Journal of the KSQC
Vol. 17, No. 2, 1989

減少하는 故障率하에서 오류예측 및 테스트 時間의 最適化에 관한 硏究

Error Forecasting & Optimal Stopping Rule under Decreasing Failure Rate

崔 明 鎬*
尹 德 均**

ABSTRACT

This paper is concerned with forecasting the existing number of errors in the computer software and optimizing the stopping time of the sottware test based upon the forecasted number of errors.

The most commonly used models have assessed software reliability under the assumption that the software failure rate is proportional to the current fault content of the software but invariant to time since software faults are independents of others and equally likely to cause a failure during testing. In practice, it has been observed that in many situations, the failure rate decrease. Hence, this paper proposes a mathematical model to describe testing situations where the failure rate of software limearly decreases proportional to testing time. The least square method is used to estimate parameters of the mathematical model.

A cost model to optimize the software testing time is also proposed. In this cost model two cost factors are considered. The first cost is to test execution cost directly proportional to test time and the second cost is the failure cost incurred after delivery of the software to user. The failure cost is assumed to be proportional to the number of errors remained in the software at the test stopping time. The optimal stopping time is determined to minimize the total cost, which is the sum of test execution cast and the failure cost.

A numerical example is solved to illustrate the proposed procedure.

* 蔚山專門大學 工業經營科 敎授
** 漢陽大學校 產業工學科 敎授

Ⅰ. 序 論

1. 研究의 目的

온라인 실시간시스템, 데이터 베이스 시스템, 컴퓨터 네트웍 및 분산처리시스템등 대규모 시스템의 보급에 있어서 소프트웨어의 信賴性에 대한 요구가 점증되고 있다. 本 研究에서는 소프트웨어의 테스트에서 필수적으로 제기되는 두가지의 問題 즉,

① 테스트가 끝난 시점에서 殘存하는 오류의 수를 예측하는 문제

② 예측을 바탕으로 테스트의 완료시점을 정하는 문제

를 해결하고자 한다. 하드웨어의 故障은 마모현상이나 우연에 의해서 유발되지만 소프트웨어의 故障은 소프트웨어 그 자체 내부에 현존하고 있는 오류가 특수한 入力資料에 의해서 현재화될 때 즉, 故障이 일어 나게된다. 오류가 있다해서 전부 故障이 발생되는 것이 아니며 특정한 자료가 入力되지 않아서 오류와 資料가 만날 수 없을때 故障은 영원히 나타나지 않을 수도 있는 것이다. 결과적으로 소프트웨어를 완벽하게 테스트하기 위해서는 모든 가능한 入力에 대해서 소프트웨어를 테스트 해야 하나, 이는 불가능하므로 소프트웨어의 오류가 있음을 입증할 수는 있어도 오류가 없음을 입증하기는 현재의 기술로도 불가능한 실정이다. 오직 가능한 것이 있다면 現存하는 오류의 갯수를 그 때까지 테스트한 결과를 기초로 예측하는 것이며, 이 예측된 결과에 따라서 어느정도 만족할만한 수준의 오류가 殘存하는 狀態로 출고할 수 밖에 없는 실정이다. 이러한 인식 아래 최근에 소프트웨어의 信賴性과 有用性과 관련하여 많은 論文들이 오류예측의 수학적 모델에 집중되고 있으나 현상을 완전히 해결하는데는 미흡한 실정에 있다. 이에 本 研究에서는 보다 현상에 가까운 오류예측모델을 제시하고 이를 기초로 테스트의 정지시점을 最適化하려는 것이다.

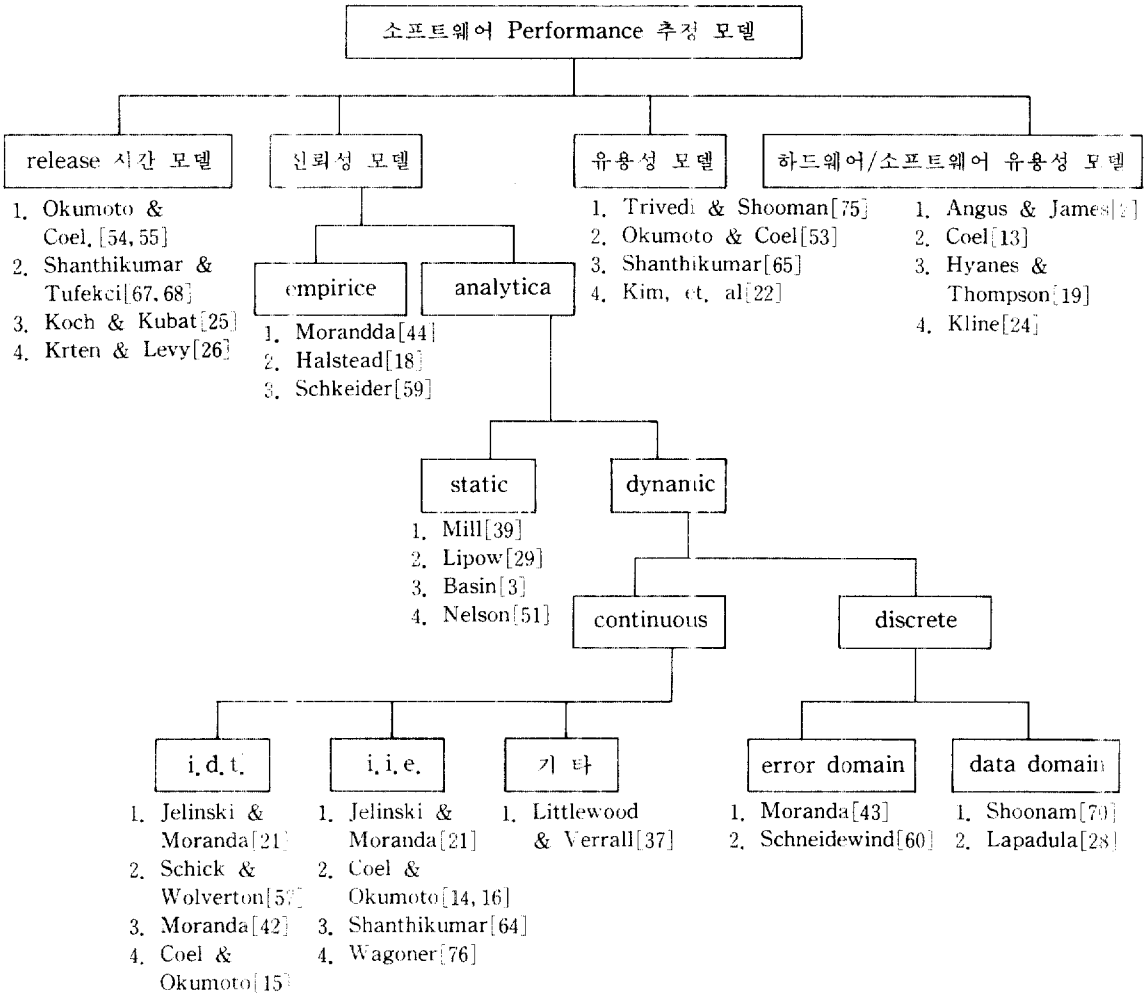
2. 研究 動向

컴퓨터의 소프트웨어의 중요성이 인식되기 시작한 1970년대 이후 많은 소프트웨어의 信賴性에 관한 研究論文이 발표되었다. 論文의 대부분은 MTBF(Mean Time Between Failure) 또는 소프트웨어 속에 포함되어 있는 오류의 갯수를 推定하는 수학적 모델을 주제로 하고 있다. 이러한 소프트웨어의 信賴性과 有用性의 理論에는 취약점이 있게 마련이다.

지금까지 發表된 대표적인 研究結果를 要約해서 살펴 보도록 한다.

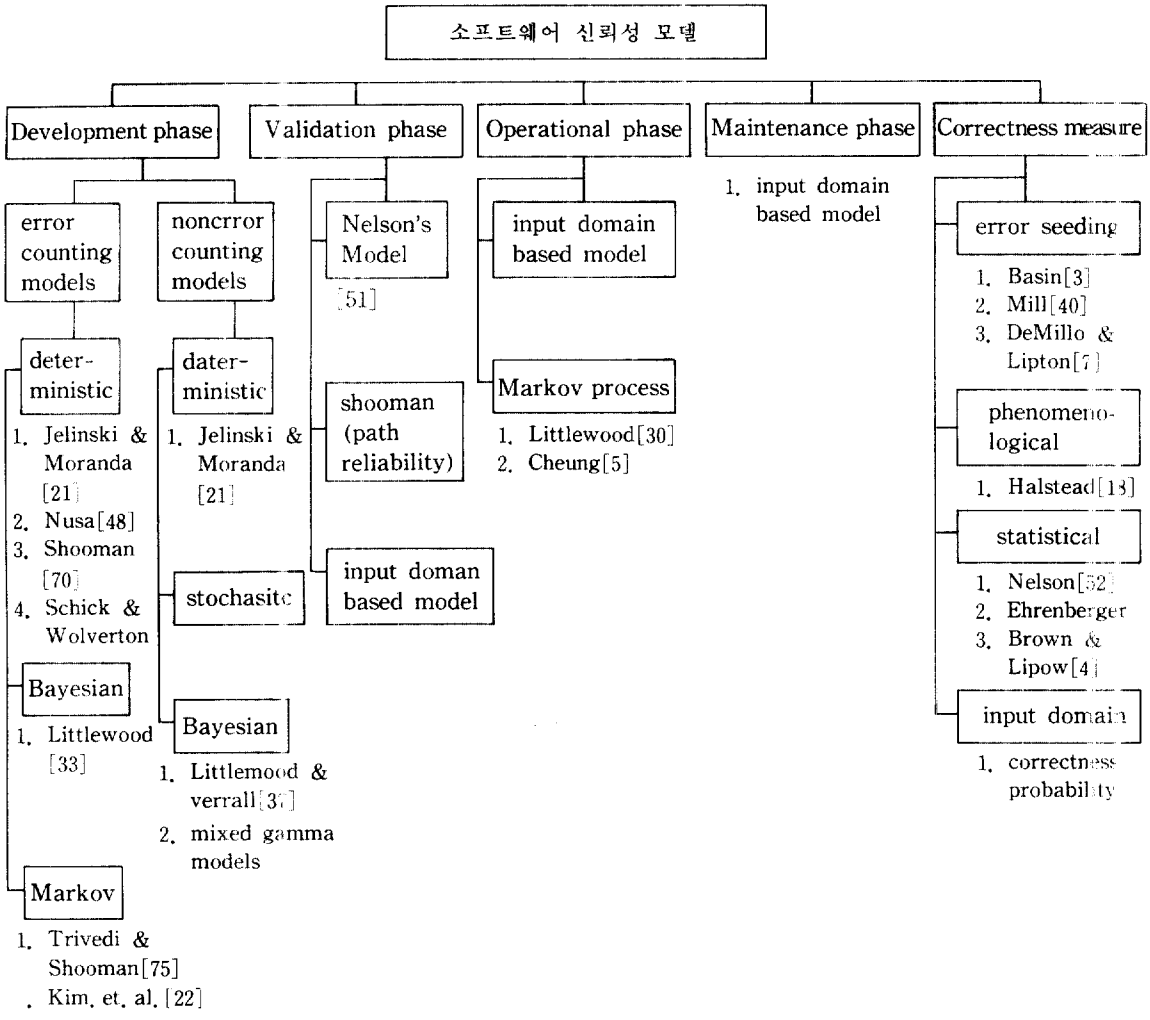
Shooman[9]은 소프트웨어의 信賴性은 예정된 기간동안 프로그램이 제시된 規格에 따라 성공적으로 그 기능을 다할 確率로 정의하였다. 반대로 합리적인 요구대로 기능을 수행하지 못할 때, 소프트웨어가 故障이 발생되었다고 하고 그 原因을 소프트웨어의 오류라고 한다. 오류가 컴퓨터에 미치는 영향에 따라 중오류(major error)와 경오류(minor error)로 分類하고 있다. Schneidewind[6]는 소프트웨어의 오류를 설계상의 오류, 프로그램작성시의 오류, 문서취급상의 오류, 검정과정에서 발생하는 오류등 오류발생의 시기적 구분을 시도하고 있다. Musa[3]는 소프트웨어의 信賴性을 예상되는 모든 상황하에서 기능을 적절히 수행할 수 있는 설계관계에서의 소프트웨어의 신뢰의 척도라 하였다. 소프트웨어의 정의는 대등소이 하지만 중요한 것은 規格作成및 設計段階에서 信賴性에 관한 개념이 확립되어야 비로소 소프트웨어의 完成段階에서 統合 및 檢定을 통한 소프트웨어의 信賴性豫測데이터를 수집할 수 있다는 것이다. 소프트웨어의 信賴性에 관한 研究結果는 Jelinski-Moranda[2]를 필두로 200여편에 달하고 있다. 이를 Shanthikumer[8]는 <그림1>과 같이 分類하고 있다. Ramamoorthy와 Bastani[4]는 소프트웨어의 라이프 사이클(life cycle)을 4段階로 分類하고 소프트웨어의 모델이 어느 단계에 쓰이는가 모델에서 사용하는 完全性의 尺度가 무엇인가에 따라 研究結果를 분류한 것이 <그림

2>와 같다. 本 硏究와 가장 밀접한 연구는 Black Box 모델로서 이를 중점적으로 검토하기로 한다. 이 모델은 소프트웨어를 블랙박스로 간주하여 크기가 비슷하고, 사용목적이 같은 소프트웨어에 포함되어 있는 오류의 수는 대략 일정하다고 보고, 故障率도 시스템에 남아 있는 오류의 수에 비례한다고 가정한다. 이러한 모델은 Jelinski와 Moranda[2]에 의해 처음으로 제시되었으며, 그 후 Schick-Wolverton[5]은 故障率이 시스템에 남아있는 오류의 수에 비례하고, 운영시간에 따라 증가하는 모델을 발표하였다. 그 대표적인 것을 보면 shooman[9], Musa[3], Goelokumoto[1], Shanthikmar[7] 등의 모델이 있다.



* i.d.t. : independently distributed inter-failure time
i.i.e. : independent and identical error liehavior.

<그림 1> Shanthikumar에 의한 신뢰성 모델의 분류



<그림 2> Ramamoorthy에 의한 신뢰성 모델의 분류

II. 殘存誤類豫測模型

1. 記號의 定義

본 論文에서 사용하는 定義는 다음과 같다.

N_0 : 檢定段階의 초기에 시스템에 존재하는 총 오류의 수.

$e(t)$: 檢定時間 t時間까지 발견된 오류의 수.

$n(t)=N_0-e(t)$: 檢定時間 t時間에 시스템에 남아 있는 오류의 수.

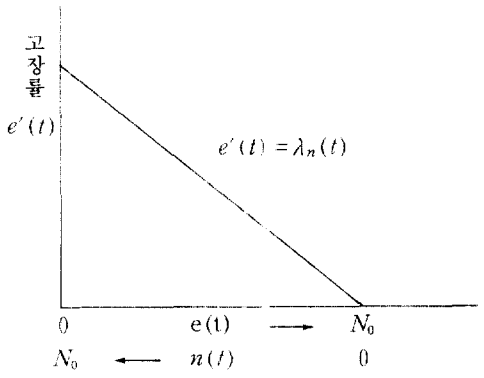
T_i : i번째 오류가 발견된 시각

$e'(t)$: t時間에서의 오류의 발견률 즉, 시스템의 故障率

C_a : 單位 檢定 時間當의 費用

C_f : 오류가 있는 狀態로 출고될 경우 오류당 실패비용

2. 故障率의 數學的 模型



<그림 3> 고장률과 잔존오류수

信賴性 豫測模型의 대표적인 것은 <그림 3>에서 보는 바와 같이 故障率은 시스템에 남아있는 오류의 수에 비례한다고 하는 것이다. 이를 數學的으로 표현하면 다음과 같다.

$$e'(t)=\lambda\cdot(N_0-e(t)),\ \lambda\text{는 비례상수} \quad \cdots\cdots(1)$$

模型의 基本 假定은 다음과 같다.

(1) 故障은 오류로 부터 연유된다.
(2) 고장은 오류와 1대1로 대응된다.
(3) 오류는 발견과 동시에 제거된다.
(4) 오류는 시스템의 故障을 통해서만이 발견된다.
(5) 초기의 오류의 발견수는 0으로 한다. 즉, $e(0)=0$
(6) 검정의 균일성

假定(6)에서 檢定時間의 시간축을 어떻게 잡느냐에 따라서 檢定의 균일성에 差異가 나타나게 된다. 이에 대하여 3個의 基本模型의 타당성에 관한 檢定은 Trachtenberg에 의하여 <그림 4>와 같이 이루어진다(컴퓨터 사용시간모델(Musa Model), 인력시간모델(Shooman Model), Calendar Time모델(Jelinski-Moranda Model)).

線型模型의 치명적 결함의 하나로 지적되는 것은 오류의 동일한 發見確率을 갖고 있다는 가정일 것이다. 오류는 동일한 발견을 갖고 있지 않다는 것은 직관적이나 간단한 경험으로 알 수 있겠다. 이에 착안해서 본 연구에서는 오류 發見率이 時間에 따라 減少하는 다음과 같은 數學的 模型을 가정하게 된것이다.

$$e'(t)=\frac{\lambda\cdot\{N_0-e(t)\}}{(t+a)} \quad \cdots\cdots(2)$$

여기서 a는 <그림5>에서와 같은 위치모수이다. a가 0이 되면 $t=0$에서 $e'(t)=\infty$가 되어 a의 값으로 조정해야한다.

$e(t)$를 구하기 위해서 식(2)를 變數分離型으로 변환시키면 다음 식을 얻게 된다.

$$\frac{d\cdot e(t)}{\{N_0-e(t)\}}=\frac{\lambda\cdot dt}{(t+a)} \quad \cdots\cdots(3)$$

양변을 적분하면, 각각 다음 식을 얻게 된다.

$$\int_0^t\frac{1}{\{N_0-e(t)\}}d\cdot e(t)$$

$$=\ln\left(\frac{N_0}{\{N_0-e(t)\}}\right)$$

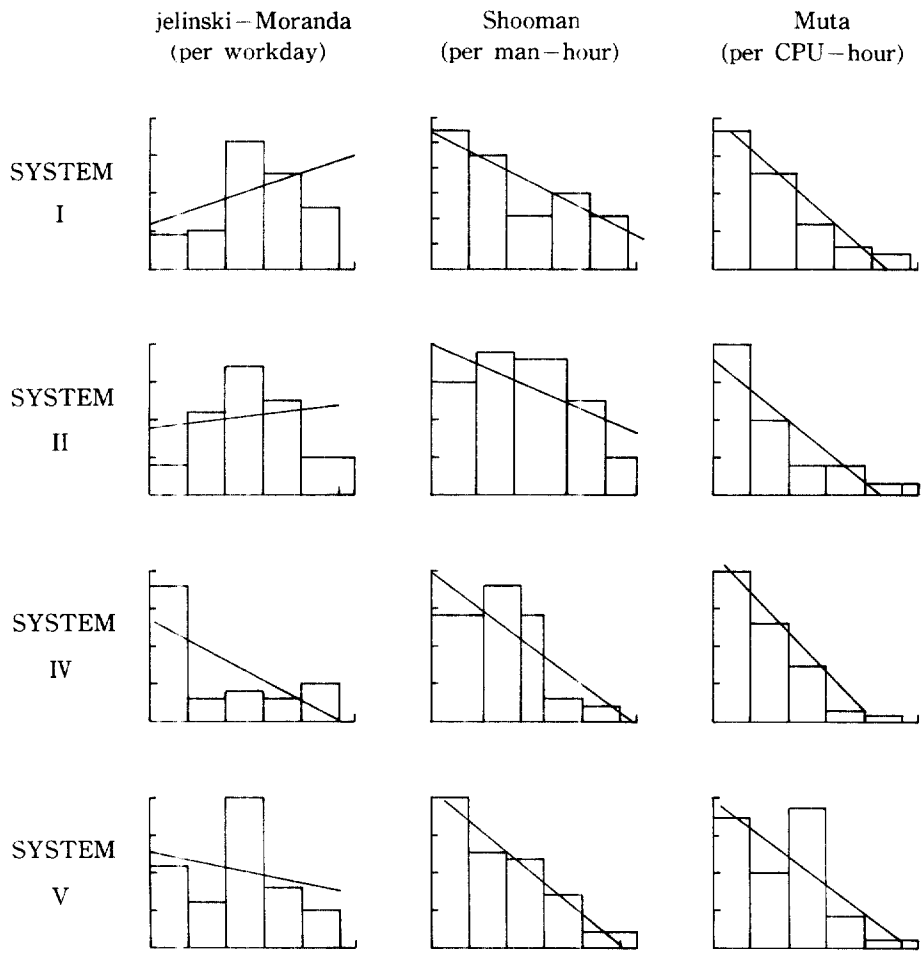
$$\int_0^t\frac{\lambda}{(t+a)}dt=\ln\left(\frac{t+a}{a}\right)^{\lambda} \quad \cdots\cdots(4)$$

결과적으로 다음 식을 얻게 된다.

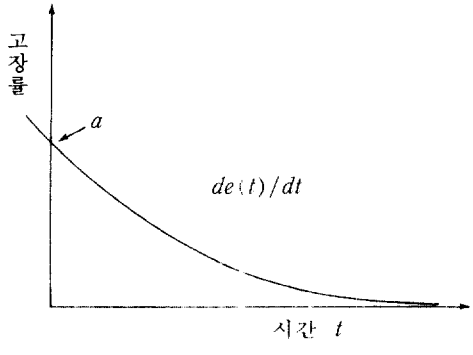
$$\frac{N_0}{\{N_0-e(t)\}}=\left(\frac{t+a}{a}\right)^{\lambda} \quad \cdots\cdots(5)$$

요약하면 다음의 식이 된다.

$$e(t)=N_0\left[1-\left(\frac{a}{t+a}\right)^{\lambda}\right] \quad \cdots\cdots(6)$$



<그림 4> 선형고장률모델의 상관계수관계



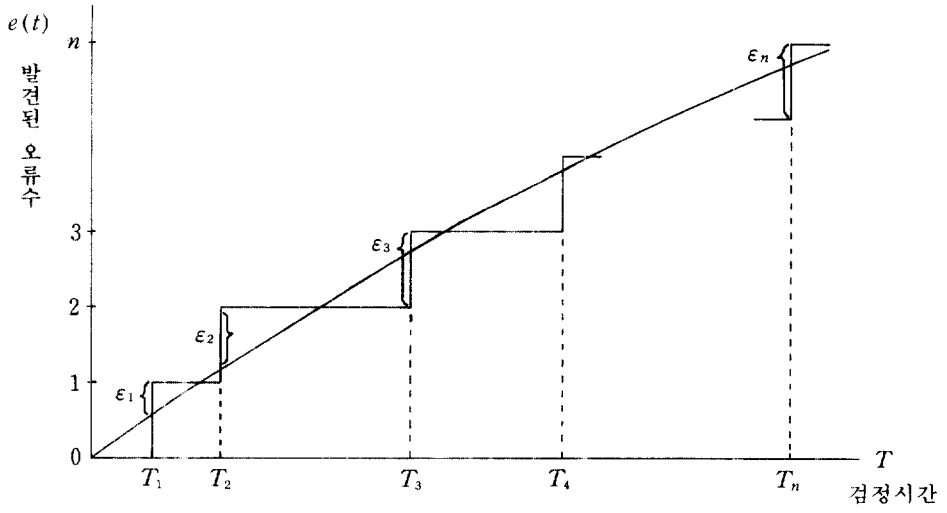
<그림 5> 오류발견률(시스템 고장률)의 시간적 변화

식(6)은 오류의 발견률이 본 논문에서 제시한 바와 같이 식(2)의 模型을 따를때의 시점 t에서의 오류의 발견수가 된다. 여기서 N_0, λ, a에 관한 것을 사전에 알고 있지 못하며, 지금까지의 테스트 결과를 가지고서 N_0, λ, a를 推定하는 문제가 제기되는 것이다.

3. 최소자승법에 의한 推定(Least Square Estimation)

어느 시점 t까지 발견된 총오류의 수가 n이라

할 때 각각의 오류가 발견된 시점 즉, T_1, T_2, T_3, …, T_n를 入力으로 해서 최초의 오류수 N_0, 비례상수 λ, 위치상수 a를 推定하는 問題로 귀결되게 될 것이다. 식(6)을 만족하는 모수초기오류수 N_0, 비례상수 λ, 위치상수 a를 推定하는 方法은 여러가지가 있겠으나, 本 論文에서는 가장 강력한 것으로 알려진 最小自乘法을 이용하기로 한다. 最小自乘法이란 <그림6>에서 보는 바와 같다. 식(6)을 만족하면서 誤差가 最小가 되도록 母數値를 推定하는 것이다.



<그림 6> 최소자승법의 개요

여기서 誤差는 실제 발견된 오류의 수와 식(6)에 의해서 예측된 오류의 수의 差에 의해서 決定되는 것이다.

즉, $\varepsilon_i = i - e(T_i)$ ……(7)

誤差의 평방화(sum of square) Q는

$$Q=\sum_{i=1}^{n}\varepsilon_i^2=\sum_{i=1}^{n}\{i-e(T_i)\}^2$$

$$=\sum_{i=1}^{n}\left[i-N_0\left\{1-\left(\frac{a}{T_i+a}\right)^{\lambda}\right\}\right]^2 \cdots\cdots(8)$$

가 된다. Q의 최소값을 구하기 위해서는 식(8)를 N_0, λ, a에 대해서 각각 편미분하면 된다. 즉, Q가 최소가 되기 위한 必要條件은 다음과 같다.

$$\frac{\partial Q}{\partial N_0}=0, \quad \frac{\partial Q}{\partial \lambda}=0, \quad \frac{\partial Q}{\partial a}=0$$

$\frac{\partial Q}{\partial N_0}=0$의 식으로 부터 다음 식을 얻고, 이를 N_0의 양함수로 정리한다.

$$-2\sum_{i=1}^{n}\left[i-\hat{N}_0\left\{1-\left(\frac{\hat{a}}{T_i+\hat{a}}\right)^{\lambda}\right\}\right]$$

$$\left(\frac{\hat{a}}{T_i+\hat{a}}\right)^{\lambda}=0 \cdots\cdots(9)$$

$$\hat{N}_0=\frac{\sum_{i=1}^{n} i\cdot\left(\frac{\hat{a}}{T_i+\hat{a}}\right)^{\lambda}}{\sum_{i=1}^{n}\left\{1-\left(\frac{\hat{a}}{T_i+\hat{a}}\right)^{\lambda}\right\}\left(\frac{\hat{a}}{T_i+\hat{a}}\right)^{\lambda}}$$

……(10)

$\frac{\partial Q}{\partial \lambda}=0$의 식으로 부터 다음 식을 얻고, 이를 N_0의 양함수로 전개한다.

$$-2\sum_{i=1}^{n}\left[i-\hat{N}_0\left\{1-\left(\frac{\hat{a}}{T_i+\hat{a}}\right)^{\lambda}\right\}\right]$$

$$\cdot\left(\frac{\hat{a}}{T_i+\hat{a}}\right)^{\lambda}\cdot N_0$$

$$\cdot\ln\left(\frac{\hat{a}}{T_i+\hat{a}}\right)=0 \quad \cdots\cdots(11)$$

$$\hat{N}_0=$$

$$\frac{\sum_{i=1}^{n} i\cdot\left(\frac{\hat{a}}{T_i+\hat{a}}\right)^{\lambda}\cdot\ln\left(\frac{\hat{a}}{T_i+\hat{a}}\right)}{\sum_{i=1}^{n}\left\{1-\left(\frac{\hat{a}}{T_i+\hat{a}}\right)^{\lambda}\right\}\cdot\left(\frac{\hat{a}}{T_i+\hat{a}}\right)^{\lambda}\cdot\ln\left(\frac{\hat{a}}{T_i+\hat{a}}\right)} \quad \cdots\cdots(12)$$

$\frac{\partial Q}{\partial a}=0$의 식으로 부터 다음 식을 얻고, N_0의 양함수로 전개한다.

$$-2\sum_{i=1}^{n}\left[i-\hat{N}_0\left\{1-\left(\frac{\hat{a}}{T_i+\hat{a}}\right)^{\lambda}\right\}\right]\cdot-\hat{N}_0$$

$$\cdot\hat{\lambda}\cdot\left(\frac{\hat{a}}{T_i+\hat{a}}\right)^{\lambda-1}\cdot\left(\frac{-T_i}{T_i+\hat{a}}\right)=0$$

$$\cdots\cdots(13)$$

$$\hat{N}_0=-\frac{\sum_{i=1}^{n} i\cdot\left(\frac{T_i}{T_i+\hat{a}}\right)^{\lambda}}{\sum_{i=1}^{n}\left\{1-\left(\frac{\hat{a}}{T_i+\hat{a}}\right)^{\lambda}\right\}\cdot\left(\frac{T_i}{(T_i+\hat{a})^{\lambda}}\right)} \quad \cdots\cdots(14)$$

結論的으로 (10), (12), (14)식을 만족하는 N_0, λ, a를 구하면 이것이 推定値가 되는 것이다. 여기서 N_0, λ, a를 구하는 方法은 식(10)과 식(12)로 부터 N_0를 소거하고, 그 다음 식(10)과 식(14)로 부터 다시 N_0를 소거하여 2원 방정식을 얻어서 간단한 數値解法에 의해서 구하면 된다.

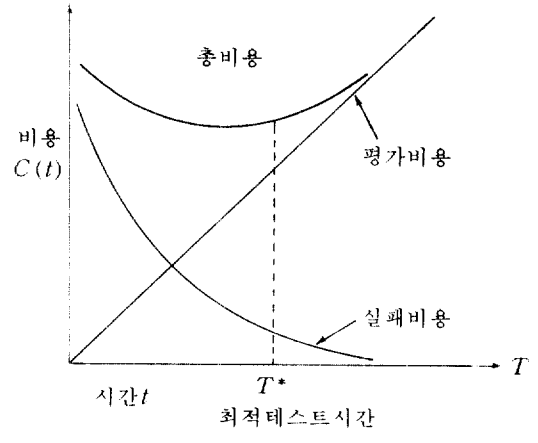
III. 테스트 時間의 最適化

1. 費用模型

오류를 발견하고 소프트웨어를 評價하는데는 하드웨어의 費用을 포함해서 評價作業에 종사하는 人件費등 많은 費用을 수반하게 된다. 소프트웨어를 테스트 없이 출고할 경우, 오류에 의한 費用이 발생하게 될 것이다. 즉 소프트웨어의 평가비용(A-코스트)과 실패비용(F-코스트)이 발생하게되며, 이러한 비용의 적절한 배분문제가 제기되게 될 것이다. 이는 <그림7>과 같이 될 것이다. 총비용(CT) 즉, 評價費用과 失敗費用을 最小化하는 테스트 時間이 존재하게 될 것이다. 본 논문에서는 費用에 관한 두가지의 基本假定을 하고 있다.

(1) 評價費用은 테스트 時間에 비례해서(비례상수 : C_t) 지출된다.

(2) 失敗費用은 잔류하는 오류의 수에 비례해서 지출된다.



<그림 7> 테스트 시간과 비용관계

가정(1)에 의해서 테스트 시간을 T라 할때 評價費用은 다음 식과 같다.

$$C_a(t) = C_t \cdot T \quad \cdots\cdots(15)$$

가정(2)에 의해서 失敗費用은 테스트 시간을 T라 할때 그 시간에 남아있는 오류수에 비례하므로 다음 식과 같다.

$$C_f(t) = C_f\{N_0 - e(T)\} \quad \cdots\cdots(16)$$

여기서 식(6)으로 부터 $e(T)$를 구해서, 대입하면 다음과 같은 결과식을 얻는다.

$$C_f(t) = C_f \cdot N_0\left(\frac{a}{T+a}\right)^{\lambda} \quad \cdots\cdots(17)$$

식(15), 식(16)으로 부터 총비용($C(T)$)을 구하면 다음 식과 같이 된다.

$$C(T) = C_t \cdot T + C_f \cdot N_0\left(\frac{a}{T+a}\right)^{\lambda}$$

$$\cdots\cdots(18)$$

2. 費用의 最小化

식(18)의 최소값을 구하기 위해서 T로 미분하고 0으로 놓으면 다음 식을 얻는다.

$$\frac{dc(T)}{dt} = C_t - \frac{C_f \cdot N_0 \cdot \lambda \cdot a^{\lambda}}{(T+a)^{\lambda+1}} = 0$$

$$\cdots\cdots(19)$$

이를 정리하고 T^*을 구하면 다음 식이 된다.

$$T^* = \left(\frac{C_f \cdot N_0 \cdot \lambda \cdot a^{\lambda}}{C_t}\right)^{\frac{1}{\lambda+1}} - a \quad \cdots(20)$$

식(6)을 만족하는 T^*가 식(18)의 최소값이 되는 것은 자명하다. 왜냐하면 식(15)와 식(17)을 자세히 관찰하면 두식이 모두 Convexity를 만족함을 쉽게 알 수 있고, 식(18)의 Convexity도 증명이 됨으로 1차 미분식(19)를 만족하는 T^*는 자동적으로 식(18)식의 最小値을 만족한다.

IV. 數値事例研究

실제 大型 시스템의 소프트웨어의 테스트 事例를 들어야 하겠으나 실제 데이터가 없어 現實性 있는 임의 데이터를 가정하여, 본 모델을 활용하는 方法을 소개한다.

<數値事例>

어떤 소프트웨어를 100단위 시간동안 테스트한 결과 다음과 같이 8개의 오류발견 시점에 관한 데이터를 얻었다. 초기오류수동을 예측하고, 최적 테스트종료시간을 결정하라. 단 소프트웨어의 평가 단위 시간당 費用(C_t)과 실패비용(C_f)의 비는 1000으로 가정한다.

오류발견순위	1	2	3	4	5	6	7	8
오류발견시각(T_i)	5	12	18	25	34	43	54	66

<數値解>

주어진 T_i資料를 식(10), 식(12), 식(14)에 넣고 풀면 $N_0 = 100.2$, $\lambda = 0.101$, $\hat{a} = 51.2$를 얻는다. $\hat{N}_0$는 정수이어야 하므로 $N_0 = 100$으로 놓으면 $\hat{\lambda} 0.102$, $\hat{a} = 51.1$을 얻는다. 이를 식(20)에 대입하면 $T^* = 6196.3$ 단위시간 즉, 6196.3 단위시간 동안 테스트함이 적절하다.

V. 結 論

本 研究 論文에서는 기존 研究分野에서 취약점의 하나로 등장한 모든 오류의 發見確率은 동

일하다는 가정을 현실에 맞도록 정정하여 새로운 故障模型을 제시하였다. 즉, 오류의 發見確率은 처음 오류 보다는 나중에 까지 남아있는 오류가 시간에 반비례해서 발견하기 어렵다는 가정하에 故障率 模型을 도출하였다. 이것은 직관적으로는 그 타당성이 쉽게 理解가 되나 資料의 不足으로 실증적 연구가 부족한 실정에서 연구가 종료됨으로서 아쉬움이 남는다. 이것은 차후에 실증적 자료를 보강해서 연구되어야 할 과제이다. 본 논문에서 제시된 最適 테스트 完了時間에 관한 간략식은 소프트웨어 開發 및 評價實務 現場에서 유용하게 사용될 수 있을 것으로 기대된다.

參 考 文 獻

1. Goel, A.L. and Okumoto, K. (1979), *"A Markovian Model for Reliability and other Performance Measure"*, Proce, National Computer Conference, pp. 769-774
2. Jelinski, Z. and Moranda, P.B. (1972), *"Software Reliability Research"*, Statistical Computer Performance Evaluation, W. Freberger(ed.), pp. 465-484
3. Musa, J.D. (1975), *"A Theory of Software Reliability and Its Application"*, IIIE Transaction on Software Engineering, Vol. SE-1, pp. 312-332
4. Ramanoorthy, C.V. and Bastani, F.B. (1982), *"Software Reliability Status and Prospectives"*, IIIE Transactions on Software Engineering, Vol. SE-8, No. 4, pp. 354-371
5. Schick, G.J. and Wolverton, R.W. (1973), *"Assesment of Software Reliability"*, proc. Operations Research, physica-Verlag, Wurzberg-Wien, pp. 395-422
6. Schneide wind, N.F. and Hoffman, H. (1979), *"An Experiment in Software Error Data Collection and Analysis"*, IIIE Transactions on Software Engineering, Vol. SE-5, No. 3, May
7. Schnthikumar, J.G. (1980), *"Software Performance Prediction Using State-Dependent Error Occurrence-Rate Model"*, proc. 19th Annual Technical System Integrity, pp. 67-73
8. Shanthikumar, J.G. (1983), *"Software Reliability Model : A Review"*, Microelectronics and Reliability.
9. Shooman, M.L. (1972), *"Probabilistic Models for Software Reliability Prediction"*, statistical Computer Performance Evaluation, W. Freiberger, Rd., New York : Academic, pp. 557-582