소프트웨어 신뢰도 평가를 위한 테스트 적용범위에 대한 연구

박 중 양† · 박 재 흥†† · 박 수 진†††

요 약

소프트웨어 신뢰도는 소프트웨어 시스템의 매우 중요한 특성으로 테스팅 하는 동안 소프트웨어 신뢰도를 평가하기 위해 테스트 적용범위 정보를 이용하는 방법이 최근 시도되고 있다. 본 논문은 최근 문헌에 나타난 테스트 적용범위를 이용하는 소프트웨어 신뢰도 성장모델들을 검토하여 이들을 2개 부류로 분류한 다음 각각의 문제점을 논의하고 현실적 타당성을 검토한다. 더불어, 새로운 평균치 함수와 적절한 적용범위를 선택하기 위한 절차를 제안한다.

A Study on Test Coverage for Software Reliability Evaluation

Joong-Yang Park† · Jae-Heong Park†† · Soo-Jin Park†††

ABSTRACT

Recently a new approach to evaluation of software reliability, one of important attributes of a software system, during testing has been devised. This approach utilizes test coverage information. The coverage-based software reliability growth models recently appeared in the literature are first reviewed and classified into two classes. Inherent problems of each of the two classes are then discussed and their validity is empirically investigated. In addition, a new mean value function in coverage and a heuristic procedure for selecting the best coverage are proposed.

키워드 : 적용범위 성장함수(Coverage growth function), 평균치함수(Mean value function), 소프트웨어 신뢰도 성장모델(Software reliability growth model), 테스팅(Testing)

1. Introduction

In recent years there is growing use of computer systems. Software is an integral part of most critical computer systems. Production of a software system is one of the most complex and unpredictable activities. Failures of a software system can cause severe consequences in terms of human life, environment impact, or economical losses. Therefore, quality of software systems has become a major challenge in software development.

An important quality attribute of a software system is the degree to which it can be relied upon to perform its intended function. One of quantifible measures is software reliability. Software reliability measurement and management in the software development process are essential to produce reliable software systems efficiently and effectively. Thus, evaluation, prediction, and improvement of software reliability have been of concern to both developers and users of software systems. Significant effort has been devoted to develop methods to deliver reliable software systems. Methods proposed include well controlled software development pratices such as the cleanroom approach, verification and testing.

The cleanroom approach and software verification are the static methods for software reliability measurement and management. The static methods are based on the software size, complexity and other static parameters. They can be used even before the software is tested. The dynamic methods are based on the software failure data, which is collected from software testing. The dynamic methods are used when the software has been tested for a while and some failure data have been collected. They evaluate software reliability during the software testing phase by predicting the

† 정 회 원 : 경상대학교 수학 · 통계정보학부 통계정보학전공 교수
†† 정 회 원 : 경상대학교 컴퓨터과학과 교수
††† 준 회 원 : 경상대학교 대학원 전자계산학과
논문접수 : 2001년 4월 24일, 심사완료 : 2001년 6월 13일

general trend of the software reliability improvement.

There are at least two kinds of uncertainty in testing. First, we can not predict when a failure will occur as a result of executing a test case. Second, we do not know what effect fixing a fault will have on the software's reliability. Thus statistical approach is usually adopted to deal with the uncertainties. Statistical methods developed for estimating software reliability are called software reliability models. The current software reliability models are usually classified into two classes, namely input domain models and time domain models.

Input domain models are based on statistical testing. The models advocate that a software system should be tested using test cases generated randomly according to the operational profile of the software. Then reliability of the software system is estimated by computing the ratio of frequency of successful executions to the total number of executions. Unfortunately, the number of test cases required to obtain an accurate estimate is generally too large to be acceptable in practice.

Time domain models or software reliability growth models (SRGMs) are the models concerned with the relationship between the cumulative number of faults detected by software testing or the time interval between software failures and the time span of the software testing. Generally the time is measured by the execution time or the number of test cases executed and testing is carried out in accordance with the operational profile. A number of SRGMs have been developed for the last two decades [4, 5, 17, 22, 24].

The existing SRGMs use only a part of the information that can be collected during testing, namely the operational profile, the times between successive failures or the cumulative number of faults detected. Other information such as code complexity and test coverage are not taken into account even if they are known to heavily affect reliability. As mentioned in Horgan et al. [8], Varadan [25] and Wood [28], there is a new approach to the reliability estimation. The new approach is indicative of a general trend to improve the accuracy of reliability estimates through the use of various test coverage measures.

The use of test coverage in reliability estimation rests on the assumption that there is a strong correlation between test coverage and reliability. Several researches [1-3, 6-8, 14, 15, 20, 21] have proposed SRGMs that account for test coverage. Such SRGMs are referred to as coverage-based SRGMs.

The main purpose of this paper is to study some problems related to the current coverage-based SRGMs. All the subsequent sections are subject to the following assumptions on software testing and test coverage.

1. The software system is tested by a functional testing method.
2. The operational profile of the software system is estimated and given.
3. A coverage tool like ATAC is used during testing. Thus test coverages as well as failure times are collected.

Section 2 reviews and discusses theoretical and practical problems inherent in coverage-based SRGMs. The rest of this paper is then devoted to the mean value function in coverage. Validity of the available mean value functions is empirically examined by analyzing 5 real test data sets in Section 3. Then a new mean value function is proposed and its applicability is investigated. Finally concluding remarks are presented in Section 4.

2. Related Works

A new trend for modeling reliability is to utilize test coverage measures. This trend is based on the assumption that the more a software system is covered, the more likely reliable is the software system. The coverage-based SRGMs are thus considered as an alternative to the traditional black-box SRGMs that do not account for the structure of the software system. Several coverage-based SRGMs have appeared in the literature [1-3, 6-8, 14, 15, 20, 21]. These coverage-based SRGMs can be classified into the coverage growth-based NHPP models and the useful testing effort -based SRGMs. We next review the two classes of coverage -based SRGMs and then discuss their problems.

2.1 Coverage Growth-Based NHPP Models

2.1.1 Piwowarski, Ohba and Caruso Model

Piwowarski, Ohba and Caruso [21] proposed a SRGM taking account of test coverage. This model is derived under the following assumptions :

1. New faults are detected only when new coverage units are covered ;
2. Faults are distributed uniformly over all the coverage units ;
3. The coverage rate is proportional to the number of uncovered coverage units.

These assumptions lead us to the simultaneous differential

equations :

$$\frac{d}{dt}n(t)=\xi\{N-n(t)\} \text{ and } \frac{d}{dt}m(t)=\frac{M}{N}\frac{d}{dt}n(t). \quad (2.1)$$

where t is the testing time, ξ is the coverage rate per coverage unit, $m(t)$ is the cumulative number of faults detected up to time t, M is the initial total number of faults in the software system, $n(t)$ is the cumulative number of coverage units covered up to time t, and N is the total number of coverage units in the software system. Note also that M/N denotes the fault density per coverage unit.

By solving the simultaneous differential equations with conditions $n(0)=0$, $m(0)=0$ and $m(\infty)=M$, we have

$$n(t)=N(1-e^{-\xi t}), \quad (2.2)$$

$$m(t)=M(1-e^{-\xi t}). \quad (2.3)$$

It is not difficult to verify that $m(t)$ of equation (2.3) is isomorphic to the Goel-Okumoto NHPP model. This model being considered as a variant of NHPP model, we can regard $m(t)$ as the mean value function denoting the expected number of faults detected up to time t.

If we let $c(t)=n(t)/N$, then from equation (2.2) we have

$$c(t)=1-e^{-\xi t}. \quad (2.4)$$

which describes the growth behavior of test coverage during testing. Henceforth, we call $c(t)$ the coverage growth function. Expressing $m(t)$ in terms of $c(t)$, we obtain

$$m(c)=Mc. \quad (2.5)$$

We thus have two different expressions for the mean value function, $m(t)$ and $m(c)$. They will be called the mean value function in time and the mean value function in coverage, respectively. Piwowarski, Ohba and Caruso model is a coverage-based NHPP SRGMs with the linear mean value function in coverage and the exponential coverage growth function.

2.1.2 Logarithmic Poisson Model

The logarithmic Poisson model is based on the logarithmic Poisson execution time SRGM of Musa and Okumoto [18] whose superior predictability over other SRGMs of the same complexity has been found to be statistically significant by Malaiya, Karunanithi and Verma [16]. The logarithmic Poisson execution time model is an NHPP model whose mean value function in time is given as

$$m(t)=\beta_0\ln(1+\beta_1 t), \quad (2.6)$$

where β_0 and β_1 are parameters. Malaiya et al. [14, 15] assume that the coverage growth may also be logarithmic if the mean value function in time is described by equation (2.6). That is, for some appropriate parameters $\widetilde{\beta}_0$ and $\widetilde{\beta}_1$,

$$c(t)=\frac{\widetilde{\beta}_0}{N}\ln(1+\widetilde{\beta}_1 t). \quad (2.7)$$

We can solve equation (2.7) for t and substitute it into equation (2.6) to obtain

$$m(c)=\beta_0\ln(1+\gamma_1(e^{\gamma_2 c}-1)), \quad (2.8)$$

where $\gamma_1=\beta_1/\widetilde{\beta}_1$ and $\gamma_2=N/\widetilde{\beta}_0$.

The coverage-based NHPP SRGM characterized by equations (2.7) and (2.8) is referred to as the logarithmic Poission model. The logarithmic Poission model has been studied further by Malaiya and his colleagues [9-13].

2.1.3 Enhanced NHPP Model

Philip et al. [20] first introduced the enhanced NHPP model, which was further studied by Gokhale et al. [6, 7]. This model is derived under the following assumptions :

1. New faults are detected only when new coverage units are covered ;
2. Faults are uniformly distributed over all the coverage units ;
3. When a coverage unit is covered at time t, any fault present at the coverage unit is detected with probability $c_d(t)$;
4. Repairs are effected instantly and without introduction of new faults.

Comparing these assumptions with those for Piwowarski, Ohba and Caruso model, we can find that there is no specific assumption about coverage growth behavior and that imperfect fault detection is allowed.

The following differential equation is derived from the above assumptions.

$$\frac{d}{dt}m(t)=Mc_d(t)\frac{d}{dt}c(t). \quad (2.9)$$

Therefore, by integrating both side of the differential equation, the mean value function is obtained as

$$m(t)=M\int_0^t c_d(\tau)c'(\tau)\,d\tau. \quad (2.10)$$

Simply assuming that $c_d(t)$ is a constant α, we have

$$m(t) = ac(t), \tag{2.11}$$

where $a = Ma$ is the total number of faults expected to be detected eventually. As in Piwowarski, Ohba and Caruso model, the mean value function in coverage of the enhanced NHPP model is also obtained as a simple linear function through the origin, i.e.,

$$m(c) = ac. \tag{2.12}$$

The enhanced NHPP model does not specify the coverage growth function. Application of the enhanced NHPP model thus requires determination of an appropriate coverage growth function. Gokhale et al. [6, 7] present a method for empirically estimating the coverage growth function from the test data. Philip et al. [20] show that the coverage growth functions for the Goel-Okumoto model, the generalized Goel-Okumoto model and the s-shaped SRGM are respectively given as

$$c(t) = 1 - e^{-\xi t}, \tag{2.13}$$

$$c(t) = 1 - e^{-\xi t^{\gamma}}, \tag{2.14}$$

$$c(t) = 1 - (1 + \xi t)e^{-\xi t}, \tag{2.15}$$

where ξ in equation (2.13) is the coverage rate per coverage unit, ξ and γ in equation (2.14) are parameters reflecting the quality of testing with respect to coverage growth, and ξ in equation (2.15) is the fault removal rate. The above coverage growth functions have been obtained by interpreting the current NHPP models with respect to the enhanced NHPP model. Nontheless, the above coverage growth functions comply with the empirical results on the relation between coverage and testing time. Ramsey and Basili [23] experimented with different permutations of the same test set and collected data relating the number of tests to statement coverage growth. A variety of models were attempted to fit the data. The best fit was obtained using the Goel and Okumoto's exponential model. This result justifies the coverage growth function of equation (2.13), which is a special case of equation (2.14). Even though no empirical justification for equation (2.15) has been studied, we can select the best one for the test data on hand from the available coverage growth functions.

2.2 Useful Testing Effort-Based SRGMs

A lot of development resources are consumed by software development projects. During the software testing phases, software reliability is highly related to the testing effort, i.e., the amount of development resources spent on detecting and correcting software faults. Most of the existing SRGMs measure testing effort in calendar time, CPU-time spent in executing the software under test, the man power spent during the testing phase, and the number of test cases executed and use it as the time domain.

Wong et al. [27] empirically show that the block coverage of a test set is more highly correlated to fault detection effectiveness than the size of the test set is. Wong [26] further shows that when the size of a test set is reduced while the coverage is kept fixed, there is little or no reduction in fault detection effectiveness. These two results lead us to believe that test cases that do not increase test coverage are likely to be ineffective in detecting faults. Chen et al. [1] and Horgan et al. [8] suggest a new approach for incorporating coverage information in estimating software reliability. They begin by the notion of useful testing effort. A testing effort is said to be useful if and only if it increases some type of test coverage. Similarly, a testing effort is useless if it does not increase any type of test coverage. It is worthy of note that this definition of usefulness does not specify which coverage should be increased for a testing effort to be useful. The testing effort in the current SRGMs is then replaced with the useful testing effort. The resulting models are called the useful testing effort-based SRGMs, which are simple to use and retain the basic structure of the existing SRGMs.

Let t_k and n_k be the cumulative testing effort and cumulative number of test cases at which the kth failure occurs. Denote by e_j the testing effort spent by the jth test case. Then the cumulative testing effort spent up to kth failure is computed as $\sum_{j=1}^{n_k} e_j$. Chen et al. [1] propose that the cumulative testing effort in SRGMs be replaced with the cumulative useful testing effort

$$\sum_{j=1}^{n_k} \rho_j e_j, \quad \text{where } \rho_j = \begin{cases} 1 & \text{if } e_j \text{ is useful} \\ 0 & \text{otherwise} \end{cases}. \tag{2.16}$$

Here, ρ_j is called the compression ratio.

It has been shown empirically that a useless test case may indeed reveal faults. This implies that what we consider as a test case amounting to useless effort may indeed be a useful test case that, when run on the software system, reveal some faults. Thus the notion of useful testing effort has been extended by Chen et al. [2, 3] so that a testing effort is said to be useful if and only if it increases some type of test coverage or it reveals some faults. It is not difficult to apply

this extended notion of useful testing effort. Instead of the binary compression ratio given in equation (2.16), they suggest the following compression ratio :

$$\rho_j = \begin{cases} 1 & \text{if } e_j \text{ is useful} \\ \dfrac{\triangle e_j^2 + \triangle e_j^2 \triangle c_j^2}{\triangle e_j^2 + \triangle c_j^2 + \triangle e_j^2 \triangle c_j^2} & \text{otherwise} \end{cases}, \quad (2.17)$$

where c_j is the cumulative coverage attained up to the jth test case,

$$\triangle e_j = \begin{cases} e_1 & \text{if } j = 1 \\ e_j - e_{j-1} & \text{if } j \geq 2, \end{cases} \quad (2.18)$$

and

$$\triangle c_j = \begin{cases} c_1 & \text{if } j = 1 \\ c_j - c_{j-1} & \text{if } j \geq 2 \text{ and } c_j - c_{j-1} \neq 0 \\ \triangle c_{j-1} & \text{if } j \geq 2 \text{ and } c_j - c_{j-1} = 0 \end{cases} \quad (2.19)$$

2.3 Reliability Function of a Software System

The reliability of a software system can be assessed by computing the reliability function

$$R(s|t) = \Pr(\text{no failure occurs in } (t, t+s] \mid \text{the software has been tested up to } t). \quad (2.20)$$

Its specific formula depends on the SRGMs used. However, it should be noted that the times t and s in the above reliability function are generally the testing effort, i.e., the execution time or the number of test runs.

Let us first discuss the useful testing effort-based SRGMs with respect to the reliability function. These SRGMs describe a fault detection phenomenon in terms of the useful testing effort. One major drawback of the useful testing effort-based SRGMs is that the above reliability function can not be computed. In order to estimate $R(s|t)$ from the fitted SRGM, we should know the useful testing efforts corresponding to the testing efforts t and $(t+s)$. The useful testing effort for testing effort t can be obtained by using the test data up to t. However, the useful testing effort for testing effort $(t+s)$ can not be computed. This is because there is no information for determining whether each of test cases between t and $(t+s)$ is useful or not. This indicates that a systematic method for relating the testing effort to the useful testing effort is necessary. Without such a method, it would be impossible to estimate $R(s|t)$ by means of a useful testing effort-based SRGM.

We next discuss the coverage growth-based NHPP SRGMs. In order to completely implement the coverage growth-based NHPP models, we need to estimate from the test data both the coverage growth function $c(t)$ and the mean value function in coverage $m(c)$. The reliability function is computed as

$$R(s|t) = e^{\hat{m}(\hat{c}(t)) - \hat{m}(\hat{c}(t+s))}, \quad (2.21)$$

where $\hat{m}(c)$ and $\hat{c}(t)$ are the estimates of $m(c)$ and $c(t)$. It is clear that $R(s|t)$ relies directly on the specific coverage growth function and mean value function in coverage. Generally, the best ones are chosen among the available coverage growth functions and mean value functions in coverage. Until now only four coverage growth functions and two mean value functions in coverage have been proposed. However, their applicability to real test data sets has not been fully studied.

Suppose now that we are primarily interested in the residual number of faults or the fault density of a software system, not the reliability function. In this case, we do not need to estimate the relation between the useful testing effort and the testing effort when a useful testing effort-based SRGM is adopted. When a coverage growth-based SRGM is employed, only the mean value function in coverage needs to be estimated but the coverage growth function do not. Most of the existing SRGMs include a parameter representing the initial total number of faults. Especially, in case of the NHPP SRGMs, the mean value function includes such a parameter. In addition, size of the software system is known at the beginning of testing. Once the parameter is estimated, the residual number of faults is computed as an estimate of the initial total number of faults minus the number of detected faults. The fault density is then computed by dividing the residual number of faults by the size of the software system.

3. Mean Value Functions in Coverage

The mean value function in coverage growth-based NHPP model is the function that expresses the relation between the coverage and the expected number of faults detected during testing. It is used for modeling the fault detection phenomenon during testing. Only two types of the mean value function in coverage have appeared in the literature. The two types are given by equations (2.8) and (2.12), which are respectively called the logarithmic mean value function in coverage and linear mean value function in coverage. This section will examine empirically the validity of these mean value

functions in coverage. Then a new mean value function will be developed and applied to the real test data sets. Finally a heuristic procedure for selecting the best coverage measure is also proposed.

3.1 Real Test Data Sets

This subsection describes and presents five real test data sets that will be used in this paper. The five test data sets will be respectively referred to as DS1, DS2, DS3, DS4 and DS5. The testing effort, equivalently the testing time, is measured by the number of test cases exercised. And four coverage measures, block coverage, branch coverage, p-use coverage and c-use coverage, are recorded.

The first data set DS1 was collected experimentally by Pasquini et al. [26] from a 6,100 line C program by applying 20,000 test cases. The measurements of the number of test cases, the number of detected faults, and the four coverage measures are reproduced in <Table 1>. The coverage measures were collected using the ATAC tool. The first 1,240 test cases revealed 28 faults. Another 18,760 test cases did not find any additional faults, even though at least five more not covered by the first 1,240 test cases were very hard to reach. They perhaps belong to sections of the code intended for handling special situations or error conditions. It is also possible that they represent dead code of some form. For the 20,000 test cases, these were the coverage values obtained : block coverage : 82%, branch coverage : 70%, and p-use coverage : 67%. This is to be expected since p-use coverage is the most rigorous coverage measure and block coverage is the least. Complete branch coverage guarantees complete block coverage and complete p-use coverage guarantees complete branch coverage. This comment also holds for other data sets.

The next three test data sets, DS2, DS3 and DS4, are from a NASA supported project implementing sensor management

<Table 1> Pasquini, Crespo and Martella Data Set (DS1)

Cumulative Number of Test Cases	Cumulative Number of Detected Faults	Block Coverage	Branch Coverage	c-use Coverage	p-use Coverage
1	1	0.34	0.20	0.26	0.23
2	2	0.42	0.28	0.34	0.30
3	3	0.48	0.33	0.40	0.34
4	4	0.54	0.34	0.44	0.36
5	5	0.56	0.37	0.48	0.37
6	6	0.56	0.39	0.49	0.39
7	7	0.57	0.40	0.49	0.39
8	8	0.58	0.41	0.50	0.40
9	9	0.58	0.42	0.50	0.41
11	10	0.59	0.44	0.51	0.42
12	11	0.59	0.44	0.51	0.43
13	12	0.59	0.45	0.52	0.44
14	13	0.60	0.45	0.52	0.44
15	14	0.60	0.46	0.54	0.45
17	15	0.62	0.46	0.56	0.45
18	16	0.64	0.48	0.58	0.46
23	17	0.69	0.52	0.60	0.49
24	18	0.70	0.53	0.62	0.50
25	19	0.71	0.55	0.65	0.52
26	20	0.71	0.55	0.65	0.53
32	21	0.72	0.58	0.66	0.55
71	22	0.74	0.60	0.66	0.56
91	23	0.76	0.64	0.70	0.62
126	24	0.77	0.66	0.72	0.64
186	25	0.78	0.66	0.72	0.64
439	26	0.79	0.67	0.72	0.65
839	27	0.80	0.68	0.73	0.66
1240	28	0.80	0.68	0.73	0.66
20000	28	0.82	0.70	0.74	0.67

<Table 2> Vouk Data Sets

(1) DS2

Cumulative Number of Test Cases	Cumulative Number of Detected Faults	Block Coverage	Branch Coverage	p-use Coverage	c-use Coverage
1	0	0.4574	0.3702	0.1987	0.5052
3	1	0.5597	0.4748	0.3134	0.6018
10	2	0.6517	0.5795	0.4137	0.6443
20	3	0.7050	0.6380	0.4500	0.6540
30	3	0.7628	0.6921	0.4815	0.6616
42	4	0.7780	0.7150	0.5030	0.6750
50	5	0.7920	0.7280	0.5150	0.6820
55	6	0.8000	0.7380	0.5220	0.6880
100	6	0.8682	0.8169	0.5971	0.7349
150	7	0.8720	0.8250	0.6010	0.7430
300	7	0.8853	0.8410	0.6152	0.7670
382	8	0.8853	0.8410	0.6152	0.7670
796	9	0.8853	0.8410	0.6152	0.7670
1196	9	0.9597	0.9376	0.6829	0.8298

(2) DS3

Cumulative Number of Test Cases	Cumulative Number of Detected Faults	Block Coverage	Branch Coverage	p-use Coverage	c-use Coverage
10	2	0.7331	0.7080	0.6392	0.7820
15	3	0.7700	0.7380	0.6550	0.8100
30	4	0.8563	0.8285	0.7102	0.8982
50	5	0.8800	0.8550	0.7400	0.9080
100	5	0.9208	0.9160	0.8131	0.9319
155	6	0.9300	0.9210	0.8280	0.9350
300	7	0.9355	0.9343	0.8691	0.9438
796	7	0.9355	0.9343	0.8691	0.9438

(3) DS4

Cumulative Number of Test Cases	Cumulative Number of Detected Faults	Block Coverage	Branch Coverage	p-use Coverage	c-use Coverage
3	2	0.5993	0.5566	0.4439	0.7083
4	3	0.6130	0.5700	0.4550	0.7200
10	4	0.6939	0.6415	0.5164	0.7881
20	5	0.7780	0.7200	0.5710	0.8330
30	6	0.8561	0.7901	0.6285	0.8763
44	7	0.8700	0.8100	0.6510	0.8810
100	7	0.9217	0.8868	0.7395	0.9118
114	8	0.9240	0.8900	0.7470	0.9125
160	9	0.9350	0.9040	0.7720	0.9150
300	9	0.9599	0.9387	0.8459	0.9229
796	9	0.9599	0.9387	0.8459	0.9229

in the inertial navigation system. The three data sets are presented in <Table 2>. They were obtained by testing three separate implementations of the sensor management system. Each implementation is about 5,000 lines of code. In the first implementation, 1,196 test cases found 9 faults. For the other two implementations, 796 test cases revealed 7 and 9 faults respectively. The number of test cases executed and four coverage measures were collected at each fault detection during testing. However, <Table 2> includes a few additional values on these measures observed at some other testing times.

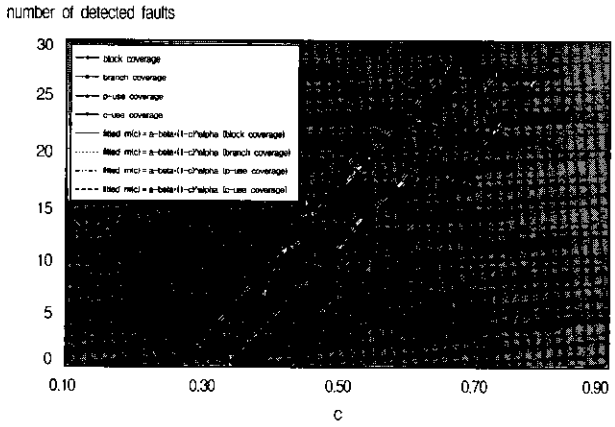
The last test data set DS5 was obtained from Dr. Mathur in Purdue university through personal communication. The data, shown in <Table 3>, was collected by testing a software system with 22 injected faults. 109 test cases were applied to the software system and 17 of the 22 injected faults were discovered. In some test cases, one test case detected more than one fault.

〈Table 3〉 Mathur Data Set (DS5)

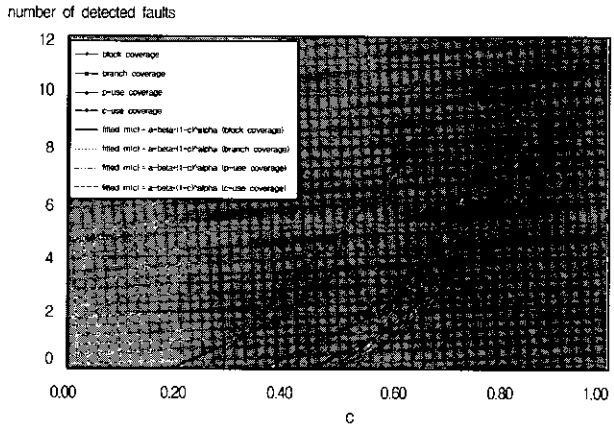
Cumulative Number of Test Cases	Cumulative Number of Detected Faults	Block Coverage	Branch Coverage	p-use Coverage	c-use Coverage
3	6	0.48	0.33	0.34	0.40
12	7	0.59	0.44	0.43	0.51
14	9	0.60	0.45	0.44	0.52
16	11	0.60	0.46	0.45	0.52
24	12	0.70	0.53	0.50	0.62
67	14	0.73	0.60	0.56	0.66
100	15	0.77	0.65	0.63	0.72
101	16	0.77	0.65	0.63	0.72
109	17	0.78	0.66	0.64	0.72

3.2 Examination of the Existing Mean Value Functions in Coverage

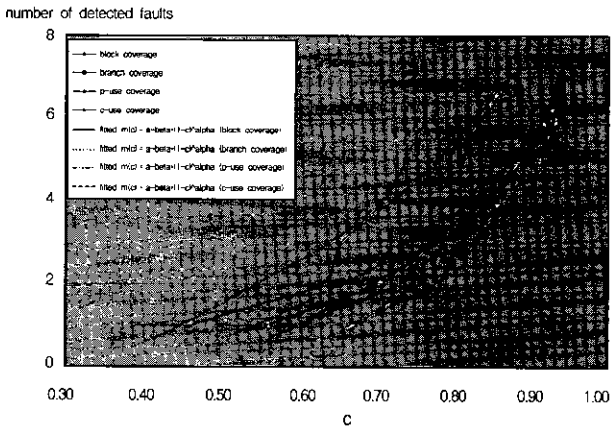
We begin by presenting the figures that show the relation between the number of detected faults and the coverage. The number of detected faults is depicted against the coverage in (Figures 1- 5) for the five test data sets. First, we can find that the relation does not much depend on which of the four coverages is considered. Whatever coverage is chosen,



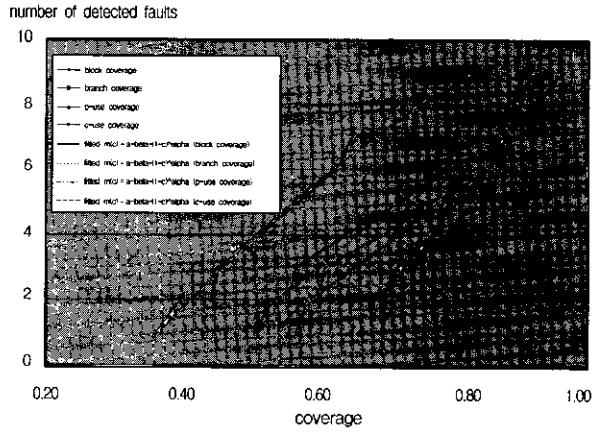
(Figure 1) Plots of the number of detected faults against the coverage and the fitted mean value functions (DS1)



(Figure 2) Plots of the number of detected faults against the coverage and the fitted mean value functions (DS2)

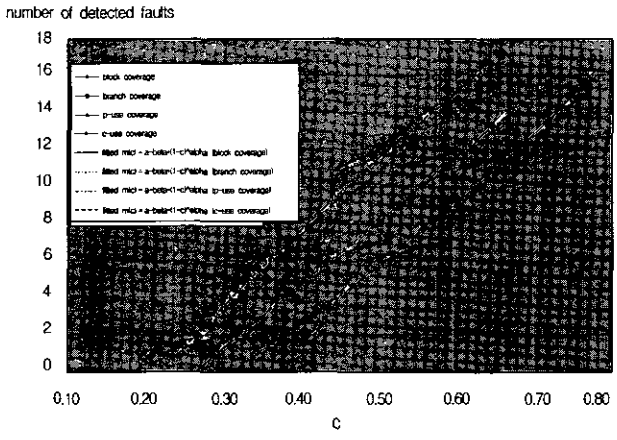


(Figure 3) Plots of the number of detected faults against the coverage and the fitted mean value functions (DS3)



(Figure 4) Plots of the number of detected faults against the coverage and the fitted mean value functions (DS4)

similar aspects of the relation are derived. For example, (Figures 1- 3) indicate a nonlinear relation, while (Figures 4 and 5) indicate a linear relation.

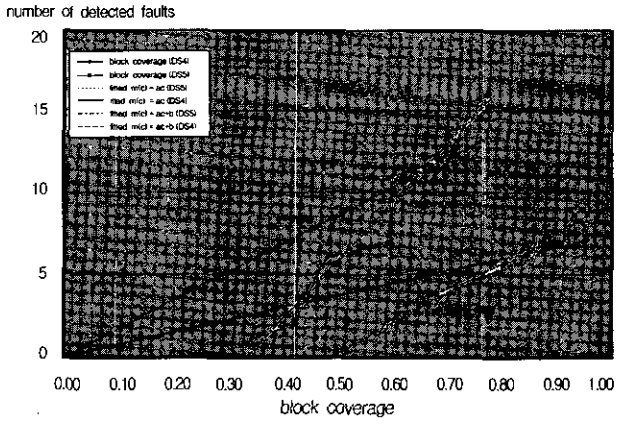


(Figure 5) Plots of the number of detected faults against the coverage and the fitted mean value functions (DS5)

Furthermore, the relation seems to be well approximated by a linear function at the later phase of testing, that is, when coverage is roughly higher than about 0.5.

Malaiya et al. [14, 15] have shown that the logarithmic mean value function in coverage approximates well the relation between the number of detected faults and the test coverage for DS1 - DS4. No further empirical validation seems to be necessary. However, the logarithmic mean value function in coverage lacks theoretical basis, i.e., it is based on the intuition that coverage growth is likely to be described by the logarithmic Poisson model if the number of faults detected follow the logarithmic Poisson execution time SRGM. Since there is no theoretical justification for this logarithmic mean value function in coverage, it is not easy to modify and refine the logarithmic mean value function in coverage for the cases where its performance is not good enough. For example, what if the number of detected faults does not follow the logarthmic Poisson execution time SRGM?

We now investigate validity of the linear mean value function. So far, this model has never been applied to real test data sets. We pointed that, some nonlinear model seems to be appropriate for DS1 - DS3 and that DS4 and DS5 seems to support a linear mean value function. The linear mean value function is thus fitted to DS4 and DS5 and the results for the block coverage are presented in (Figure 6). Apparently the linear mean value function in coverage does not perform well even for DS4 and DS5. This is mainly because the linear mean value function in coverage is forced to pass through the origin. The figures do not include the trivial observation that the coverage is zero when no test case is executed. That is, the plots in the figures are not connected with the origin. The plots suggest existence of a certain relationship between the number of detected faults and the coverage when the coverage c is greater than or equal to c_1, where c_1 is the coverage attained by execution of the first test case. No faults can be found before execution of the first test case, i.e., $m(c) = 0$ for $0 < c_1$. We should focus on a non-trivial relation between the number of detected faults and the coverage for $c \geq c_1$. The test cases are selected randomly according to the operational profile. The test coverage attained by the first test case, c_1, is also random. Therefore, the mean value function $m(c)$ is considered to be defined for $c \geq c^*$, where c^* is the expected value of c_1. Thus, the linear mean value function with intercept, $m(c) = ac + b$, is suggested as an alternative to $m(c) = a$. For the sake of reference, the results obtained by fitting $m(c) = ac + b$ to DS4 and DS5 are also shown in (Figure 6). (Figure 6) strongly indicates that the linear mean value function in coverage with intercept works much better than the linear mean value function in coverage without intercept. Malaiya et al. [14, 15] also suggest $m(c) = ac + b$ as an approximation to the logarithmic mean value function in coverage when c is large enough.



(Figure 6) Mean value functions m(c) = ac and m(c) = ac + b fitted to DS4 and DS5

3.3 A New Mean Value Function in Coverage

We first restate the common assumptions for Piwowarski, Ohba and Caruso model and the enhanced NHPP models.

1. New faults are detected only when new coverage units are covered.
2. Faults are distributed uniformly over all the coverage units.

It has been shown empirically that the covered coverage units can contain faults. In this case, the fault density of the covered coverage units is generally much lower than that of the uncovered coverage units. The first assumption postulates that the fault density of the covered coverage units is negligible. The second assumption implies that each and every coverage unit has an equal fault density. This uniform fault density assumption is just a simplification of reality, since generally different coverage units have different fault densities. Even though the uniform fault density assumption holds approximately, it seems to be more reasonable to adjust the fault density as the testing proceeds.

We now generalize the second assumption so that the fault density of the uncovered coverage units is adjusted. If the first assumption remains unchanged, all the remaining faults would exist in the uncovered coverage units. We can adjust the fault density by applying the uniform density assumption to the remaining faults and the uncovered coverage units. That is, we suppose that the fault detection phenomenon is subject to the following two assumptions :

1. New faults are detected only when new coverage units are covered.
2. Remaining faults are distributed uniformly over all the uncovered coverage units.

If the coverage growth is governed by $c(t)$, the above assumptions lead us to the following differential equation :

$$\frac{d}{dt}m(t) = a\frac{a-m(t)}{1-c(t)}\frac{d}{dt}c(t), \qquad (3.1)$$

where a is the total number of fault expected to be detected eventually and α is the failure occurrence rate per fault. Solving this differential equation, we obtain the mean value function in coverage as

$$m(c) = a-\beta(1-c)^{\alpha}. \qquad (3.2)$$

The suggested mean value function given by equation (3.2) is flexible to some extent. The suggested mean value function becomes the linear mean value function with intercept when $\alpha = 1$. Therefore, the linear mean value function with or without intercept is a special case of the suggested model. The shape of the suggested mean value function is determined by the value of α. The shape is convex when $0 < \alpha < 1$ and concave when $\alpha > 1$.

3.4 Application to Real Test Data Sets

The new mean value function proposed in the previous subsection was applied and fitted to the test data sets DS1 - DS5. The fitting was performed by using the nonlinear least squares procedure of the SAS system. Estimation results are summarized in <Table 4>. *SSE* and *TSS* in <Table 4> respectively denote the residual sum of squares and the total sum of squares. One simple method for evaluating goodness of fit is to compute *SSE*/*TSS*, which can be interpreted as

<Table 4> Least Square Estimates of Parameters in the Suggested Mean Value Function in Coverage Fitted to DS1 - DS5

(1) DS1

parameter	coverage measure			
	block coverage	branch coverage	p-use coverage	c-use coverage
a	45.0004	60.6242	46.9881	47.0045
beta	74.9995	73.5680	67.3136	68.9970
alpha	0.9000	0.6886	1.1365	0.9340
SSE	149.2053	62.7881	76.3115	131.8749
TSS	2002.9655			

(2) DS2

parameter	coverage measure			
	block coverage	branch coverage	p-use coverage	c-use coverage
a	13.2402	13.1650	20.2980	21.8790
beta	19.0368	18.1492	24.8682	35.0258
alpha	0.5296	0.6057	0.6740	0.6092
SSE	8.6320	8.5064	10.1194	10.2940
TSS	110.0000			

(3) DS3

parameter	coverage measure			
	block coverage	branch coverage	p-use coverage	c-use coverage
a	18.0000	17.0000	17.6882	16.7522
beta	20.6000	19.0000	19.9063	20.4638
alpha	0.2099	0.2195	0.2964	0.2333
SSE	1.9261	1.9346	2.1617	2.4439
TSS	46.9000			

(4) DS4

parameter	coverage measure			
	block coverage	branch coverage	p-use coverage	c-use coverage
a	10.7354	10.0335	10.2335	14.6050
beta	16.0849	17.3302	18.3184	24.9001
alpha	0.6847	0.9842	1.5005	0.5841
SSE	2.8259	2.3409	2.1089	7.5300
TSS	101.2308			

(5) DS5

parameter	coverage measure			
	block coverage	branch coverage	p-use coverage	c-use coverage
a	34.6027	28.4895	24.7924	27.6890
beta	42.6246	34.6134	35.2314	35.1072
alpha	0.5557	0.9644	1.4020	0.8632
SSE	10.9143	9.8560	10.3703	11.2875
TSS	302.0000			

the proportion of *TSS* not explained by the model. The smaller *SSE*/*TSS* is, the better is goodness of fit. We can say that the proposed model performs well for all the data sets with respect to *SSE*/*TSS*. For visual inspection of goodness of fit, the estimated mean value functions are overlaid in (Figures 1- 5). Figure 1 reveals some discrepancy between the actual and estimated number of detected faults for the early phase of testing. To be coservative, we exclude DS1 from the forthcoming discussion in this subsection. Judging from <Table 4> and (Figures 2-5), we conclude that the suggested mean value function in coverage is an adequate model for other four data sets.

Irrespective of the mean value function in coverage chosen for analysis, we still need to determine which coverage is the best for estimating the mean value function when several coverages are collected during testing. A heuristic procedure for dealing with this problem is now presented.

1. Classify the coverages into groups with respect to the estimates of the most important parameter such as a.
2. If goodness of fit of the groups are considerably different, we choose the group with higher goodness of fit.
3. If goodness of fit of the groups are not considerably different, we choose the group containing more rigorous coverages.
4. If there are more than one coverage in the chosen group, we choose the most rigorous coverage.

Let us examplify the procedureby using <Table 4>. First consider DS2. Use of block and branch coverage produces estimates of a similar to each other. So does use of p-use and c-use coverages. Considering four estimates of a, we can divide four coverages into two groups. One group consists of block and branch coverages ; the other group, p-use and c-use coverages. Since estimates of a for the two groups are somewhat significantly different, we have to decide which group is better for reliability evaluation and testing management. For DS2, difference between *SSE*s of the two groups are less than 1.7% of *TSS*, which is almost negligible. Since p-use coverage is more rigorous than block and branch coverages, we choose the group with p-use and c-use coverage. However, there is no subsume relations between p-use and c-use coverages, we may use the estimates obtained by using either p-use coverage or c-use coverage.

Next we consider DS3. Four coverages provide us with estimates close to one another and show equivalent goodness of fit. Thus we do not have to apply the heuristic procedure. Any of the four coverages can be used for estimating the mean value function.

For DS4, we can constitute two groups of coverages. One group consists of block, branch and p-use coverages and the other group consists of c-use coverage. Difference between *SSE*s of the two groups ranges from 4.65% of *TSS* to 5.36% of *TSS*. It seems reasonable to choose the group with three coverages. Furthermore, since p-use coverage is known to be more strict than block and branch coverages, the estimates corresponding to p-use coverage is perferable. Similarly, we may use the estimates obtained by using either p-use coverage or c-use coverage for DS5. Since DS5 was collected from testing of software system with 22 injected faults, we can say that p-use coverage gives the best estimates.

4. Concluding Remarks

There is a new trend for the estimation of software reliability. The new approach takes into account the structural coverage obtained during testing. This paper is concerned with the coverage-based SRGMs, SRGMs incorporating test coverage. The followings are the summary of this paper.

1. Implementation of the useful testing effort-based SRGMs requires a study on the relation between the testing effort and the useful testing effort.
2. The coverage growth-based NHPP models are characterized by the mean value function in coverage and the coverage growth function.
3. The linear mean value function in coverage is to be modified to include the intercept term.
4. The mean value function in coverage newly suggested in this paper works well.
5. A heuristic procedure for selecting the most appropriate coverage is suggested.

The last three results were verified by investigating real test data sets.

Since tools for collecting coverage information are available, the coverage-based SRGMs have emerged as a new technique for evaluating software reliability. It should be

standard to gather coverage values during testing. More efforts should be devoted to development of new coverage-based SRGMs and improvement of the existing coverage-based SRGMs. Especially, for the coverage growth-based NHPP models, more effective mean value functions in coverage and coverage growth functions are to be developed. The useful testing effort-based SRGMs have not been applied to real test data sets. This is mainly because relation between the testing effort and the useful testing effort is required for practical application and no such study has been published yet. Future research should be directed to development of models relating the useful testing effort to the testing effort.

Bibliograph

[1] M. H. Chen, J. R. Horgan, A. P. Mathur, and V. J. Rego, "A Time/Structure Based Model for Estimating Software Reliability Estimation," Technical Report SERC-TR-117-P, Purdue University, Dec. 1992.

[2] M. H. Chen, M. R. Lyu and W. E. Wong, "An Empirical Study of the Correlation Between Code Coverage and Reliability Estimation," Proceedings of the 3rd IEEE International Symposium on Software Metrics, Berlin, Germany, March 1996.

[3] M. H. Chen, M. R. Lyu and W. E. Wong, "Incorporating Code Coverage in the Reliability Estimation for Fault-Tolerant Software," Proceedings of the 16th IEEE Symposium on Reliable Distributed System, pp.45-52, Durham, NC, Oct. 1997.

[4] A. L. Goel, "Software Reliability Model : Assumptions, Limitations, and Applicability," IEEE Transactions on Software Engineering, Vol.SE-11, No.12, pp.1411-1423, 1985.

[5] S. S. Gokhale, P. N. Marinos, and K. S. Trivedi, "Important Milestones in Software Reliability Modeling," Communications in Reliability, Maintainability and Serviceability, 1996.

[6] S. S. Gokhale, T. Philip, P. N. Marinos, and K. S. Trivedi, "Non-Homogeneous Markov Software Reliability Model with Imperfect Repair," Technical Report TR-96/12, CACC Duke University, 1996.

[7] S. S. Gokhale, T. Philip, P. N. Marinos, and K. S. Trivedi, "Unification of Finite Failure Non-Homogeneous Poisson Process Models Through Test Coverage," Technical Report TR-96/36, CACC Duke University, 1996.

[8] J. R. Horgan, A. P. Mathur, A. Pasquini, and V. J. Rego, "Perils of Software Reliability Modeling," Technical Report SERC-TR-160-p, Software Engineering Research Center, Purdue University, 1995.

[9] N. Li and Y. K. Malaiya, "Fault Exposure Ratio Estimation and Application, "Technical Report CS-96-130, Colorado State University, 1996.

[10] M. N. Li and Y. K. Malaiya, and J. Denton, "Estimating the Number of Defects : A simplified Intuitive Approach, "Proceedings of International Symposium of Software Engineering, Paderborn, Germany, Nov. 1998.

[11] Y. K. Malaiya, "Estimating The Number of Residual Defects," Proceedings of the 3rd IEEE International High-Assurance Systems Engineering Symposium, Washington DC, pp.98-105, Nov. 1998.

[12] Y. K. Malaiya and J. Denton, "What Do the Software Reliability Growth Model Parameters Represent?" Proceedings of the 8th International Symposium on Software Reliability Engineering, Albuquerque, NM, pp.124-135, Nov. 1997.

[13] Y. K. Malaiya and J. Denton, "Estimating Defect Density Using Test Coverage," Technical Report CS-98-104, Colorado State University, 1998.

[14] Y. K. Malaiya, N. Li, J. Bieman, R. Karcich, and R. Skibbe, "The Relationship Between Test Coverage and Reliability," Proceedings of the 5th International Symposium on Software Reliability Engineering, pp.186-195, Monterey, CA, Nov. 1994.

[15] Y. K. Malaiya, N. Li, J. Bieman, R. Karcich, and R. Skibbe, "Software Test Coverage and Reliability," Technical Report CS-96-128, Colorado State University, 1996.

[16] Y. K. Malaiya, N. Karunanithi, and P. Verma, "Predictability of Software Reliability Models," IEEE Transactions on Reliability, pp.539-546, 1992.

[17] J. D. Musa, A. Iannino, and K. Okumoto, Software Reliability : Measurement, Prediction, Application, McGraq-Hill, 1987.

[18] J. D. Musa and K. Okumoto, "A Logarithmic Poisson Execution Time Model for Software Reliability Measurement," Proceedings of the 7th International Conference on Software Engineering, pp.230-238, Orlando, 1984.

[19] A. Pasquini, A. N. Crespo and P. Matrella, "Sensitivity of Reloability-Growth Models to Operational Profile Errors vs Testing Accuracy," IEEE Transactions on Reliability, Vol. 45, No.4, pp.531-540, 1996.

[20] T. Philip, P. N. Marinos, K. S. Trivedi, and J. Lala, "A Multiphase Software Reliability Model : From Testing to Operational Phase," Technical Report TR-96-01, CACC Duke University, 1996.

[21] P. Piwowarski, M. Ohba and J. Caruso, "Coverage Measurement Experience During Function Test," Proceedings of the 15th International Conference on Software Engineering, pp.287-300, Baltimore, MD, May 1993.

[22] C.V. Ramamoorthy and F.B. Bastani, "Software Reliability -Status and Perspective," IEEE Transactions on Software Engineering, Vol.SE-8, No.8, pp.354-371, 1982.

[23] J. Ramsey and V. R. Basili, "Analyzing the Test Process Using Structural Coverage," Proceedings of the 8th International Conference on Software Engineering, pp.306-312, Aug. 1985.

[24] J. G. Shanthikumar, "Software Reliability Model : A Review," Microelectronics and Reliability, Vol.23, No.5, pp.903-943, 1983.

[25] G. S. Varadan, "Trends in Reliability and Test Strategies," IEEE Software, Vol.12, No.3, pp.10, 1995.

[26] W. E. Wong, "On Mutation and Data Flow," PhD Thesis, Department of Computer Science, Purdue University, W. Lafayette, IN, Dec. 1993.

[27] W. E. Wong, J. R. Horgan, S. London, and A. P. Mathur, "Effect of Test Set Size and Block Coverage on the Fault Detection Effectiveness," Proceedings of the 5th IEEE International Symposium on Software Reliability Engineering, pp.230-238, Monterey, CA. Nov. 1994.

[28] A. Wood, "Software Reliability Growth Models : Assumptions vs. Reality," Proceedings of the 8th International Symposium on Software Reliability Engineering, pp.136-141, Albuquerque, New Mexico, Nov. 1997.

박 중 양

e-mail : parkjy@nongae.gsnu.ac.kr

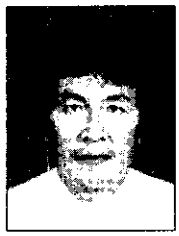
1982년 연세대학교 응용통계학과(학사)

1984년 한국과학기술원 산업공학과 응용통계전공(석사)

1994년 한국과학기술원 산업공학과 응용통계전공(박사)

1985년~현재 경상대학교 수학·통계정보학부 통계정보학전공 교수

관심분야 : 소프트웨어 신뢰성, 신경망, 선형통계 모형, 실험계획법 등



박 재 흥

e-mail : pjh@nongae.gsnu.ac.kr

1978년 충북대학교 수학교육과(학사)

1980년 중앙대학교 대학원 전산학과(석사)

1988년 중앙대학교 대학원 전산학과(박사)

1983년~현재 경상대학교 컴퓨터과학과 교수

관심분야 : 소프트웨어 신뢰성, 시험도구 자동화 등

박 수 진

e-mail : lelia@thrunet.com

1990년 경상대학교 전산통계학과(학사)

1995년 경상대학교 대학원 전자계산학과(석사)

1998년~현재 경상대학교 대학원 전자계산학과(박사수료)

관심분야 : 소프트웨어 신뢰성, 소프트웨어 테스팅 등