〈研究論文〉

Journal of the Korean Institute of Industrial Engineers
Vol. 13, No. 1, June. 1987

Software 신뢰성의 연구동향 및 전망 †

Software Reliability : A State-of-the-Art Survey

김 영 휘*

Abstracts

This paper dsals with current and future research directions in the modelling of computer software reliability. The paper reviews some of the major software reliability models published during the past two decades and examines problems associated with these models.

I. 서 론

컴퓨터 software의 중요성이 새로이 인식되어감에 따라 1970년대 이후 많은 software 신뢰성에 관한 연구논문이 발표되었다. 이러한 논문은 대부분이 software의 신뢰성, 유용성과 이와 관련된 software의 특성이라 할 수 있는 MTTF (Mean Time To Failure), 또는 software에 포함되어 있는 오류의 갯수등에 관한 추정 및 예측에 관한 수학적 모델을 주제로 하고 있는 것으로 software 신뢰성에 관한 제반문제들을 해소시키는 데에는 미흡한 상태이다. 부분적이기는 하지만 기존모델을 적용한 결과 [1, 8, 37]는 만족스러운 것이 되지 못하고 있다.

기존 software 신뢰성에 관한 이론에는 다음과 같은 취약점[1, 37, 52]이 있음이 지적되고 있다. 즉, software에는 hardware에서와 같은 磨耗現象에 따른 고장이나 우연요인에 따른 고장이 일어나지 않으므로 이 경우에는 software가 주어진 요구조건을 만족하는가를 입증하는 것이 바람직하지만 현재의 프로그램 입증기술(verification technique)은 real-time applic-

† 이 연구는 고려대학교 교내 연구비 지원아래 이루어진 것임.

* 고려대학교 공과대학 산업공학과

ation에서 이용되는 software와 같이 복잡하고 그 규모가 방대한 software를 다루기에는 미흡한 상태에 있다. 또 software를 검정(testing)하는 경우에 있어서도 가능한 모든 입력에 대해서 software를 검정한다는 것은 대개의 경우 입력의 가지수가 방대하기 때문에 실제로는 불가능하다. 요컨데 software에 오류가 있음을 입증할 수는 있어도 오류가 없음을 입증한다는 것은 현재의 기술로는 불가능한 일이다.

기존 software의 신뢰성 모델은 software의 修正段階(debugging phase)나 인증단계(validation phase)에서 발생한 오류를 근거로 software의 신뢰성이나 또는 이와 관련된 特性値의 추정을 시도한 것이 대부분이다. 아래에서는 지금까지 발표된 대표적인 연구결과를 토대로 software 신뢰성에 관한 연구동향 및 전망을 살펴보기로 한다.

Ⅱ. Software 신뢰성의 개념 및 정의

software 신뢰성의 개념을 이해하기 위해서는 software 오류와 software 고장(failure) 에 관한 정의부터 알아 둘 필요가 있다. 현재로서 software 오류에 대한 결정적인 정의가 내려진 것은 없지만 일반적으로 software가 합리적인 요구대로 기능을 수행하지 못할 때 software고장이 발생했다고 하며 그 원인을 software오류라고 한다.

Shooman[70]은 오류가 컴퓨터 시스템에 미치는 영향의 경중에 따라 중오류(major error)와 경오류(minor error)로 분류하고 있으며 Schneidewind[60]는 software 오류를 설계상의 오류, 프로그램 작성시의 오류, 문서취급상의 오류, 수정오류, 검정과정에서 발생하는 오류 등으로 구분하고 있다.

시스템의 신뢰성이란 시스템이 예정된 기간동안 요구되는 운영조건하에서 성공적으로 그 기능을 다할 확률을 의미한다. Software 신뢰성의 정의 몇가지를 살펴보면 다음과 같다.

Shooman[69]은 software 신뢰성은 예정된 기간동안 프로그램이 제시된 규격(specification)에 따라 성공적으로 그 기능을 다할 확률로 정의하였다. 규격에는 hardware의 세부적인 성공의 정의 그리고 입력 및 출력 데이타등에 관한 사항이 포함된다.

Musa[48]는 software 신뢰성을 예상되는 모든 상황하에서 기능을 적절히 수행할 수 있는 설계단계에서의 software의 신뢰의 척도라 하였다. 또 Littlewood[34]는 예정된 기간동안 제시된 환경조건하에서 software가 고장없이 운영될 확률로 정의하였다.

이들 정의는 그 내용이 대동소이하지만 중요한 것은 규격작성 및 설계단계에서 software신뢰성에 관한 개념이 확립되어야 비로서 software의 완성단계에서 통합(integration) 및 검정을 통한 software 신뢰성 예측 데이타를 수집할 수 있다.

위에서 제시된 정의에서 보는 바와 같이 software 신뢰성이나 이에 관한 척도는 대개가 확률적 개념에 바탕을 둔것이 많다. Software 신뢰성을 나타내는 데 흔히 사용되는 유용성 A(t)나 신뢰성 R(t), MTTF, MTBF (Mean Time Between Failure), MTTR (Mean Time To Repair)등은 원래 hardware의 신뢰성을 나타내기 위해서 정의된 척도들이다. 이러한 척도들은 software 신뢰성문제를 다루는 데에 적절히 사용되는 경우도 있지만 또 경우에 따라서는 실제 일어나는 현상을 적절히 설명하는 데에는 적합치 않은 경우도 많다. 이와 같은 사실은 주로 software에서 일어나는 현상과 hardware신뢰성이론의 기본 가정사이에 존재하는 근본적인 차이와 기존이론을 software 문제에 적용하는 방법이 미숙한 데에 기인하는 것으로 해석된다.

Littlewood[32]는 hardware와 software 는 그 특성이 근본적으로 상이하므로 hardware를 대상으로 발전한 신뢰성이론을 software분야에 무분별하게 적용하는 데에는 문제가 있음을 지적하고 있다. 그는 특히 software를 평가하는 데에는 software의 "상태"를 나타내는 정적 특성이라 할 수 있는 "프로그램에 포함된 오류의

갯수" 보다는 동적특성을 나타내는 운영적 신뢰성(operational reliability)의 척도를 사용하는 것이 바람직하다고 주장하고 있다. 즉 어떠한 프로그램이냐 보다는 이 프로그램이 실행과정에서 어떤 동적특성을 나타내느냐가 보다 중요하다는 주장이다. 그는 또 hardware에서 널리 이용되는 MTBF와 같은 수치를 무분별하게 사용하는 것 보다는 다음 고장이 일어날 때 까지의 시간에 관한 확률분포를 이용하는 것이 보다 합리적이라는 주장을 하고 있다.

현재로서 software 신뢰성을 평가하는 데 항상 적용될 수 있는 만능적인 척도는 존재하지 않는다.

Ⅲ. Software신뢰성 모델의 분류

Jelinski-Moranda[21]의 연구결과를 필두로 지금까지 발표된 software 신뢰성에 관한 연구논문은 대략 200편에 달하는 것으로 추산된다. Shanthikumar[66]는 100여편의 연구논문을 연구의 방법 및 software 신뢰성의 척도에 따라 다음 그림 1과 같이 분류하였다.

또 Ramamoorthy와 Bastani[56]는 software의 life-cycle을 검정 및 수정단계(Testing and debugging phase), 인증단계(Validation phase), 운영단계(Operational phase) 및 유지단계(Maintenance phase)로 분류하고 software 모델이 어떤 단계의 문제를 대상으로 하고 있는가 또 모델에서 사용하는 correctness measure가 무엇인가를 기준으로 연구결과의 분류를 시도하였다. Ramamoorthy와 Bastani의 분류와 이를대표하는 연구결과를 도시한 것이 그림 2이다.

또 다른 관점[23]에서 보면 software 신뢰성 모델은 크게 경험적 모델(empirical model) 과 해석적 모델(analytic model)로 나뉘어 진다. 경험적 모델은 software 고장에 관한 과거의 경험적 자료를 이용하여 complexity 척도와 같은 software의 정량적 특성과 신뢰성간의 관계를 다룬 것으로, Moranda[44], Halstead[18], Schneider[59]등의 모델이 이에 해당된다. 해석적 모델은 software의 검정(test) 과정과 관련된 일련의 가정에 입각하여 수립되며, software시스템을 보는 관점과 신뢰성의 척도에 따라서 black-box 모델과 시스템의 실행과정을 Markov과정으로 가정하는 Markov모델의 두 가지 형태로 구분할 수 있다.

여기서는 기존의 해석적인 software 신뢰성 모델을 이 두가지로 구분하고 이들과 함께 software의 유용성 추정을 위한 유용성 모델과 최적 공개시기 모델들을 Shanthikumar와 Ramamoorthy 등의 분류를 참고하여 개괄적으로 살펴보고자 한다.

Ⅳ. Software 신뢰성 모델

아래에서는 software 시스템의 신뢰성 평가를 위한 기존의 해석적인 모델들을 black-box모델과 Markov 모델로 구분하여 알아보고자 한다. 앞으로 자주 사용되는 기호들에 관한 정의는 다음과 같다.

- N : 검정단계가 시작될 때 software 시스템에 있는 오류의 수
- T : 검정이 끝난 후 신뢰성을 추정하려는 시점
- X_i : $(i-1)$번째 고장과 i번째 고장간의 시간
- m : 관측된 software 고장수
- m_i : i번째 기간동안 제거된 오류의 수 (고장 즉시 오류가 수정될 때 $m_i=1$)
- n_i : 처음부터 i기간까지 제거된 오류의 누적수. 즉 $n_i=\sum_{j=1}^{i} m_j$
- n : 시간$(0, t]$동안 제거된 총오류의 수 $(n-n_m)$
- $Z_i(\cdot)$: $(i-1)$번째 고장과 i번째 고장사이에서의 harzard 함수
- $\lambda(\cdot)$: 특정 오류에 의해 발생되는 고장의 harzard 함수
- N_T : 검정이 끝난 후 software에 남아있는 오류의 수

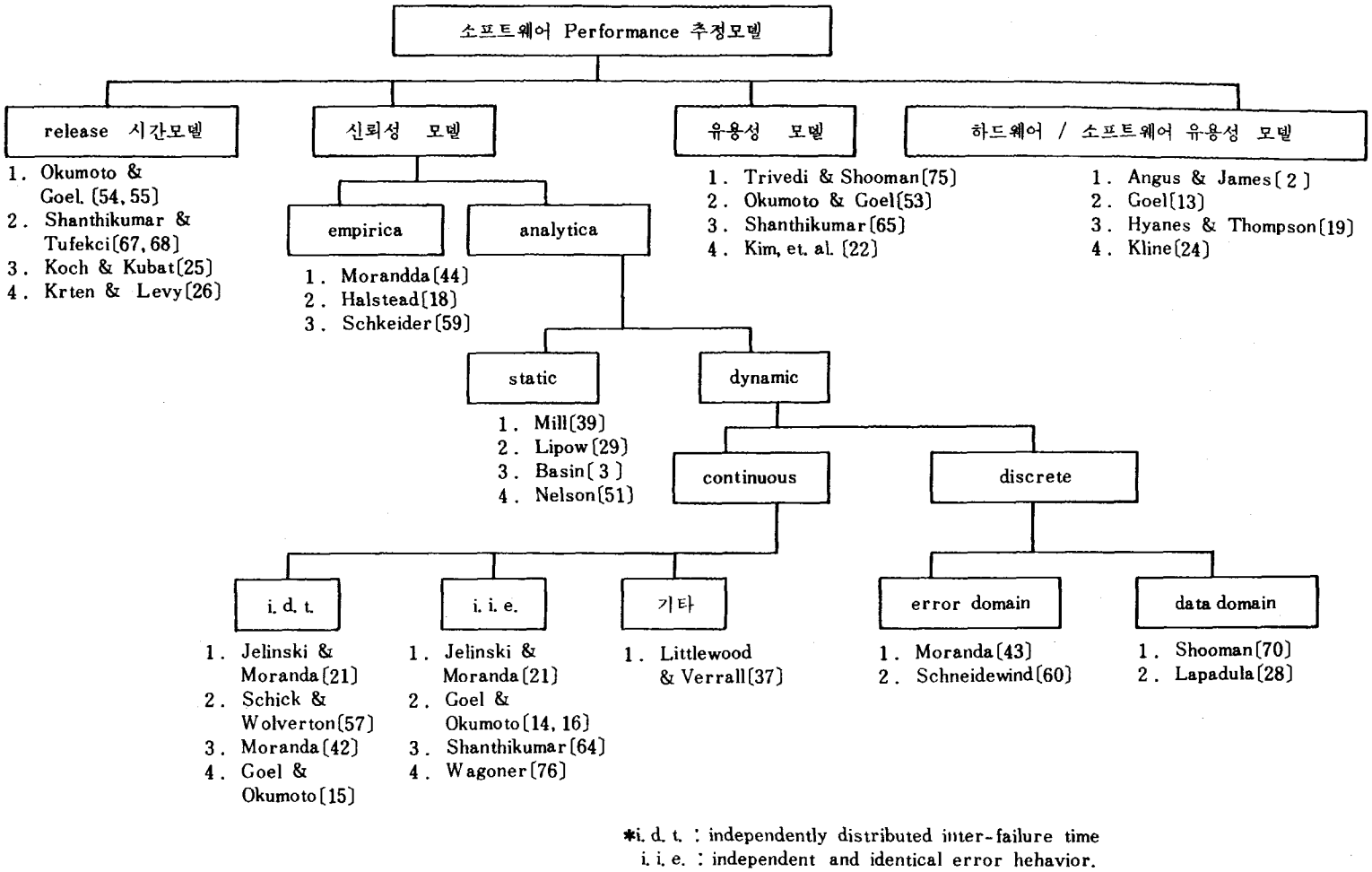

그림 1. Shanthikumar에 의한 신뢰성 모델의 분류

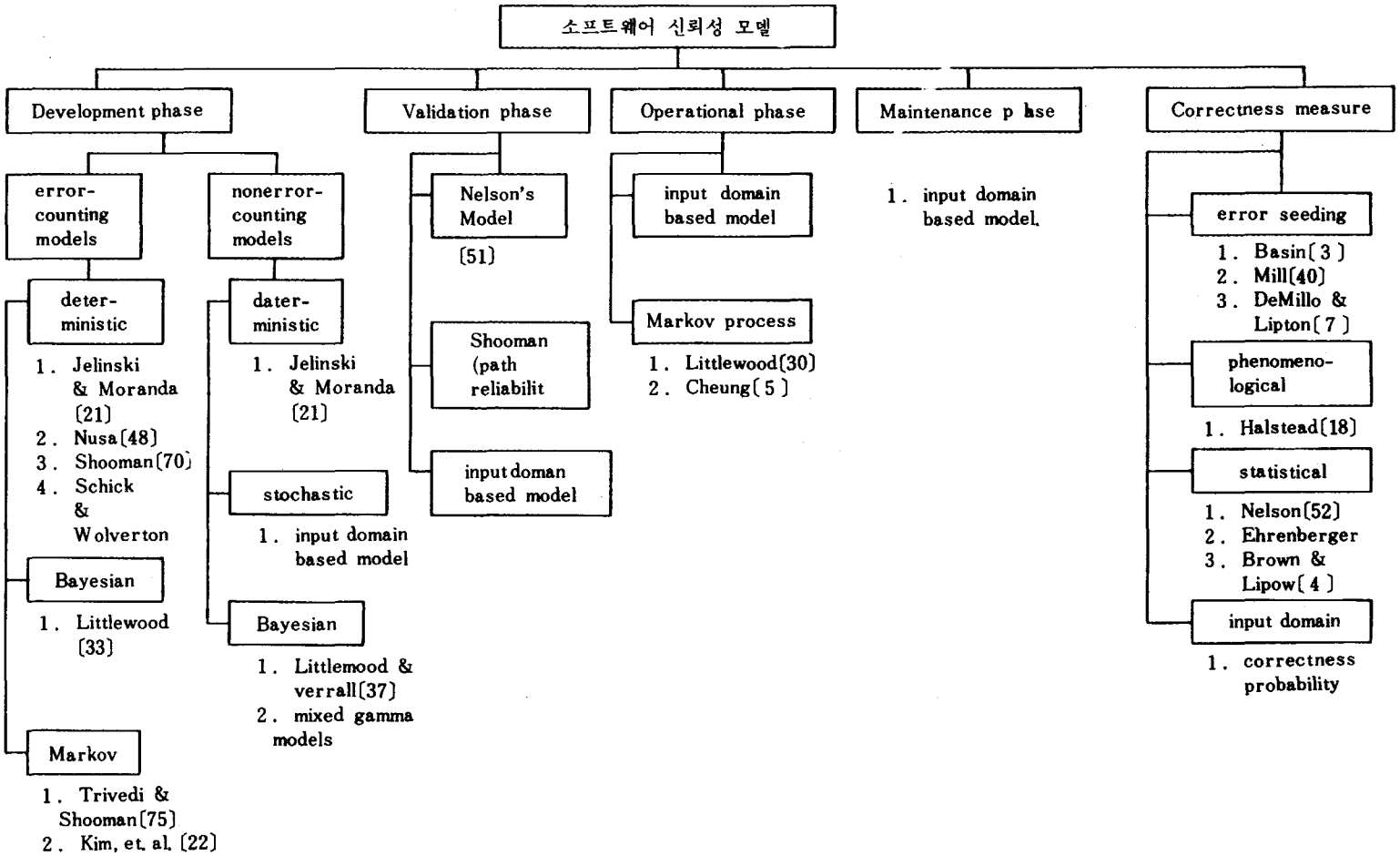

그림 2. Ramamoorthy에 의한 신뢰성 모델의 분류

$MTTF_T$: 검정이 끝난 시점에서 다음 고장이 발생할 때 까지의 평균시간

$R_T(\cdot)$: software의 신뢰성

T_P : 시점T 이후에 software의 오류를 완전히 제거하는데 소요되는 시간

1. Black-Box모델

이 모델은 software 시스템을 Black-box로 간주하여 크기가 비슷하고 사용목적이 같은 software에 포함되어 있는 오류의 수는 대략 일정하다고 보고, software 시스템의 고장율은 시스템에 남아있는 오류의 수에 비례한다고 가정한다.

이러한 모델은 Jelinski와 Moranda[21]에 의해 처음으로 제시되었으며, 그 후 Schick과 Wolverton[57]은 고장율이 시스템에 남아있는 오류의 수에 비례하고 운영시간에 따라 증가하는 모델을 발표하였다. 이들 외에 대표적인 black-box 모델에는 Shooman[70], Musa[48], Wagoner[76], Lipow[29], Goel과 Okumoto[15], Shanthikumar[61]등의 모델이 있다.

(1) Jelinski와 Moranda 모델

Jelinski와 Moranda는 Software 시스템의 (i−1)번째 고장과 i번째 고장사이의 고장율이 시스템에 남아있는 오류의 수 $(N-n_{i-1})$에 비례한다고 가정하여 harzard 함수 $Z_i(t)$를

$$Z_i(t)=(N-n_{i-1})\phi,\quad i=1,2,\cdots,m \qquad (4.1)$$

ϕ는 비례상수

라 하고, 고장과 고장사이의 간격이 이를 모수(parameter)로 하는 지수분포(exponential distribution)를 따른다고 가정하였다. 모델의 입력자료는 Xi, m 및 m_i이며, 모델의 모수N과 ϕ는 다음 방정식의 해로서 추정된다.

$$\sum_{i=1}^{m}\frac{1}{N-n_{i-1}}-\sum_{i=1}^{m}\phi\cdot x_i=0 \qquad (4.2)$$

$$\frac{n}{\phi}-\sum_{i=1}^{m}(N-n_{i-1})\cdot x_i=0 \qquad (4.3)$$

이 모델에서의 신뢰성 척도는 다음과 같이 구하여진다.

① $N_T=N-n$

② $MTTF_T=\dfrac{1}{(N-n)\cdot\phi}$

③ $R_T(t)=\exp\{-(N-n)\phi t\}\cdot t\geq 0$

④ $E[T_P]=\sum_{i=1}^{N-n}\dfrac{1}{i\phi}$

$$Var(T_P)=\sum_{i=1}^{N-n}\frac{1}{(i\phi)^2} \qquad (4.4)$$

단, 시점T 이후에 수정시마다 한개의 오류를 수정할 경우이다.

Jelinski와 Moranda의 모델은 Apollo 프로그램과 미해군의 software 시스템 고장자료에 대한 분석 가능성을 알아보기 위하여 적용되었다. Shooman[70]은 (4.1) 식에서

$N=I$

$\phi=k/I$

$$n_i=I\cdot\varepsilon_c(\tau)+1 \qquad (4.5)$$

단, k : 임의상수

I : instruction 총수

$\varepsilon_c(\tau)$: (시간 τ까지 수정된 총 오류의 수)/I

로 두고 software의 총 instruction 수를 근거로 한 모델을 제시하였다. 또한 Musa[48]의 실행시간모델(execution time model)에서는 (4.1) 식에서

$$\phi=k\cdot f \qquad (4.6)$$

단, k는 비례상수

f는 실행빈도(execution frequency)

로 두어 harzard 함수를 정의하였다. 그는 software의 실행시간이 software의 고장에 가장 큰 영향을 미친다고 보고, 실행시간이 τ일때

$$MTTF=\frac{1}{f\cdot k\cdot N}\exp(f\cdot k\cdot\tau) \qquad (4.7)$$

와 같은 신뢰성 척도를 구하였다.

또 Jelinski와 Moranda모델은 Lipow[29] 에 의하여 매 고장시에 1개 이상의 오류를 수정할 가능성을 고려한 모델로 확장되었다.

(2) Schick와 Wolverton 모델

Schick와 Wolverton은 software 시스템의 (i−1)번째 고장과 i번째 고장사이의 고장율이 시스템에 남아있는 오류의 수 $(N-n_{i-1})$에 비례

하고 운영시간 t에 따라 증가한다고 가정하였다. 즉, harzard 함수 $Z_i(t)$는

$Z_i(t) = (N - n_{i-1})\phi t,\ i = 1, 2, \cdots, m$ (4.8)

이며, 고장과 고장간의 시간간격은 Rayleigh 분포를 따른다. 모델의 입력자료는 Xi, m 및 m_i이며, 모델의 모수 N, ϕ는 최우추정법(maximum likelihood estimation)에 의하여 다음 두 방정식

$$\sum_{i=1}^{m} \frac{1}{N - n_{i-1}} - \sum_{i=1}^{m} X_i^2/2 = 0 \qquad (4.9)$$

$$\frac{n}{\phi} - \sum_{i=1}^{m} (N - n_{i-1}) X_i^2/2 = 0 \qquad (4.10)$$

에서 구해진다. 신뢰성 척도로는 다음과 같은 것들이 제시되었다.

① $N_T = N - n$

② $MTTF_T = \{\pi/[2(N-n)\phi]\}^{1/2}$

③ $R_T(t) = \exp\{-(N-n)\phi t^2/2\},\ t \geq 0$

④ $E[T_F] = \sum_{i=1}^{N-n} \left(\frac{\pi}{2i\phi}\right)^{1/2}$ (4.11)

$$Var(T_F) = \sum_{i=1}^{N-n} (2 - \frac{\pi}{2})/(i\phi)$$

(3) Wagoner 모델

Wagoner의 모델은 각 오류에 대하여 고장발생율은 동일하며, 각 오류에 의해 발생되는 고장까지의 시간이 Weibull분포를 따른다고 가정하였다.

Harzard함수 $\lambda(t)$는

$$\lambda(t) = (\alpha/\beta)\ (\frac{t}{\beta})^{\alpha-1},\ t \geq 0 \qquad (4.12)$$

단,

α : Weibul 분포의 shape 모수

β : Weibull 분포의 scale 모수

와 같다. 모델의 입력자료는 t_i, n_i, n이며, 모델의 모수 α와 β를 추정하기 위하여 다음 선형 방정식에 대해 최소자승법(least square method)을 적용하여 a, b를 추정하였다.

$$Y(i) = a + bX(i) \qquad (4.13)$$

단, $Y(i) = \ell_n\{\ell_n[1/(1-F(i))]\},\ i = 1, 2, \cdots m$

$X(i) = \ell_n t_i \qquad i = 1, 2, \cdots m$

$F(i) = n_i/n \qquad i = 1, 2, \cdots m$

이 결과로 부터 α, β는

$$\alpha = [\sum_{i=1}^{m} (Y(i) - Y1)\ (X(i) - X1)]/\sum_{i=1}^{n} (X(i) - X1)^2$$

$$\beta = \exp\{-(Y1 - \alpha X1)/\alpha\} \qquad (4.14)$$

단, X1 : X(i)의 기하평균(geometric mean)

Y1 : Y(i)의 기하평균

와 같이 추정된다. 신뢰성 척도는

① $MTTF_T = \alpha\beta\Gamma(1/\alpha)$

② $R_T(t) = \exp\{-(t/\beta)^\alpha\},\ t \geq 0$ (4.15)

단, $\Gamma(\cdot)$: 감마함수(gamma function)

들이다.

(4) Lipow모델

이 모델은 Schick와 Wolverton모델을 수정한 것으로서 harzard 함수 $Z_i(t)$가

$Z_i(t) = (N - n_{i-1})\ \phi(t_{i-1} + \frac{t}{2}),\ t \geq 0$ (4.16)

이다. 입력자료 X_i, m, m_i로 부터 모델의 모수 N, ϕ를 최우추정법에 의해 구한다. 즉,

$$\sum_{i=1}^{m} \frac{1}{N - n_{i-1}} - \sum_{i=1}^{m} (t_{i-1}X_i + X_i^2/2) = 0$$

$$\frac{n}{\phi} - \sum_{i=1}^{m} (N - n_{i-1})\ (t_{i-1}X_i + X_i^2/2) = 0 \qquad (4.17)$$

의 해가 N, ϕ의 최우추정치(maximum likelihood estimate)이다. 모델의 신뢰성 척도는 다음과 같다.

① $N_T = N - n$

② $MTTF_T = \int_0^\infty R_T(t)dt$ (4.18)

③ $R_T(t) = \exp\{-(N-n)\phi(T_i + t^2/4)\},\ t \geq 0$

(5) Goel과 Okumoto 모델

Goel과 Okumoto의 모델은 NHPP(Non-Homogeneous Poisson Process) 모델이라고 하며, Moranda[42]의 모델과 유사하다. 여기서는 각 오류에 대한 harzard 함수 $\lambda(t)$를

$\lambda(t) = a\ b\ e^{-bt},\ t \geq 0$ (4.19)

라 두고, a, b는 상수로서 모델의 모수이다. 모델에서 요구되는 입력자료는 t_i, m(=n)이며, 모수 a, b의 최우추정치는 다음 방정식들의 해이다.

$$\frac{n}{a} - 1 + \exp(-bT) = 0$$

$$\frac{n}{b} - \sum_{i=1}^{m} t_i - aT \exp(-bT) = 0 \qquad (4.20)$$

다음과 같은 신뢰성 척도를 제시한다.

① $E[N_T] = a \exp(-bT)$ (4.21)

② $R_T(t) = \exp\{-a[e^{-bt} - e^{-b(t+T)}]\}$, $t \geq 0$

이 모델은 그 후 Okumoto와 Goel[50, 51]에 의하여 software 최적 release 시기를 결정하는 모델에 응용되었다.

이상에서 살펴본 모델들외에도 Angus 등[1] Duvall등[9], Forman과 Singpurwalla[11], Koch와 Kubat[25], Moranda[45, 46], Shanthikumar[61, 62, 63]등이 있다.

이러한 black-box 모델은 모델의 적용 및 모수의 추정은 비교적 쉬운편이나 시스템의 모든 오류가 균일한 기회로 시스템의 고장발생에 영향을 미친다고 가정하기 때문에 software의 실행과정, 오류의 특성등을 감안할 때 실제 적용에 다소 문제점이 있다.

2. Markov 모델

Software 시스템의 내부구조와 실행과정에서의 동작특성을 신뢰성 모델에 반영하고져 시도한 것은 Littlewood[32]이다. Littlewood[30, 32]는 software 시스템이 R개의 module로 구성되어 있으며, 시스템이 실행중일때 각 module간의 관계는 천이확률(transition probability)로서 설명될 수 있고, 주어진 처리조건에 따라 각 module이 실행되는 과정은 Markov 과정을 따른다고 하였다. 또 각 module의 고장과정은 poisson 과정을 따르고, 시스템의 고장율이 각 module간의 실행천이율에 비해 아주 작다는 가정하에서 시스템의 asymptotic behavior를 분석하였다.

즉, 그는

ν_i : module i의 고장율

λ_{ij} : $i \neq j$일때 module i에서 j로 천이할 때 고장이 일어날 확률

π_i : 평형상태방정식(equilibrium state equation)

$$\Pi Q = 0, \quad \sum_i \pi_i = 1$$

의 해, 단 Q는 transition rate matrix, $Q = \{q_{ij}\}$

이고, $\lambda_{ij} \to 0$, $\nu_i \to 0$ 일때 시스템의 고장과정은 고장율이

$$\sum_i \pi_i [\nu_i + \sum_{j \neq i} q_{ij}\, \lambda_{ij}] \qquad (4.22)$$

인 poisson 과정이 된다는 결과를 제시하였다. 따라서 시스템의 다음 고장때까지의 시간을 T라고 하면 근사적으로

$$E[T] \simeq \{\sum_i \pi_i [\nu_i \sum_{j \neq i} q_{ij}\, \lambda_{ij}]\}^{-1} \qquad (4.23)$$

이다.

이 모델의 실효성은 Markov Process가 software의 실행과정의 동작특성을 얼마나 잘 나타내느냐에 달려있다. Kim[23]이 실시한 대학의 학적업무용 on-line software에 관한 한 조사결과에 의하면 이 software의 동작특성이 stationary Markov Process를 따르고 있음을 보여주고 있다. 그러나 현재 software의 동작특성이 Markov Process를 따른다는 일반적인 결론을 내릴만한 연구결과나 근거가 존재한다고는 할 수 없다. Littlewood의 Semi-Markov Model[31]이나 Modular Program Structure Model[36]은 위에서 고찰한 모델을 확장한 모델이라 할 수 있다.

3. Software 유용성 모델

Software 유용성 모델은 software가 운영단계에 있을때 신뢰성과 아울러 유용성에 관한 정보를 제공한다. 오류수정시간(error correction time)을 모델에 포함한다는 점이 신뢰성모형과의 중요한 차이점이다. 그러나 software시스템의 오류수정시간을 정의하기 어렵고, 이에 대한 실제자료를 얻기가 어렵다. 이러한 이유로 software 유용성 모델은 Trivedi와 Shooman[75], Okumoto와 Goel[53], Kim등[22, 23], Shanthikumar[65]의 모델등이 있는 정도이다.

그러나 실제 운영되고 있는 시스템의 효율(effectiveness)을 나타낸다는 관점에서 보면 유용성의 중요성은 신뢰성보다 큰 실정이다.

Trivedi와 Shooman은 최초 미지의 수 n개의 오류를 포함하는 software시스템이 오류가 모두 제거될 때까지 가동상태(up-state)와 고장상태(down-state)를 반복하는 과정이 계속될 때, 시스템의 고장과정을 고장율 λ와 수리율 μ인 Poisson 과정으로 간주하고 시스템의 유용성을 추정하는 모델을 개발하였다. 이들의 모델을 간략히 소개하면 다음과 같다.

$p_{n-k}(t)$: 시간 t에 시스템이 (n−k)개의 오류를 포함한채 가동상태에 있을 확률

$q_{n-k}(t)$: 시간 t에 시스템이 (n−k)개의 오류를 포함한채 고장상태에 있을 확률

이라고 할때 시스템의 상태확률(state probability)은 다음 미분방정식(differential equations)을 만족한다.

$$p_{n-k}(t)+\lambda p_n(t)=0$$

$$p_{n-k}(t)+\lambda p_{n-k}(t)=\mu q_{n-k+1}(t),\quad k=1,2,\cdots$$

$$q_{n-k}(t)+\mu q_{n-k}(t)=\lambda p_{n-k}(t),\quad k=0,1,\cdots$$

단, $p_k(t)=\frac{dp_k(t)}{dt}$, $p_k(t)=\frac{dq_k(t)}{dt}$ (5.1)

초기조건(initial condition)은,

$$p_n(0)=1$$

$$p_{n-k}(0)=0,\quad k=1,\ 2,\ \cdots \qquad (5.2)$$

$$q_{n\ k}(0)=0,\quad k=0,\ 1,\ \cdots$$

고장율 λ와 수리율 μ가 상수일때는 (5.1)식, (5.2)식으로 부터 $p_{n-k}(t)$의 해를 구하고, 시스템의 유용성 A(t)는

$$A(t)=\sum_{k=0}^{m} p_{n-k}(t) \qquad (5.3)$$

가 된다. 고장율 λ가 오류의 수에 비례하는 경우에 대해서는 수치해석적인 방법에 의하여 $p_{n-k}(t)$를 구하였다.

이 모델은 Okumoto와 Goel [53]에 의하여 오류를 완전히 수정하지 못할 가능성을 포함하는 모델로 확장되었다. 그리고 이 모델은 Kim등 [22]에 의해 시스템의 고장율이 시스템내에 있는 오류의 수에 비례하고, 고장시에 오류를 수정하는 과정에서 옳바르게 수정하여 시스템내의 오류의 수가 1개가 제거되는 경우와 또 다른 오류를 범하여 시스템내의 오류의 수가 변하지 않게 되는 두 경우를 고려하여 해석적인 해를 구하는 모델로 수정되었다.

그리고 Shanthikumar [65]는 Kim등의 모델에서 오류수정시에 수리율이 시스템내에 있는 오류의 수에 의존하는 모델로 수정하여 해석적인 해를 구하였다.

이상의 software 유용성에 관한 모델외에 최근들어 hardware에서의 문제와 결합한 hardware/software 유용성 모델이 연구되고 있다. 이러한 모델로는 Angus와 James [2], Goel과 Soenjoto [17], Haynes와 Thompson [19], Kline [24]들이 있다.

4. Software release 시기 모델

Software 시스템의 신뢰성 모델은 시스템의 검정단계를 끝내고 release하는 적절한 시기를 결정하는데 보조수단은 되지만, 이러한 문제의 직접적인 답을 제공하지는 못한다. Software release 시기모델은 수학적으로 최적 release 시기를 결정하는 것을 목적으로 하는데 Forman과 Singpurwalla [12], Okumoto와 Goel [54, 55], Shanthinumar와 Tufekci [67, 68]들의 모델이 이에 속한다고 할 수 있다.

Forman과 Singpurwalla는 비용을 고려하지 않았으나 Okumoto와 Goel은 그들의 NHPP모델 [15]을 이용하여 software의 최적 release시기결정을 위한 비용모델을 제시하였다. 그러나 그들의 비용모델은 software 고장의 동작특성과 자금의 시간적 가치를 고려하지 않았다. Shanthikumar와 Tufekci는 software release 시기 결정문제를 일반화된 decision tree 문제로 모델을 개발하고 이를 위한 알고리즘(algorithm)을 제공하였다. 이외에도 Koch와 Kubat [25], Kreten과 Levy [26]의 연구를 들 수 있다.

V. 결 언

지금까지 우리는 software 신뢰성 모델 전반에 걸쳐서 문헌에 자주 인용된 바 있는 대표적인 신뢰성 모델의 개요를 살펴보았다. 이들 모델 가운데에는 Musa[49]와 같이 모델의 적용이 성공적이었음을 보고한 경우도 있고 Schick-Wolverton[58], Angus[1] 등과 같이 일부 모델에 국한된 것이지만 모델 적용의 문제점을 보고한 것도 있다. Miyamoto[41]는 기존 모델 대부분이 실제 적용과정에서 발생하는 문제의 원인이 모델 수립과정에서 설정한 그릇된 가정에 있음을 지적하고 있다. Dale-Harris[6]는 Sukert[71]의 실제 software모델의 비교결과를 인용하여 어떤 특정한 software에 포함된 오류의 수를 추정하는 데 한 모델에서는 8개 또 다른 모델에서는 732개나 되는 즉 1:10의 차이가 나는 상반된 결과가 나타나고 있음을 예시하고 있다.

모델의 적용결과가 비현실적이거나 우리의 상식을 벗어날 정도로 정확하지 못한 원인으로는 다음과 같은 요인을 생각해 볼 수 있다.

(1) 모델을 세울 때 설정한 그릇된 가정

(2) 불충분하거나 또는 정확하지 않은 데이타

(3) 적합치 못한 모델

결과적으로 현재 software신뢰성에 관한 기술수준은 우리의 기대와는 거리가 먼 상태에 있는 것이 현실정이다.

Software신뢰성을 보다 효과적으로 평가하고 오류를 사전에 예방할 수 있는 수단을 제시하기 위해서는 보다 많은 모델의 적용결과와 오류에 관한 실제 데이타의 수집, 분석을 통해 언제, 어떻게, 왜 사람들이 software life cycle을 통해서 오류를 범하게 되며, 또 어떠한 방법으로 이러한 오류를 탐지하며 수정하는가를 이해할 필요가 있다. 이를 위해서 무엇보다도 중요한 것은 효과적인 데이타의 수집 분석체제이다. 현재의 데이타의 수집분석과정에는 Miyamoto[41]가 지적한 바와 같이 많은 문제점을 내포하고 있다.

References

1. Angus, J.E., R.E. Schafer, and A. Sukert, "Software Reliability Model Validation", Proc. Annual Reliability and Maintainability Symposium 1980. pp. 191–198.
2. Angus, J.E. and L.E. James, "Combined Hardware/Software Reliability Models", Proc. Annual Reliability and Maintainability Symposium 1982, pp. 176–181.
3. Basin, S.L., "Estimation of Software Error Rate Via Capture-Recapture Sampling, "Science Application, Inc., Palo Alto, CA, 1974.
4. Brown, J.R. and M. Lipow, "Testing for Software Reliability", Proc. 1975 Int. Conf. Rel. Software, Los Angeles, CA, Apr. 1975.
5. Cheung, R.C., "A User-Oriented Software Reliability Model, "Proc. COMPSAC 1978, pp. 565–570.
6. Dale, C.J. and L.N. Harris, "Approaches to Software Reliability Prediction", Proc. Annual Reliability and Maintainability Symposium, 1982, pp. 167–175.
7. DeMillo, R.A., Lipton, R.J. and Sayward, F.G., "Hints on Test Data Selection:Help for the Practicing Programmer", Computer, Apr. 1978, pp. 34–41.
8. Dickson, J.C., J.L. Hesse, A.C. Kientz, and M.L.Shooman, "Quantitative Analysis of Software Reliability", Proc. Annual Reliability and Maintainability Symposium, 1972, pp. 148–157.

9. Duvall, L., J.Martens, D Swearingen, and J.Donahoo, "Data Needs for Software Reliability Modelling," Proc. Annual Reliability and Maintainability Symposium, 1980, pp. 200–208.
10. Endres, A., "An Analysis of Errors and Their Causes in System Program", Proc. International Conference on Reliable Software, 1975, pp. 327–336.
11. Forman, E.H. and N.D. Singpurwalla, "An Empirical Stopping Rule for Debugging and Testing Computer Software," Journal of American Statistical Association, Vol. 72, 1977, pp. 750–757.
12, Forman, E.H. and N.D. Singpurwalla, "Optimal Time Intervals for Testing Hypothesis and Computer Software Errors," IEEE Transaction on Reliability, Vol. R–29, 1979, pp. 250–253.
13. Goel, A.L., "A Summary on the Discussion on 'An Analysis of Competing Software Reliability Models'," IEEE Transactions on Software Engineering, Vol. SE–6, 1980, pp. 501–502.
14. Goel, A.L. and K. Okumoto, "A Time Dependent Error Detection Rate Model for a Large Scale Software System", Proc. Third USA–Japan Computer Conference, 1978, pp. 35–40.
15. Goel, A.L. and K. Okumoto, "A Markovian Model for Reliability and Other Performance Measures, Proc. National Computer Conference, 1979, pp. 769–774m
16. Geol, A.L. and K. Okumoto, "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures", IEEE Transactions on Reliability, Vol. R–28, 1979, pp. 206–211.
17. Goel, A.L. and J.Soenjoto, "Models for Hardware-Software System Operation-Performance Evaluation", IEEE Transactions on Reliability, Vol. R–30, 1981, pp. 232–239.
18. Halstead, M.H., "Elements of Software Science", New York:Elsevier, 1977.
19. Haynes, R.D. and W.E. Thompson, "Combined Hardware and Software Availability", Proc. Annual Reliability and Maintainability Symposium, 1981, pp. 365–370.
20. Islam, Md. M. and F. Lombardi, "Estimation of Total Errors in Software", Microelectronics and Reliability, Vol. 22, 1982, pp. 281–285.
21. Jelinski, Z. and P.B. Moranda, "Software Reliability Research", Statistical Computer Performance Evaluation, W. Freiberger(ed.), 1972, pp. 465–484.
22. Kim, J.H., Kim, Y.H. and Park, C.J., "A Modified Markov Model for Estimation of Computer Software Performance," Journal of Operations Research Letters, Vol. 1–6, Dec. 1982.
23. Kim, J.H., "A Markov Model for the Prediction of Computer Software Availability", Korea Univ., 1983, unpublished Ph.D. dissertation.
24. Kline, M.B., "Software and Hardware R & M:What are the Differences?" Proc. Annual Reliability and Maintainability Symposium, 1980, pp. 179–185.
25. Koch, H.S. and P. Kubat, "Optimal Release-Time of Computer Software", Working Paper No. 8017, Graduate School of Management, University of Rochester, 1980.
26. Keten, O.J. and D.A. Levy, "Software Modelling for Optimal Field Entry", Proc. Annual Reliability and Maintainability Symposium, 1980, pp. 410–414.
27. Kubat, P. and H.S. Koch, "On the Estimation of the Number of Errors and Reliability of Software Systems", Working paper No. 8012, Graduate School of Management, University of Rochester, 1980.
28. LaPadula, L.J , "Engineering of Quality Software System", Report RADCTR–74–325, Vol. VIII:Software Reliability Modelling and Measurement Techniques, 1973.

29. Lipow, M., "Estimation of Software Package Residual Errors", Report TRWSS-72-09, TRW Software Series, Redondo Beach, CA., 1972.
30. Littlewood, B.,"A Reliability Model for Systems with Markov Structure", Applied Statistics, Vol. 24, 1975, pp. 172-177.
31. Littlewood, B., "A Semi-Markov Model for Software Reliability with Failure Costs", Proc. MRI Symposium on Software Engineering, 1976, pp. 281-300.
32. Littlewood, B., "How to Measure Software Reliability and How Not To", IEEE Transactions on Reliability, Vol. R-28, 1979, pp. 103-110(also Proc. International Conf. on Software Engineering, 1978, pp. 37-45)
33. Littlewood, B., "A Bayesian Differential Debugging Model for Software Reliability", Proc. COMPSAC, 1980, pp. 489-500.
34. Littlewood, B., "Theory of Software Reliability:How Good are They and How Can They be Improved", IEEE Transactions on Software Engineering Vol. SE-6, 1980, pp. 489-500.
35. Littlewood, B., "Stochastic Reliability Growth:A Model With Applications to Computer Software Faults and Hardware Design Faults", Performance Evaluation Review, Vol. 10, 1981, pp. 139-152.
36. Littlewood, B., "Software Reliability Model for Modular Program Structure", IEEE Transactions on Reliability, Vol. R-28, No. 3, Aug. 1979.
37. Littlewood, B. and J.L. Verrall, "A Bayesian Reliability Model with a Stochastically Monotone Failure Rate", IEEE Transactions on Reliability, Vol. R-30, 1981, pp. 108-114.
38. Miller, D., "Exponential Order Statistics Models of Software Reliability Growth," IEEE Transactions on Reliability, Jan. 1986.
39. Miller, H.D., "On the Statistical Validation of Computer Programs", Report 72-6015, IBM Federal Systems Division, Gaithesburg, MD, 1972.
40. Miller, H.D., "On the Development of Large Reliable Software", Rec. IEEE Symp. Comput. Software Rel. New York, May 1973, pp. 155-159.
41. Miyamoto, I., "Towards an Effective Software Reliability Evaluation", Proc. International Conference on Software Engineering, 1978, pp. 45-55.
42. Moranda, P.B., "Prediction of Software Reliability During Debugging", Proc. Annual Reliability and Maintainability Symp., 1975, pp. 327-332.
43. Moranda, P.B., "Quantitative Methods for Software Reliability Measurements", Defence Documentation Center, Alexandria, VA., 1976.
44. Moranda, P.B., "Limits to Program Testing with Random Number Inputs", Proc. COMPSAC, 1978, pp. 521-526.
45. Moranda, P.B., "Event-Altered Rate Models for General Reliability Analysis", IEEE Transactions on Reliability, Vol. R-28, 1979, pp. 376-381.
46. Moranda, P.B., "An Error Detection Model for Application During Software Development", IEEE Transactions on Reliability, Vol. R-30, 1981, pp. 309-312.
48. Musa, J.D., "A Theory of Software Reliability and Its Application", IEEE Transactions on Software Engineering, Vol. SE-1, 1975, pp. 312-3327.
49. Musa, J.D., "Validity of Execution-Time Theory of Software Reliability", IEEE Transactions on Reliability, Vol. R-28, 1979, pp. 181-191.
50. Musa, J.D., "The Measurement and Management of Software Reliability", Proc. of The IEEE, Vol. 68, No. 9, Sep. 1980.

51. Nelson, E.C., "A Statistical Basis for Software Reliability Assessment", TRW Software Series TRW–SS–73–03 Redondo Beach, CA., 1973.
52. Nelson, E.C., "Estimating Software Reliability from Test Data", Microelectronics and Reliability, Vol. 17, 1978, pp. 67–74.
53. Okumoto, K and A.L. Goel, "Availability and Other Performance Measures of System Under Imperfect Maintenance", Proc. COMPSAC, 1978, pp. 66–71.
54. Okumoto, K and A.L. Goel, "Optimum Release Time for Software Systems", Proc. COMPSAC, 1979, pp. 500–503.
55. Okumoto, K and A.L. Goel, "Optimal Release Time for Software Systems Based on Reliability and Cost Criteria", Journal of Systems and Software, Vol. 1, 1980, pp. 315–318.
56. Ramamoorthy, C.V. and F.B. Bastani, "Software Reliability-Status and Perspectives", IEEE Transactions on Software Engineering, Vol. SE–8, No. 4, 1982, pp. 354–371.
57. Schick, G.J. and R.W. Wolverton, "Assessment of Software Reliability", Proc. Operations Research, Physica-Verlag, Wurzberg-Wien, 1973, pp. 395–422.
58. Schick, G.J. and R.W. Wolverton, "An Analysis of Competing Software Reliability Models", IEEE Transactions of Software Engineering, Vol. SE–4, 1978, pp. 104–120.
59. Schneider, V., "Some Experimental Estimators for Developmental and Delivered Errors in Software Development Projects", Proc. COMPSAC, 1980, pp. 495–498.
60. Schneidewind, N.F. and H.Hoffman, "An Experiment in Software Error Data Collection and Analysis", IEEE Transactions on Software Engineering, Vol. SE–5, No. 3, May 1979.
61. Shanthikumar, J.G., "Software Performance Prediction Using State-Dependent Error Occurrence-Rate Model", Proc. 19th Annual Technical Symp. Pathways to System Integrity, 1980, pp. 67–73.
63. Shanthikumar, J.G., "A State-and Time-Dependent Error Occurrence-Rate Software Reliability Model with Imperfect Debugging", Proc. National Computer Conference, 1981, pp. 311–315.
64. Shanthikumar, J.G., "A General Software Reliability Model for Performance Prediction", Microelectronics and Reliability, Vol. 21, 1981, pp. 671–682.
65. Shanthikumar, J.G., "On A Software Availability Model with Imperfect Maintenance", Working Paper 83–011, Univ. of Arizona, Apr. 1983.
66. Shanthikumar, J.G., "Software Reliability Models:A Review", Microelectronics and Reliability, 1983.
67. Shanthikumar, J.G. and S. Tufekci, "Optimal Software Release Time Using Generalized Decision Trees", Proc. 14th Annual Hawaii International Conference on System Sciences, 1981, pp. 58–65.
68. Shanthikumar, J.G. and S. Tufekci, "Application of a Software Reliability Model to Decide Software Release Time", Microelectronics and Reliability, Vol. 22, 1982.
69. Shooman, M.L., "Software Engineering:Reliability, Design, Management," McGraw-Hill Book Co. N.Y., 1980.
70. Shooman, M.L., "Probabilistic Models for Software Reliability Prediction," Statistical Computer Performance Evaluation,W. Freiberger, Ed., New York:Academic, 1972, pp. 485–502.
71. Sukert, A.N., "Empirical Validation of Three Error Prediction Models", Proc. COMPSAC, 1978, pp. 577–582.
72. Sukert, A.N. and A.L. Goel, "A Guidebook for Software Reliability Assessment",Proc. Annual Reliability and Maintainability Symposium, 1980, pp. 186–190.

73. Thayerm, T.A., M Lipow, and E.C. Nelson, "Software Reliability Study", Report RADC–TR–76–238 Rome Air Development Center, Rome, N.Y., 1976.
74. M.Trachtenberg, "The Linear Software Reliability Model and Uniform Testing", IEEE Transactions on Reliability, Vol. 34, No.1, Apr., 1985.
75. Trivedi, A.K. and M.L. Shooman, "A Many State Markov Model for the Estimation and Prediction of Computer Software Performance Parameters", Proc. International Conf. on Reliable Software, 1975, pp. 208–220.
76. Wagoner, W.L., "The Final Report on a Software Reliability Measurement Study", Report TOR–0074(41221)–1, The Aerospace Corp., El Sequndo, CA., 1973.