1–DS3 explains more than 99% of the total variation in the data sets; therefore, the dimension of DS1–DS3 can be reduced to one dimension with a negligible loss of information.

If we use the first principal component as the virtual coverage metric, the virtual coverage for DS1 is computed as

$$0.5365 * \text{block coverage} + 0.6005 * \text{branch coverage}$$
$$+\ 0.5059 * p\text{-use coverage} + 0.3092 * c\text{-use coverage.}$$

However, values of the virtual coverage (equivalently scores of the first principal component) do not in general satisfy the constraint that any coverage metric takes values between 0 and 1. Therefore, we need to normalize the first principal component so that the constraint is satisfied. One way is to divide the first principal component by the sum of components of the first principal component. We finally

Table 9: Normalized first principal components of DS1–DS3

Coverage	DS1	DS2	DS3
Block(node)	0.2748	0.2409	0.2647
Branch(arcs)	0.3076	0.2673	0.2527
p-use(pu)	0.2591	0.2398	0.1882
c-use(pudu)	0.1584	0.2520	0.1755
pdu	-	-	0.1190

Table 10: Parameter estimates and SSPEs of the lognormal CGF fitted to DS3

Parameter	Coverage metric					
	Nodes	Arcs	pu	pudu	pdu	Virtual
c_{max}	0.7528	0.6861	0.7656	0.6930	0.3802	0.6170
μ	−1.5827	−3.0686	−5.9269	−5.9500	−5.3736	−2.8336
σ	3.4601	4.2167	6.6349	6.7491	5.2210	4.4209
SSPE	0.0060	0.0055	0.0038	0.0028	0.0010	0.0022

propose the normalized first principal component as the virtual coverage metric. Table 9 shows the normalized first principal components of DS1–DS3. The virtual coverage for DS1 is then computed as

$$\begin{aligned}&0.2748 \times \text{block coverage} + 0.3076 \times \text{branch coverage}\\ &+\ 0.2591 \times p\text{-use coverage} + 0.1584 \times c\text{-use coverage}.\end{aligned}$$

The virtual coverage for DS3 were computed and presented in the rightmost column of Table 1. Correlation coefficients between the virtual coverage metric and the actual coverages metrics for DS1–DS3 are shown at the margins of Table 2–Table 4. As expected, the virtual coverage metric is highly correlated to the actual coverage metrics.

4. Application of Virtual Coverage Metric

The CGF is an important tool to evaluate the progress of testing and to determine when to stop testing. There are SRGMs into which a CGF is integrated. This section illustrates application of the virtual coverage metric to CGF.

Park and Lee (2010) fitted three CGFs to DS3 and found that the lognormal CGF

$$\begin{aligned}\widetilde{\pi}(t) &= c_{max}\,[1-\pi(t)]\\ &= c_{max}\left[1-\int_0^\infty e^{-\lambda t}\frac{e^{-(\ln\lambda-\mu)^2/2\sigma^2}}{\lambda\sigma\sqrt{2\pi}}d\lambda\right]\end{aligned}$$

showed the best performance in terms of the sum of squares of prediction error (SSPE)

$$\sum_{i=1}^{n}\left[c_{t_i}-c_{t_{i-1}}-(c_{max}-c_{t_{i-1}})\,\pi\,(t_i-t_{i-1}|t_{i-1})\right]^2, \tag{4.1}$$

where t_i is ith testing time at which coverage is measured, c_{t_i} is the coverage value at t_i, c_{max} is the maximum achievable coverage and

$$\pi\,(t|t_i) = \frac{\pi\,(t_i+t)-\pi\,(t_i)}{1-\pi\,(t_i)}. \tag{4.2}$$

The results estimates and SSPE were reproduced in Table 10 for comparison. The lognormal CGF was fitted to the virtual coverage metric (see Table 10 and Figure 2).

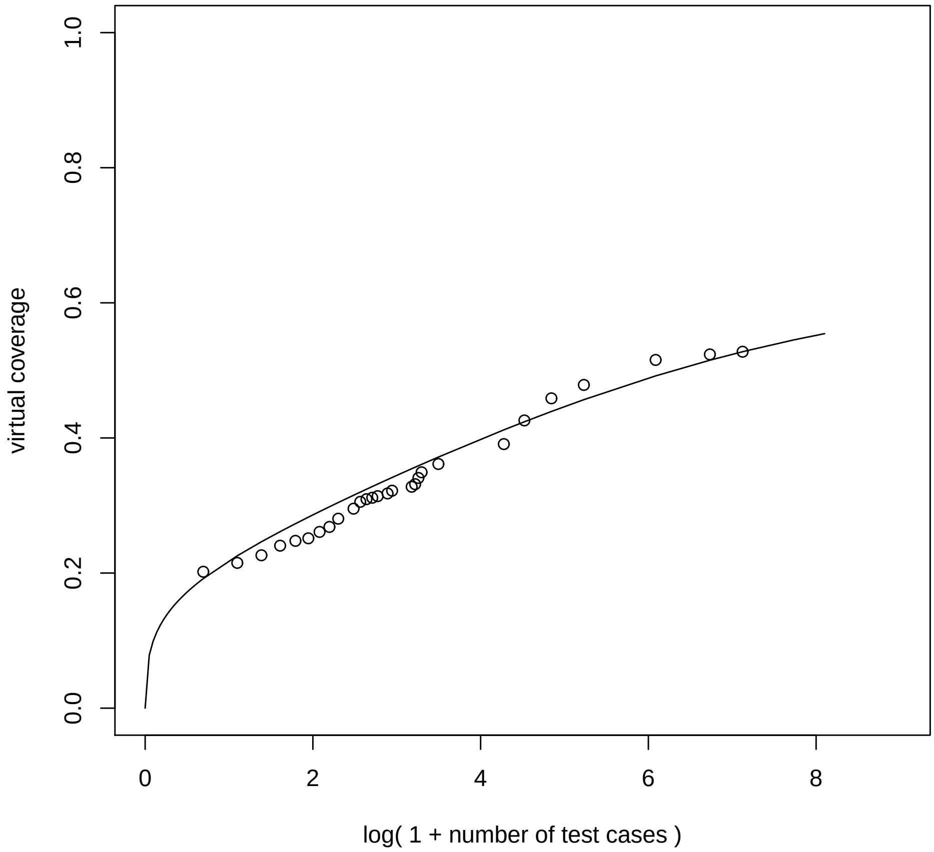

Figure 2: *The lognormal CGF fitted to the virtual coverage metric of DS3*

5. Conclusions

Coverage information is essential for software reliability engineering. Coverage-based SRGMs and CGFs have been developed and successfully applied. Software engineers usually measure multiple coverage metrics during testing and apply SRGMs and CGFs to each coverage metric; however, we may encounter the cases where analysis results for the individual coverage metrics are different from one another. We suggest the concept of a virtual coverage metric instead of the actual coverage metric to avoid conflicting cases. First, it was shown empirically that the coverage metrics are highly correlated. Then the normalized first principal component was proposed as the virtual coverage metric. The virtual coverage metric solves the confliction problem as well as lessens the analysis burden with a negligible loss of information.

References

Crespo, A. N., Pasquini, A., Jino, M. and Maldonado, J. C. (2008). A binomial software reliability model based on coverage of structural testing criteria, *Empirical Software Engineering*, **13**, 185–209.

Crespo, A. N., Pasquini, A., Jino, M. and Maldonado, J. C. (2009). Applying code coverage approach to an infinite failure software reliability model, *Proceedings of 23rd Brazilian Symposium on Software Reliability Engineering*, 216–226.

Fujiwara, T., Park, J.-Y. and Park, J. H. (2007). Evaluation and application of MVFs in coverage for coverage-based NHPP SRGM frameworks, *Proceedings of 5th ACIS International Conference of Software Engineering Research, Management & Applications (SERA2007)*, Busan Korea, 385–392.

Gokhale, S. S., Philip, T., Marinos, P. N. and Trivedi, K. S. (1996). Unification of finite failure non-homogeneous Poisson process models through test coverage, *Proceedings of 7th IEEE International Symposium on Software Reliability Engineering*, 299–307.

Kapur, P. K., Singh, O. P. and Yadav K. (2006). Software reliability growth model incorporating testing coverage and related testing effort control problem, *Communications in Dependability and Quality Management*, **9**, 132–147.

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

Malaiya, Y. K., Li, M. N., Bieman, J. M. and Karcich, R. (2002). Software reliability growth and test coverage, *IEEE Transactions on Reliability*, **51**, 420–426.

Musa, J. D. (1999). *Software Reliability Engineering: More Reliable Faster Development and Testing*, McGraw-Hill, New York.

Musa, J. D., Iannino, A. and Okumoto, K. (1987). *Software Reliability: Measurement, Prediction, Application*, McGraw-Hill, New York.

Park, J.-Y. and Fujiwara, T. (2006). Coverage growth functions for software reliability modeling, *Proceedings of 2nd Asian International Workshop on Advanced Reliability Modeling*, 435–442.

Park, J.-Y., Kim, Y. S. and Park, J. H. (2003). Modeling a relationship between reliability and software coverage, *International Journal of Reliability and Application*, **4**, 131–139.

Park, J.-Y., Kim, Y. S. and Park, J. H. (2004). Modeling the failure rate function in coverage and software reliability growth, *The Asian Journal on Quality*, **5**, 110–121.

Park, J.-Y. and Lee, G. (2010). Selection of a predictive coverage growth function, *Communications of the Korean Statistical Society*, **17**, 909–916.

Park, J.-Y., Lee, G. and Park, J. H. (2007). A class of discrete time coverage growth functions for software reliability engineering, *Communications of the Korean Statistical Society*, **14**, 497–506.

Park, J.-Y., Lee, G. and Park, J. H. (2008a). A class of coverage growth functions and its practical application, *Journal of the Korean Statistical Society*, **37**, 241–247.

Park, J.-Y., Lee, G. and Park, J. H. (2008b). A general coverage-based NHPP SRGM framework, *Communications of the Korean Statistical Society*, **15**, 875–881.

Pasquini, A., Crespo, A. N. and Matrella, P. (1996). Sensitivity of reliability-growth models to operational profile errors vs. testing accuracy, *IEEE Transactions on Reliability*, **45**, 531–540.

Pham, H. and Zhang, X. (2003). NHPP software reliability and cost models with testing coverage, *European Journal of Operational Research*, **145**, 443–454.

Vouk, M. A. (1992). Using reliability models during testing with nonoperational profile, *Proceedings of 2nd Bellcore/Purdue workshop on issues in software reliability estimation*, 103–111.

Received September 1, 2013; Revised October 14, 2013; Accepted October 21, 2013