韓國OA學會論文誌
第1卷 第1號, 1996. 8

論文 96-1-1-4-7

지연 S자형 소프트웨어 신뢰도 성장모델에 관한 연구

문 외 식*

A Study On The Delayed S Shaped Software Reliability Growth Model

Wae Sik Moon*

요 약

본 논문에서는 소프트웨어 신뢰성을 평가하는 신뢰도 성장모델 중에서 NHPP(Non Homogeneous Poission Process)를 기초로 하는 지연 S자형 신뢰도 성장모델의 파라미터값 추정 및 적합도 검정 등을 통하여 소프트웨어 품질평가를 위한 신뢰성 평가척도를 추정하고 예측할 수 있는 신뢰도 평가 지원도구를 설계 및 구현하고 구현된 도구에 실제 관측된 에러 발견수 데이터를 적용하고 비교 분석한다.

ABSTRACT

For predicting the parameters and estimating the goodness of fit reliability growth model based on NHPP(Non Homogeneous Poission Process) among various reliability growth models, a Delayed S Shaped SRGM Tool is designed and implemented. The implemented tool is applied to real software error data, and the result is compared and annalized.

Ⅰ. 서 론

소프트웨어 품질의 신뢰성을 향상시키기 위해서는 신뢰성 실현기술, 검증 및 분석기술, 신뢰성 평가기술이 필요하다[1]. 신뢰성 실현기술은 신뢰성 높은 소프트웨어 개발을 위한 프로세서 또는 방법론 등을 말하며 검증 및 분석기술은 소프트웨어 개발중 신뢰성이 높은 것을 나타내기 위한 방법론 등을 말한다. 신뢰성 평가기술은 개발중이거나 이미 개발된 소프트웨

* 창원전문대학 사무자동화과

어의 신뢰성이 사용자가 만족하는 수준인가를 평가하면서 다른 소프트웨어 신뢰성을 비교 검토하기 위한 프로세서 또는 방법론을 말한다.

특히 평가기술에 관해서는 인간의 경험이나 주관에 따라서 정성(定成)적인 평가를 하는 것이 아니고 객관적인 데이터를 기초로 해서 신뢰성의 비교·검토를 행하는 가능한 정량(定量)적인 평가를 실현하는 것이 좋다. 1970년대 부터 시작되어온 소프트웨어 품질향상을 위한 신뢰도 연구가 꾸준한 발전을 거듭하고 있지만 하아드웨어와 같은 명확한 현상을 고려한 완전한 소프트웨어 신뢰도 모델은 아직 어려운 실정이다. 대부분의 소프트웨어 신뢰도 모델들은 일정한 시각까지 데스트한 결과 얻어지는 에러수를 기초로 소프트웨어 내에 잔존하는 에러수를 예측하고 이 결과에 따라 사용자가 어느정도 만족할만한 수준의 에러가 잔존하는 상태로 소프트웨어를 Release 시키고 있는 실정이다. 이러한 이유등으로 소프트웨어 신뢰성에 관련된 논문들이 에러예측의 수리적 모델을 많이 연구되고 있다. 수리적 모델 중에서 신뢰성 평가를 위해서 소프트웨어 개발과정에서 프로그램 속에 삽입된 에러등을 소프트웨어 개발 마지막 단계인 테스트에 의해서 수집하여 이를 소프트웨어 품질 및 신뢰성을 높이는 정량적 방법으로 Jelinski와 Moranda가 처음 제안한 소프트웨어 신뢰도 성장모델(SRGM : Software Reliability Growth Model)이 있다. SRGM[2]은 테스트에 의해서 발견된 총에러수와 소요된 테스트 시간과의 관계를 수학적으로 표현한 것을 말하며 SRGM에서 도출되는 평가척도에 의해서 사용자가 만족할만한 소프트웨어 신뢰성 목표의 달성상황 또는 Release되는 소프트웨어 품질파악 등을 할 수 있다. 본 논문에서는 SRGM 중에서 비교적 중·소규모의 소프트웨어 품질평가에 적합한 지연 S자형 신뢰도 모델의 수리적 예측기법을 고찰하여 이론적 기법을 정리하고 알고리즘 구현으로 평가도구를 설계하고 구현하였으며 테스트에 의해 수집된 실측 에러 데이터를 구현된 도구에 적용하여 비교·분석하였다.

Ⅱ. 관련 연구

2.1 소프트웨어 신뢰도 정의

소프트웨어 신뢰도란 주어진 일정기간 동안 그환경 아래에서 소프트웨어가 고장없이 명세서대로 정상적으로 사용될수 있는 확률로 정의된다[3]. 여기에서 언급된 신뢰도, 주어진 일정기간, 환경, 명세서, 확률에 대한 정의는 다음과 같다.

① 일정기간이란 프로그램의 실행시간인 CPU 실행시간 또는 캘린더(calander)시간을 말하며 신뢰도는 주어진 일정기간 동안 측정된다.

② 환경이란 프로그램 개발 및 운영환경을 말하며 같은 프로그램 이라도 환경이 다르면 신뢰도는 다르게 나타난다. 이때의 환경은 요구사항 정의, 계획, 설계, 구현, 태스트 및 유지 보수의 각 단계의 환경을 말하며 운영환경 보다 개발환경이 더 신뢰도에 영향을 미치므로 이러한 요소들을 많이 고려할수록 좋은 모델이라 할 수 있다.

③ 명세서란 소프트웨어 요구정의를 말하며 이는 소프트웨어의 정확한 동작을 평가하는 기준이 되므로 신뢰도는 주어진 명세서를 기초로 해야 한다. 예로서 실행결과에 에러가 발생하더라도 명세서에서 허용가능한 범위 이내이면 이를 고장이라 하지 않는다.

④ 신뢰성이라는 것은 신뢰할수 있는가 없는가 하는 추상적인 표현을 말하며 이것을 양적으

로 표현한 것이 신뢰도라 말할 수 있다. 따라서 신뢰성의 척도로서 신뢰도는 확률이므로 신뢰도 추정을 위해 추정이론(maximum likelihood estimation), 각종 확률분포(weibul, gammer, beta, poisson, exponential, rayleigh)와 마코브프로세스, 최우추정 개념[4] 등의 통계적 기법이 사용된다.

2.2 용어정의

소프트웨어 신뢰도에서 사용하게 되는 결함(fault), 에러(error), 고장(failure)의 개념은 다음과 같다.

① 결함 - 개발자의 실수에 의해 소프트웨어 내에 삽입되는 것으로 이는 프로그램 실행 결과가 본래 주어진 명세서와 다를때를 결함이라 하며 이는 버그(Bug)와 동의어이다. 따라서 디버깅(Debuging)이란 이러한 결함을 없애는 작업을 말한다.

② 에러 - 소프트웨어 결함을 발생시키는 우발적 또는 부주의한 행위를 말한다.

③ 고장 - 결함이 있으면 생길수 있는 현상이 바로 고장이며 이는 소프트웨어가 주어진 요구 명세서 대로 바르게 동작하지 않는 것을 말한다.

위의 용어들은 통일적인 정의는 되어 있지 않지만 일반적으로 표준화 되어 가고 있으며 소프트웨어 개발 테스트단계 또는 운용단계에서 발견·수정되는 결함을 소프트웨어 에러라고 부르며 소프트웨어 결함(fault)과 구별하지 않고 주로 사용되며 이후 본 논문에서도 결함을 에러라고 부르기로 한다.

2.3 신뢰성 평가 척도

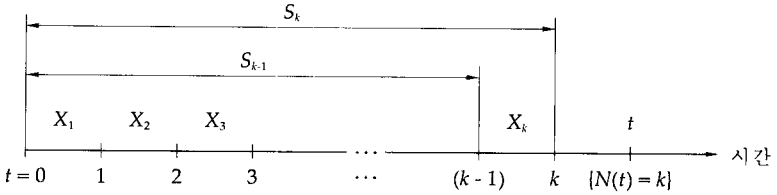
2.3.1 기본적 확률변수

테스트를 통해 수집된 소프트웨어 에러수나 소프트웨어 고장 발생시기에 관련된 기본적인 확률변수는 표 1과 같다.

표 1. 기본 확률변수

Table 1. Basic random variable

확률변수	내 용
$N(t)$	테스트시각 t까지에 발견된 총에러수(고장수)를 나타낸다. $\{N(t),\ t \geq 0\}$
S_k	k번째의 고장발생 시각을 나타낸다. $(k = 0, 1, 2, \cdots, S_0 = 0)$
X_k	$(k\text{-}1)$번째와 k번째 사이의 고장발생 시간간격을 나타낸다. $(k = 0, 1, 2, \cdots, X_0 = 0)$



$(X_0 = 0,\ S_0 = 0)$ $(X$: 소프트웨어 에러 발견, 고장발견 시점)

그림 1. 소프트웨어 고장발생 현상에 대한 확률량

Fig 1. Probability mass about failure occurrence

그림 1은 테스트 시각 t까지에 k개의 에러가 발견되고 있는 것으로 $N(t) = k$라는 사상이 일어난다. 이때, 변수 $N(t)$, S_k 및 X_k는 확률변수로 취급하며 변수 S_k와 X_k에는 식 (1)과 같은 관계가 성립된다.

$$S_k = X_1 + X_2 + X_3 + \cdots + X_k = \sum_{i=1}^{k} X_i,$$
$$X_k = S_k - S_{k-1} \qquad (1)$$

$N(t)$, S_k 또는 X_k의 확률분포로 여러가지 신뢰성 척도를 측정하는 것이 가능하다. t_k까지에 발견된 총에러수 $N(t_k)$의 실제 측정치 $X_k(k = 1, 2, \cdots, n)$에 관한 측정 데이터를 에러 발견수 데이터라 하고 각 고장 발생시각 S_k의 실제 측정치 $S_k(k = 1, 2, \cdots, n)$에 관한 측정 데이터를 고장 발생시간 데이터라 한다.

2.3.2 신뢰성 데이터

본 논문에서는 테스트시간의 계측단위를 기초로하는 S자형 신뢰도 모델에 적합한 데이터 즉 일정의 테스트 시간간격 $(0, t_k]$까지에 발견된 총에러수 $y_k(k = 1, 2, \cdots, n : 0 < t_1 < t_2 < t_n)$에 관한 측정 데이터를 에러발견수 데이터라 하며 이 데이터를 SRGM에 적용하여 소프트웨어 신뢰성 예측을 하는 것은 테스트된 과거의 데이터 $(t_k, y_k)(k = 1, 2, \cdots, n)$를 사용해서 미래의 일정시각 t_{n+1}, t_{n+2}, $\cdots$까지 발견되는 총에러수 $N(t_{n+1})$, $N(t_{n+2})$, $\cdots$의 값을 예측한다[5].

그림 2는 표 2와 같이 나타낼 수 있으며 y_k는 시각 t_k까지 발견된 총에러수를 나타내는 확률변수 $N(t_k)(k = 1, 2, \cdots, n)$ 실제 관측치이다.

2.4 NHPP 소프트웨어 신뢰도 성장모델

본 논문에서는 신뢰성 모델 중에서 발견된 에러수를 기초로 하고 개수계측 모델에 속하는 대표적인 NHPP(NonHomogeneous Poission Process) 모델에 대해서 고찰한다. 임의의 테스트시간 t까지 발견되는 에러수를 열거하는 확률변수 $N(t)$를 도입하여 식 (2)에서 주어진 확률분포 즉, Poission 과정으로 나타낸 SRGM

표 2. 에러발견수 데이터의 t_k와 y_k

Table 2. t_k and y_k Number of detected data

테스트 시간간격$(0, t_k]$	발견된 에러수 y_k
$(0, t_1]$	y_1
$(0, t_2]$	y_2
⋮	⋮
$(0, t_k]$	y_k
⋮	⋮
$(0, t_n]$	y_n

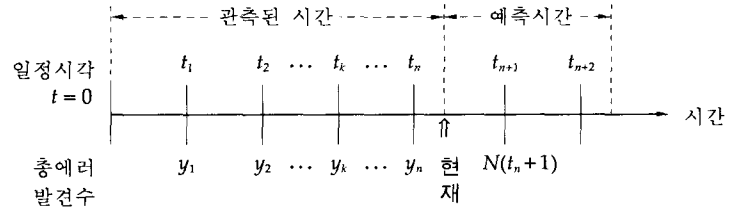


그림 2. 에러 발견수 데이터

Fig 2. Number of detected error data

이 NHPP 모델이다[6]. 이때 Poisson 과정 중에서도 태스트 공정 또는 운용단계에 있어서 에러발견 사상에 대해 다음의 4가지 성질을 만족할 때 $N(t)$는 NHPP에 따른다고 말한다[7].

① $N(0) = 0$ (테스트시각 $t = 0$에서 에러는 발견되지 않는다)

② $\{N(t),\ t \geq 0\}$는 독립증분을 갖는다(서로 다른 테스트 시간구간에서 발견된 에러수는 통계적으로 독립이다)

③ $\Pr\{N(t + \Delta t) - N(t) \geq 2\} = o(\Delta t)$ (임의의 작은 테스트 시간구간 Δt에서 2개 이상의 에러가 발견될 확률은 무시할 정도로 작다)

④ $\Pr\{N(t + \Delta t) - N(t) = 1\} = h(t)\Delta t + o(\Delta t)$ (임의의 작은 테스트 시간구간 Δt에서 1개의 에러가 발견될 확률은 테스트 시각에서 발견율 또는 소프트웨어 고장의 발생율에 비례한다).

위의 4가지 성질을 사용하여 확률변수 $N(t)$의 확률분포를 구할 때 식 (2), (3)을 얻는다.

$$\Pr\{N(t) = n\} = \frac{\{H(t)^n\}}{n!} e^{-H(t)},$$
$$t \geq 0, (n = 0, 1, 2, \cdots) \qquad (2)$$

식 (2)에서 $\Pr\{N(t) = n$은 에러발견 사상확률을 말하며 $H(t)$는 $N(t)$의 평균치를 나타내는 평균치 함수이다. 이는 테스트 시각 t까지 발견되는 총기대 에러수를 나타내며 식 (3)과 같다.

$$H(t) = \int_0^t h(x)\,dx \quad (t \geq 0) \qquad (3)$$

식 (3)의 $h(t)$는 NHPP의 강도함수(intensity function)라고 하며 테스트 시각 t에서 순간 에러 발견율을 나타낸다. 테스트 시작전에 잠재하는 총기대 에러수를 $a(a > 0)$라고 할때 $H(\infty) = a$로 되어 식 (2), (3)에 의해 $N(\infty)$의 확률분포($N(t)$의 極限分佈)는 식 (4)가 되며 충분한 기간에 걸쳐서 테스트할때 $N(t)$의 극한분포는 평균치가 a의 Poisson 분포에 따른다.

$$\lim_{t\to\infty} \Pr\{N(t) = n\} = \frac{a^n}{n!} e^{-a} \ (n = 0, 1, 2 \cdots) \qquad (4)$$

식 (2), (3)에서 정식화된 NHPP 모델에서 SRGM에 유용한 정량적 평가척도를 도출할 수 있다. 테스트 시각 t에서 기대잔존 에러수를 $\overline{N}(t)$라 할 때 $\overline{N}(t) = N(\infty) - N(t)$으로 이때의 $\overline{N}(t)$의 평균치는 식 (5)가 된다.

$$n_r(t) \equiv E[\overline{N}(t)] = E[N(\infty) - N(t)]$$
$$= a - H(t) \qquad (5)$$

테스트가 시각 t까지 진행하고 있을때 시간구간 $(t,\ t+x)$에서 에러가 발생하지 않을 확률은 식 (6)과 같이 되어 이를 소프트웨어 신뢰도라 한다.

$$R(x|t) = \exp[-\{H(t+x) - H(t)\}]$$
$$(t \geq 0,\ x \geq 0) \qquad (6)$$

여러가지 테스트 환경요인을 고려하여 식 (2)의 평균치 함수 $H(t)$에 적절한 함수형을 주는 대표적인 NHPP 모델들이 여러가지 제안되었다[8].

2.5 지연 S자형(delayed S-shaped) 모델

지연 S자형 모델은 山田, yamada등[9]에 의해 제안되었다. 기본 개념은 소프트웨어 고장발생시간에서 부터 그 고장의 원인인 에러를 제거하는 시간까지의 시간지연을 고려하고 있다.

테스트에 의해 에러를 발견하기 위해서는 먼저, 고장현상을 확인한 후 원인분석을 해야 한다는 개념으로 비교적 복잡한 소프트웨어 테스트시 고장의 원인을 분석하여 에러를 특정 지우는 작업에 많은 시간이 필요하며 이러한 시간적 지연을 무시할 수 없다. 소규모의 소프트웨어 테스트에 있어서는 이러한 시간적인 지연이 적다. 여기서 고장현상을 확인하는 과정을 고장 발견과정이라 하며 원인분석을 하여 에러발견에 도달하는 과정을 에러 발견과정이라 한다.

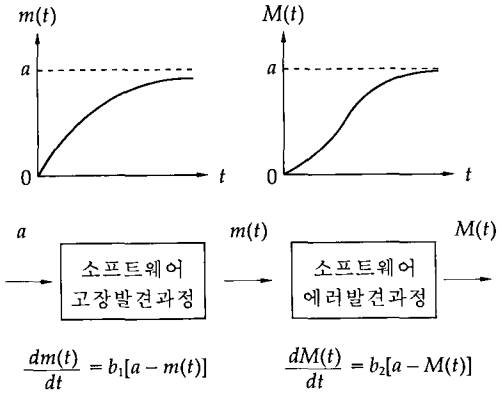


그림 3. 지연 S자형 신뢰도 성장모델

Fig 3. Delayed S-Shaped SRGM

고장발견과정에서 시각 t까지에 발견되는 총 기대 고장율이 $m(t)$일때 식 (7)이 된다.

$$\frac{dm(t)}{d(t)} = b_1[a - m(t)] \tag{7}$$

여기서 파라미터 a는 테스트 시각전 소프트웨어내에 존재하는 총기대 에러수이며 파라미터 b_1은 고장발견율을 나타낸다. 에러발견과정에서 시각 t까지에 발견되는 총기대 에러수가 $M(t)$일때 식 (8)이 된다.

$$\frac{dM(t)}{d(t)} = b_2[m(t) - M(b_1)] \tag{8}$$

식 (8)에서 파라미터 b_2는 에러발견율을 나타내며 식 (7)과 식 (8)의 미분방정식을 $M(t)$에 대해서 풀면 이것을 식 (2)의 평균치 함수라면 식 (9)를 얻는다.

$$H(t) \equiv M(t) = a[1 - (1 + bt)e^{-bt}] \quad (a, b > 0) \tag{9}$$

테스트시각 t에서 순간 에러발견율 $h(t)$와 1개당의 에러발견율은 식 (10)의 각각이 된다.

$$h(t) = ab^2te^{-bt}, \quad d(t) = \frac{b^2}{(1 + bt)} \tag{10}$$

식 (10)에서 $d(t)$의 파라미터 b는 정상상태에서의 1개당의 에러 발견율을 나타내며 그림 4에서 볼 수 있듯이 $d(t)$의 그래프는 0에서 부터 시간의 증가와 함께 $d(\infty) = b$에 가까와 진다. 이는 1개당의 에러 발견율이 서서히 나아지는 것을 나타내고 있다.

신뢰성 평가척도로서 식 (5)의 소프트웨어내의 기대잔존 에러수는 식 (11)이 되며 식 (6)의 소프트웨어 신뢰도는 식 (12)가 된다.

$$n_r(t) = a(1 + bt)e^{-bt} \tag{11}$$

$$R(x|t) = \exp[-a\{(1+bt)e^{-bt} - (1+b(t+x))e^{-b(t+x)}\}] \quad (12)$$

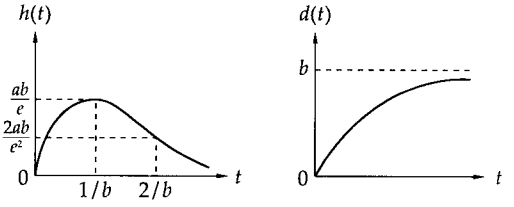


그림 4. 지연 S자형 신뢰도 성장모델의 에러발견률

Fig 4. Detected rate of error about Delayed S-Shaped SRGM

Ⅲ. 파라미터 추정과 데이터 분석과정

3.1 파라미터 추정

본 논문에서는 지연 S자형 모델에 포함된 파라미터값 추정을 위해 에러발견수 데이터를 기초로 한 최우법(Method Of Maximum Likelihood)으로 소프트웨어 내에 포함된 파라미터 추정을 유도한다[10]. 파라미터를 최우법에 의해 추정하기 위하여 일정의 테스트 시간간격 $(0, t_k]$에서 발견된 총 에러수 y_k에 관한 n쌍의 측정 데이터 $(t_k, y_k)(k = 1, 2, \cdots, n)$가 관측되는 것으로 한다. 이때 이 데이터에 대한 평균치 함수 $H(t)$를 가지는 NHPP 모델에 대한 우도함수는 NHPP의 독립증분 성질에 의해 식 (13)으로 된다.

$$L = \Pr\{N(t_1) = y_1, N(t_2) = y_2, \cdots, N(t_n) = y_n\} = \exp[-H(t_n)] \prod_{k=1} \frac{\{H(t_k)\} - H(t_{k-1})\}^{(y_k - y_{k-1})}}{(y_k - y_{k-1})!} \quad (13)$$

식 (13)에서 $t_0 = 0$, $y_0 = 0$으로 초기값을 주고 양변에 자연대수를 취하면 지연 S자형 모델의 대수 우도함수는 식 (14)가 된다.

$$\ln L = y_n \cdot \ln a + \sum_{k=1}^{n} (y_k - y_{k-1}) \ln[(1+bt_{k-1})e^{-bt_{k-1}} - (1+bt_k)e^{-bt_k}] - a[1-(1+bt_k)e^{-bt_n}] - \sum_{k=1}^{n} \ln[(y_k - y_{k-1})!] \quad (14)$$

식 (14)의 대수 우도함수를 정리하면 에러발견수 데이터의 우도방정식은 식 (15)가 되며 이를 수학적으로 풀면 모델 파라미터 최우추정치가 구해진다[11].

$$a = \frac{y_n}{[1 - (1+bt_n)e^{-bt_n}]}$$

$$\frac{y_n t_n^2 e^{-bt_n}}{[1 - (1+bt_n)e^{-bt_n}]} = \sum_{k=1}^{n} \frac{y_n t_n^2 e^{-bt_n}}{\{(1+bt_{k-1})e^{-bt_{k-1}} - (1+bt_k)e^{-bt_k}\}} \quad (15)$$

3.2 데이터 분석과정

3.2.1 데이터 수집

신뢰성 데이터(에러 발견수)를 수집한다. 또한 데이터 분석에 충분한 실측 데이터 수를 적어도 10개 이상 수집한다.

3.2.2 파라미터 추정

에러 발견수 데이터 $(t_k, y_k)(k = 1, 2, \cdots, n)$에 대응하는 우도 방정식을 풀어 모델 파라미터를 추정한다.

3.2.3 평균치함수 $H(t)$ 추정

평균치 함수 $H(t)$의 최우 추정치 $\hat{H}(t)$에서 식 (16)과 같은 추정모델을 얻는다.

$$\Pr\{N(t) = n\} = \frac{\{\hat{H}(t)\}^n}{n!}\exp[-\hat{H}(t)] \quad (n = 0, 1, 2, \cdots) \tag{16}$$

식 (16) $\hat{H}(t)$의 확률 γ에서 추정되는 평균치 함수의 존재범위, 즉 $100\gamma\%$ 신뢰한계를 식 (17)에 의해 계산해 둔다.

$$\hat{H}(t) \pm K_\gamma\sqrt{\hat{H}(t)} \tag{17}$$

3.2.4 적합도 검정

지연 S자형 모델이 관측된 데이터에 적합한가를 통계적으로 판정하기 위하여 적합도 검정을 한다. 본 논문에서는 관측된 데이터수가 적은 경우에도 효과적인 콜모루그로브-스미르노(Kolmogorov Smirnov) 적합도 검정법(K-S 검정법)을 적용하였다. 여기서 에러발견수 데이터 (t_k, y_k) $(k = 1, 2, \cdots, n)$가 관측될 때에는 식 (18)로 되는 K-S 검정량을 계산한다.

$$\begin{cases} D = \max\limits_{1\le i\le n}\{D_i\} \\ D_i = \max\left\{\left|\dfrac{H(t_i)}{H(t_n)} - \dfrac{y_i}{y_n}\right|, \left|\dfrac{H(t_i)}{H(t_n)} - \dfrac{y_{i-1}}{y_n}\right|\right\} \end{cases} \tag{18}$$

K-S 검정법에서는 데이터로부터 계산된 검정 통계량 D와 위험율이 α(1% 또는 5%)일때의 기각한계와 비교된다. 식 (18)에 대해서 자유도가 n일때의 기각한계 $D_{n:\alpha}$와 비교하여 식 (19)의 조건을 만족하면 위험률 α에서 관측데이터에 대해 식 (16)의 지연 S자형 모델은 적합하다고 판정한다.

$$D < D_{n:\alpha} \tag{19}$$

식 (19)의 조건에 맞지 않게 되어 지연 S자형 모델이 적합하지 않다면 추가 데이터를 수집한다.

3.2.5 신뢰성 평가척도 추정

기대잔존 결함수 $n_r(t)$, 소프트웨어 신뢰도 $R(x|t)$등의 신뢰성 평가척도를 추정·예측한다.

3.2의 데이터 분석과정은 그림 5로 나타낼 수 있다.

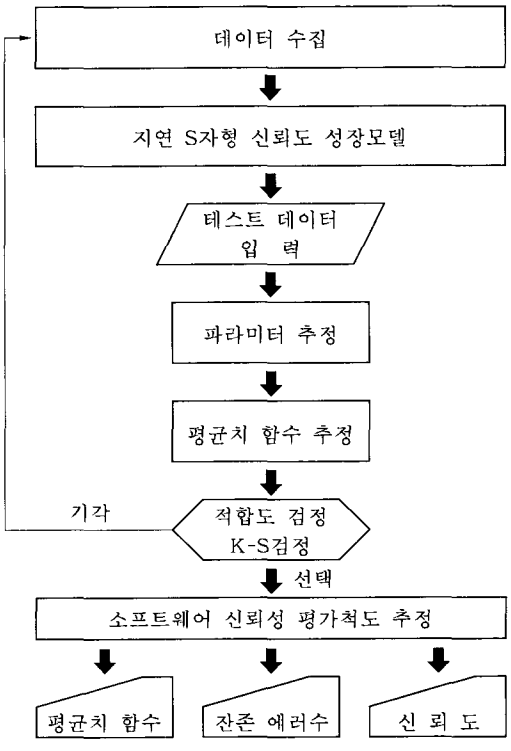


그림 5. 신뢰성 데이터 분석과정

Fig 5. Analysis procedure of reliability data

4. 시스템 설계 및 알고리즘

지금까지 지연 S자형 모델의 수리적 개념을 정립하고 신뢰성 데이터 분석과정을 확립하여 알고리즘을 구현하고 프로그래밍 하여 소프트웨어 신뢰성을 정량적으로 추정 및 예측을 하기 위한 자동화된 지연 S자형 모델을 설계 및 구현한다. 이때 사용된 프로그래밍 언어는 MS-DOS상에서 실행 가능한 Turbo-C이며 실행 결과는 화면에 추정결과 출력 및 그래프로 나타낸다.

4.1 시스템 구조도

지연 S자형 모델의 신뢰도 평가지원 도구는 그림 6에 나타낸 것처럼 6가지의 각각 독립된 루틴으로 구성되어 있다.

4.2 모듈설계 내용

그림 6의 구조도에서 나타낸 각각의 모듈들에 대한 설계 내용은 다음 표 3과 같다.

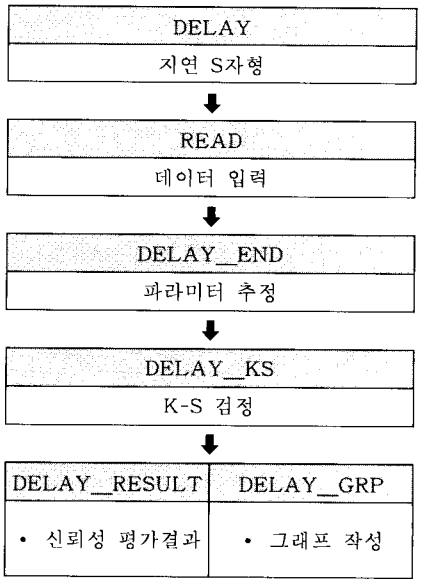


그림 6. 시스템 구조도

Fig 6. System sturcture

표 3. 각 모듈 설계내용

Table 3. Design contents of each module

모 듈 명	기 능
DELAY	지연 S자형 신뢰도 성장모델의 주루틴.
READ	디스크에 입력된 테스트 데이터 선택 루틴 이때, 디스크에 등록된 테스트 데이터는 일정의 테스트 시간간격 $(0, t_k]$까지에 발견된 총에러수 y_k에 관한 측정 데이터인 소프트웨어 에러발견수 데이터를 말한다. 테스트 데이터 파일명은 *.DAT이다
DELAY_END	지연 S자형 신뢰도 성장모델의 파라미터 추정(최우법) 루틴. 파라미터의 최우추정은 비선형 방정식 방법으로 수렴성이 좋은 이분법(bisections method)을 적용함
DELAY_KS	지연 S자형 신뢰도 성장모델의 적합도 검정 루틴(Kolmogorov Smirnov 검정). 이때 위험율을 1% 또는 5%를 선택한다.
DELAY_RESULT DELAY_GRP	지연 S자형 신뢰도 모델에 신뢰성 데이터 분석결과인 평균치함수, 기대잔존에러수 및 소프트웨어 신뢰도, K-S검정량, 편차자승합을 출력 및 그래프로 작성한다.

4.3 파라미터 값 추정 알고리즘 구현

지연 S자형 SRGM 파라미터값 a, b를 추정하는 알고리즘을 C로 구현하였으며 다음과 같다.

```
int del_para(int n, M)
/* n : 에러 발견수 데이터 개수,
  M(n) : 에러 발견수 데이터 */
{
  double b1,b2;                                          /* 1 */
  double b3;                                             /* 2 */
  double rel_b1, rel_b2, rel_b3;
  b1=0.001;                                              /* 3 */
  b2=0.9999;                                             /* 4 */
   for(i=1;i<=200;i++)                                   /* 5 */
   {
        b=b1;                                            /* 6 */
        rel_b1=delay_b(b);                               /* 7 */
        b=b2;                                            /* 8 */
        rel_b2=delay_b(b);                               /* 9 */
        b3=(b1+b2)/2;                                   /* 10 */
        b=b3;                                           /* 11 */
        rel_b3=delay_b(b);                              /* 12 */
        if((fabs(b1-b2)/fabs(b2))<.000001)
          {
                  delay_end(b3);                        /* 13 */
        if((rel_b1*rel_b3)<0)
                  b2=b3;                                /* 14 */
        else
                  b1=b3;                                /* 15 */
        }                                               /* 16 */
  }

  double delay_b(double b_imsy)                         /* 17 */
  {
    double r_over, r_under, l_side;                     /* 18 */
    double first, diff;                                 /* 19 */
    int k;
```

```
  first=m(1)*t(1)*t(1)*exp(-b_imsy*t(1))/
       (1-(1+b_imsy*t(1))*exp(-b_imsy*t(1)));          /* 20 */
  diff=0;
   for(k=2;k<=max-2;k++)
   {
   r_over=(m(k)-m(k-1))*
          (t(k)*t(k)*exp(-b_imsy*t(k))
          -t(k-1)*t(k-1)*exp(-b_imsy*t(k-1)));
   r_under=(1+b_imsy*t(k-1))*
           exp(-b_imsy*t(k-1))-(1+b_imsy*t(k))
           *exp(-b_imsy*t(k));
   diff=diff+r_over/r_under;
  }                                                    /* 21 */

  l_side=m(n)*t(n)*t(n)*exp(-b_imsy*t(n))
         /(1-(1+b_imsy*t(n))*exp(-b_imsy*t(n)));       /* 22 */
 diff=first+diff-l_side;                               /* 23 */
   return(diff);
}

double delay_end(double b_imsy2)                       /* 24 */
{
  double ass_a, ass_b;
  ass_b=b_imsy2;                                       /* 25 */
  ass_a=m(n)/(1-(1+ass_b*t(n))*exp(-ass_b*t(n)));      /* 26 */
}
```

알고리즘에 대한 개념은 주석번호대로 다음과 같이 기술한다.

(1) 에러 발견율 상한 변수

(2) 에러 발견율(상한값+하한값)/2, 이분값

(3) 에러 발견율의 초기치 설정(하한값)

(4) 에러 발견율의 초기치 설정(상한값)

(5) 최대 200번 반복

(6) b에 하한값 할당

(7) 에러 발견율이 $b1$일 때의 우도방정식 값을 구함

(8) b에 상한값 할당

(9) 에러 발견율이 $b2$일 때의 우도방정식 값을 구함

(10) $b3$에 중간값 할당

(11) b에 중간값 할당

(12) 에러 발견율이 $b3$일 때의 우도방정식 값을 구함

(13) 에러 발견율의 상대오차가 10^{-6} 이하가 되면 2분법을 종료하고 최종 파라미터 추정치를 구함

(14) 수렴값이 범위 $(b1, \frac{b1+b2}{2})$ 내에 존재

(15) 수렴값이 범위 $(\frac{b1+b2}{2}, b2)$ 내에 존재

(16) for

(17) b에만 한정된 우도방정식

(18) 우도방정식의 좌우측값

(19) 우도방정식의 첫번째 값과 좌우측 차이값

(20) 우도방정식 우측 첫번째 값

(21) 우도방정식 우측값 계산.

$$\sum_{k=1}^{n}\left[\frac{(y_k - y_{k-1})(t_k^2 e^{-bt_k} - t_k^2 e^{-bt_{k-1}})}{(1+bt_{k-1})e^{-bt_{k-1}} - (1+bt_k)e^{-bt_k}}\right]$$

(22) 우도방정식 좌측값 계산.

$$\frac{y_n t_n^2 e^{-bt_n}}{[1 - (1+bt_n)e^{-bt_n}]}$$

(23) 우도방정식 좌우측 차이

(24) 이분법 종료 후 파라미터 추정치 $\hat{a}$, $\hat{b}$ 값 계산

(25) 추정된 파라미터 b값을 대입

(26) 파라미터 a값의 추정치 계산.

$$\hat{a} = \frac{y_n}{[1 - (1+bt_n)e^{-bt_n}]}$$

4.4 도구실행 및 분석

구현된 지연 S자형 신뢰도 평가도구를 다음 표 4에 나타낸 에러발견수 데이터를 가지고서 실행하여 결과를 나타낸다[12]. 표 4에 나타낸 테스트 데이터는 참고문헌[12]에서 사용된 데이터로 분석대상의 데이터는 약 3000 LOC (Line of Code)로 된 온라인 단말제어 프로그램(어셈블리어)에 대한 온라인 기능 테스트 데이터로 캘린더 시간으로 10일간의 실측데이터이다.

표 4. 에러발견수 데이터

Table 4. Number of detected error data

테스트시각(日)	발견된 에러수	총발견누적에러수
1	1	1
2	2	3
3	8	11
4	4	15
5	3	18
6	3	21
7	2	23
8	4	27
9	3	30
10	1	31

표 4의 데이터로 구현된 평가도구를 실행한다. 평가도구를 실행시키면 다음의 메뉴가 화면에 나타낸다. 이때 〈ENTER〉키 또는 마우스로 클릭하면 된다.

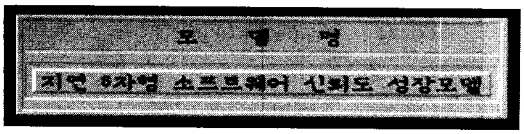


그림 7의 평가도구 메뉴를 실행시키면 두 번째 메뉴로 자동으로 넘어가게 되며 실제의 테스트 데이터를 다음의 메뉴에서 선택한다. 본 논문에서는 임의로 표 4의 데이터인 MHS2.DAT를 선택한다.

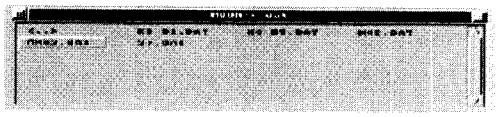


그림 8. 테스트 데이터 선택메뉴

Fig 8. Option menu of test data

테스트 데이터를 선택하면 모델 파라미터값 및 평균치 함수를 계산한 후 모델에 대한 적합도 계산과정을 다음과 같은 메뉴에서 선택한다.

이때는 위험율을 1% 또는 5%를 임의로 선택할 수 있다. 본 논문에서는 1%를 선택하였다.

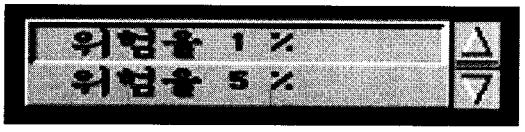


그림 9. 위험율 선택메뉴

Fig 9. Option menu of hazard rate

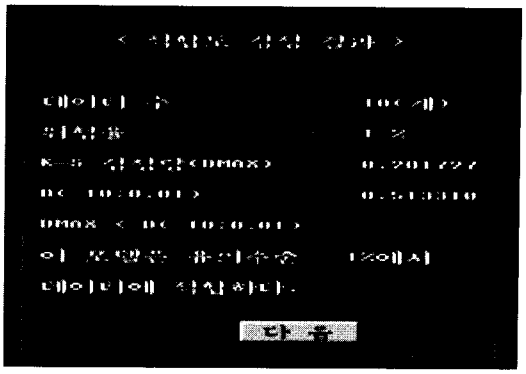


그림 10. 적합도 계산 결과

Fig 10. Result about test of goodness of fit and result

그림 10는 위험율 1% 선택시 표 4의 테스트 데이터는 지연 S자형 모델에 적합함을 화면에 나타내 주고 있다. 최종적으로 그림 11의 메뉴에서 계산된 평가 결과를 출력 또는 그래프로 출력할 수 있다.

그림 11. 평가결과 출력메뉴

Fig 11. Output menu of evaluation result

그림 11에서 결과 메뉴를 선택하면 그림 12과 같이 신뢰성 평가결과를 화면에 나타내며 그래프 메뉴를 선택하면 평균치함수, 소프트웨어 내의 잔존 에러수, 소프트웨어 신뢰도를 그래프로 나타낸다. 그림 12의 결과를 분석해 보면 실제의 테스트 데이터는 위험율 1% 즉, 90%의 신뢰한계에서 지연 S자형 모델에 적합함을 알수 있으며 결과로서 테스트 시작전에 잠재해 있는 총기대 에러수가 약 38개이며 소

프트웨어 내에 잔존에러 발견율은 0.31699이며 테스트에 의해서 발견되지 않는 기대 잔존 에러수는 약 7개이며 또한 K-S 검정량과 편차자승합은 각각 0.20173과 12.16694임을 알 수 있다.

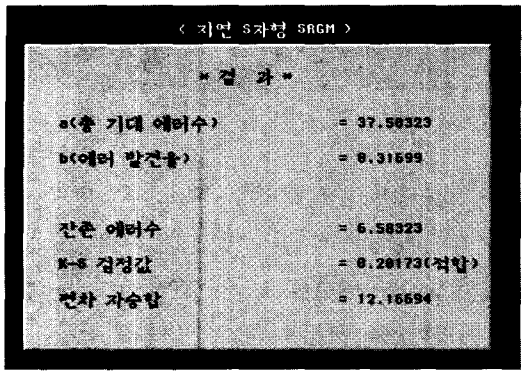


그림 12. 신뢰성 평가결과

Fig 12. Result of reliability evaluation

그림 13는 그림 11의 평가결과 메뉴에서 그래프 메뉴를 선택했을 때의 소프트웨어 내의 잔존에러수를 그래프로 나타내고 있다. 이때 가로 축은 테스트시각(日)을 나타내고 세로축은 누적 에러수를 나타낸다. 기타 평균치 함수, 신뢰도를 〈ENTER〉키 또는 마우스를 클릭함으로써 그래프로 화면에 출력할 수 있다.

5. 결론 및 향후 연구

소프트웨어 신뢰도는 아직 초기단계에 있다. 소프트웨어 개발과정의 마지막 단계인 테스트에서 얻어지는 에러발견수 데이터로 정량적인 소프트웨어 품질평가를 하기위해 SRGM 중에서 비교적 경험적으로 중,소 규모의 소프

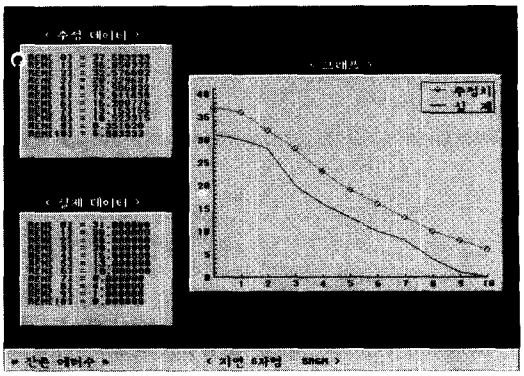


그림 13. 잔존 에러수 그래프

Fig 13. Graph about number of errors remaining

트웨어 개발시에 유리한 지연 S자형 SRGM의 복잡한 수리적 개념을 고찰하고 정립하여 알고리즘을 구현하고 신뢰성 데이터 분석순서를 정립하여 모델에 포함된 파라미터값, 적합도검정, 편차자승합 등을 자동으로 산출함 으로서 대상의 소프트웨어 평가척도인 총기대 에러수, 에러발견율, 잔존에러수, K-S검정, 편차자승합, 신뢰도에 이르는 데이터를 얻을수 있다. 실제로 평가도구를 실행한 결과 참고문헌[12]에서의 지연 S자형 모델의 평가결과와 같음을 알 수 있었다. 향후 연구과제 로서는 구현된 지연 S 자형 모델에 많은 실제 데이터를 적용하여 평가 타당성 검토를 계속하여 검증해 보는 것이 필요하며 또한, 지연 S자형 모델의 가정들을 계속하여 개선이 필요하다.

참 고 문 헌

[1] Jeliski,Z. and Moranda,P.B.: "Software reliability research", in Statistical Computer Performance Evaluation, Freiberger, W. (ed.), pp.465-484, Academic Press, 1972.

[2] Musa,J.D.: "A theory of Software reliability and its application", IEEE Trans. Software Engineering, Vol.SE-1, No3, pp.312-327, 1975.

[3] Capers Johnes : Applied Software Measurement : Assuring Productivity Quality, pp.279～338., 1991.

[4] Goel,A.L.: "Software reliability Models: Assumptions, limitations, and applicability", IEEE Trans. Software Engineering, Vol. SE-11, No.12, pp.1411-1423, 1985.

[5] Shooman, M.L.:Software Engineering: Design, Reliability, and Management, McGraw-Hill, 1983.

[6] Xie,M.: Software Reliability Modeling, World Scientific, 1991.

[7] Goel, A.L. and Okumoto, K : "Time dependent error-detection rate model for Software Reliability Reliability and other performance measures", IEEE Trans. Reliability, Vol.R-28, No.3, pp.206-211, 1979.

[8] Yamada,S., Ohba,M.and Osaki,S.:"S shaped-reliability growth modeling for Software error deteection", IEEE Trans. Reliability, Vol.R-32, No.5, pp.475-478, 484, 1983

[9] Ohba,M., "Inflection S shaped software Reliability growth model in stochastic models in reliability theory, Osaki, S and Hatayama, Y", pp.144-162, Springer-Verlag, Berl in, 1984.

[10] 山田茂: "ソフトウェアの品質評價に關する考え方と動向-ソフトウェア信賴度成長モチルに基づく定量的品質評價法-", 情報處理, Vol.32, No.11 pp.1189-1202, 1991

[11] 高宗雄, "ソフトウエア信賴度成長モデルに基づく定量的 品質評價法", 九州大學學位論文, 1989.

[12] M. ohba: "Software quality = Test coverage × Test accuracy" Proc. 6th Int. Conf. Software Engineering, pp.287-293(1982).

□ 著者紹介

문 외 식

1974년 ~ 1980년 울산대학교 공과대학 전자계산학과 졸업(공학사)
1983년 ~ 1986년 부산대학교 산업대학원 전자계산학과 졸업(공학석사)
1993년 ~ 1995년 경남대학교 공과대학 전자계산학과 박사과정 수료
(소프트웨어공학 전공)
1981년 ~ 1984년 8월 한국전력공사 전자계산소 근무
1984년 9월 ~ 현재 창원전문대학 사무자동화과 부교수

※ 관심 분야 : 소프트웨어공학(소프트웨어 신뢰도 평가), UNIX 시스템