S/W 로지스틱 테스트 노력함수의 적정성에 관한 연구

최규식*
*건양대학교 의공학과
e-mail:che@konyang.ac.kr

A Study on the Reasonability of Logistic Testing Efforts on S/W

Che Gyu Shik*
*Dept of Biomedical Eng., Konyang University

요 약

소프트웨어 개발 후 인도 전 테스트 단계중에 발생되는 테스트 노력 소요량을 고려한 소프트웨어 신뢰도 성장 모델을 제시하여 테스트 노력소요량 동태를 시간함수인 로지스틱 곡선으로 설명한다. 그러므로, 본 논문에서는 로지스틱 테스트노력 곡선이 소프트웨어의 개발/테스트 노력곡선으로 적절하게 표현될 수 있다는 것과 실제 데이터를 근거로 하여 적용하여서 예측성이 매우 좋은 능력을 가지고 있다는 것을 보이고자 한다.

1. 서 론

소프트웨어 신뢰도는 고객의 입장에서 본 소프트웨어의 품질 관점이다. 신뢰도와 비용을 고려한 연구로서 Okumoto와 Goel은 전체 평균 소프트웨어 비용을 최소화시키는 비용-최적 SRP를 발표하였다. Yamada와 Osaki는 전체 평균 비용을 최소화시키고 소프트웨어 신뢰도를 만족시키는 전체평균비용-신뢰도-최적 SRP를 도입하였다. 이러한 연구결과를 참조하여 Hou, Kuo, Chang은 지수 곡선과 로지스틱 곡선에 적용하는 연구를 수행하였다.

본 논문에서는 로지스틱 곡선으로서 테스트 노력의 시간종속 거동을 설명한다.

2. 테스트 노력 함수에 대한 이론적 배경

2.1 웨이불형

Yamada가 제안한 웨이불형 테스트 노력 함수에 의하면 소프트웨어의 테스트 노력이 테스트 단계 전체에 걸쳐서 일정하다고 가정해서는 안된다고 한다. 순간적인 테스트 노력이 결국은 테스트 수명주기 동안 감소한다. 여러 관련문헌에서는 테스트 노력이 웨이불형 분포로 설명될 수 있고, 아래와 같은 3개의 케이스를 가진다는 것을 보여주고 있다.

1) 지수함수곡선 : (0, t]에서 소모되는 누적 테스트 노력은

$$W(t) = N[1-\exp(-\beta t)] \tag{2.1}$$

로서 웨이불함수의 m=1인 경우에 해당된다.

2) 레일레이 곡선 : 소모되는 누적 테스트 노력은

$$W(t) = N[1-\exp(-\frac{\beta}{2} \cdot t^2)] \tag{2.2}$$

로서 웨이불함수의 m=2인 경우에 해당된다.

3)웨이불 곡선 : 소모되는 누적 테스트 노력은

$$W(t) = N[1-\exp(-\beta t^m)] \tag{2.3}$$

웨이불형 곡선(2.3)에 대해서 m=1 또는 m=2일 때 그 결과는 각각 지수함수 곡선이거나 레일레이 곡선이다. 따라서, 이들은 웨이불 함수의 특수한 경우에 해당된다. m=3, 4, 5일 때는 소프트웨어 개발 공정 기간 동안 공칭 피크 현상(원활하지 못하게 증가하며, 소모 곡선을 열화시킴)을 나타낸다. 그러므로, 웨이불형은 테스트 노력 소모량을 모델링하는 데는 부적합한 것으로 사료된다.

2.2 로지스틱형

실제 테스트 노력 데이타가 여러 가지 소모 패턴을 나타내므로 때때로 테스트 노력 비용을 지수함수나 레일레이 곡선만으로 설명하기는 어렵다. 그 대안으로 제시된 것이 로지스틱형 테스트노력 함수이다. 이 함수는 실제 프로젝트 탐사에 의해서 보고된 바와 같이 매우 정확하다. (0, t]에서의 누적 테스트 노력 소모는

$$W(t)) = \frac{N}{1 + A \cdot \exp(-\alpha t)} \qquad (2.4)$$

이다. 현재의 테스트 노력 소모량은 W(t)의 미분치

$$w(t) = \frac{dW(t)}{dt} = \frac{NA\alpha \cdot \exp(-\alpha t)}{[1 + A \cdot \exp(-\alpha t)]^2}$$

$$= \frac{NA\alpha}{[\exp(\alpha \frac{t}{2}) + A \cdot \exp(-\frac{\alpha t}{2})]^2} \qquad (2.5)$$

와 같이 표현된다.

3. 로지스틱 테스트노력

기본 SRGM은 아래와 같은 가정 하에 적용한다.

1) 결함검출 공정이 NHPP를 따른다.
2) 소프트웨어 시스템은 시스템에 잔존하는 결함에 의하여 발생되는 무작위 고장에 의존한다.
3) w(t)에 의하여 (t, t+Δt]동안에 검출되는 결함의 평균수는 잔여하는 결함의 평균치에 비례한다.
4) 비례상수는 시간에 따라 변하지 않는다.
5) 테스트 노력은 로지스틱 함수로 모델링한다.
6) 고장이 발생할 때마다 결함은 새로운 결함 도입 없이 즉시 그리고 완벽하게 제거된다.
7) 결함 제거 시간은 무시할 정도이며, 검출된 오류는 완벽하게 제거된다.

3.1 소프트웨어 신뢰도 척도

평균치 함수 m(t)를 가진 NHPP 모델에 의해서 소프트웨어 신뢰도 평가에 대한 두 가지 정량적 평가 척도를 얻을 수 있다. 시각 t에서의 신뢰도는 다음과 같다.

$$R(x \mid t) = \exp[-m(t+x) + m(t)] \qquad (3.1)$$

3.2 로지스틱형 성장모델

통계적인 예측에 의하여 현재의 결함 내용은 어느 경우이든 유한하므로 m(t)는 t에 대한 증가함수이며 m(0)=0이다. 이러한 가정 하에

$$\frac{m(t+\Delta t) - m(t)}{w(t)} = r[a - m(t)] \cdot \Delta t \qquad (3.2)$$

이며, 현재의 테스트 노력 비용에 의해서 검출되는 결함의 수가 잔여결함의 수에 비례하면

$$\frac{dm(t)}{dt} \cdot \frac{1}{w(t)} = r[a - m(t)] \qquad (3.3)$$

과 같이 방정식을 세울 수 있다.
즉 식(3.3)은 현재의 테스트 노력에 의해서 검출되는 결함의 수를 나타낸다.
경계조건 m(0)=0인 조건에서 (3.3)을 풀면

$$m(t) = a(1 - e^{-r[W(t) - W(0)]})$$

$$= a\{1 - \exp[-r \cdot W^*(t)\} \qquad (3.4)$$

이고, 고장강도는

$$\lambda(t) = \frac{dm(t)}{dt} = arw(t) \cdot \exp[-rW^*(t)]$$

$$= \frac{arNA\alpha}{\left[e^{\frac{\alpha t}{2}} + Ae^{-\frac{\alpha t}{2}}\right]^2} exp\left[\frac{N}{1 + Ae^{-\alpha t}} - \frac{N}{1+A}\right] \qquad (3.5)$$

이다. 그러므로, 테스트를 무한시간 계속한다면 검출되지 않는 평균 결함수는

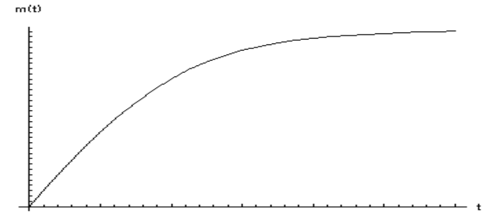

그림 1. 평균치함수

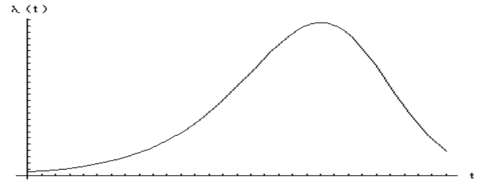

그림 2. 고장강도함수

$$a - m(\infty) = a \cdot \exp\left[-r \cdot \frac{NA}{1+A}\right] \qquad (3.6)$$

이다. 결국 테스트 단계 기간 동안에 쓰이는 노력이 N으로 제한되므로 유한 테스트 노력으로는 소프트웨어 시스템 내에 존재하는 모든 결함을 검출할 수는 없다는 것을 알 수 있다.

4. 인도 시각

목표 신뢰도를 맞추면서 전체 소프트웨어 비용이 최소가 되도록 하는 것이 중요하다. 그림 3에서는 테스트 기간을 횡축으로 하여 비용과 신뢰도의 관계를 동시에 나타낸 것이다. 최적 소프트웨어 인도문제를 다음과 같이 정의한다.

$c_2 > 0$, $c_1 > c_3 > 0$, $x \geq 0$, $0 < R_0 < 1$인 경우에 대해서
$R(x|T) \geq R_0$, $T \geq 0$인 조건하에 $C(T)$를 최소화 (4.1)

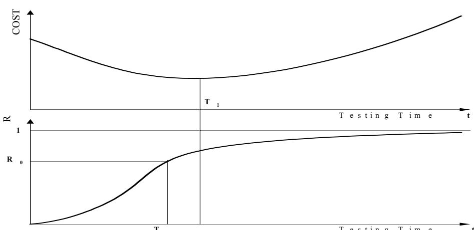

그림 3. 비용-신뢰도 관계 곡선

이러한 방법으로 하여 비용-신뢰도최적 소트웨어 인도시각에 대한 해를 구할 수 있다.

$$T^* = \max\{T_1, T_2\} \qquad (4.2)$$

4.1 비용 기준에 의한 인도 시각

비용 기준에 의해서 산출된 소프트웨어 비용을 이용한 소프트웨어 테스트/개발 단계 기간 동안의 테스트 노력 비용과, 인도 전후의 오류 수정 비용은 다음과 같다.

$$C(T) = C_1 m(T) + C_2[m(T_{LC}) - m(T)]$$

$$+ C_3 \int_0^T w(x)dx \qquad (4.3)$$

C_2는 보통 C_1의 크기보다 크기 때문에 $C_1 < C_2$이다.

최적값(비용 최저값)을 구하기 위해 방정식(4.3)을 T에 관하여 미분한다.

$$\frac{d}{dT}C(T) = C_1\frac{dm(T)}{dT} - C_2\frac{dm(T)}{dT} + C_3 w(T)$$
$$= w(T)\{-(C_2 - C_1)ar \cdot \exp[-rW^*(T)] + C_3\} \qquad (4.4)$$

이 식에서 비용이 최소가 되는 시각을 구하면

$$T_1 = \frac{1}{\alpha}log\frac{N - \frac{1+A}{r}log\frac{C_3}{ar(C_2 - C_1)}}{N + \frac{1+A}{Ar}log\frac{C_3}{ar(C_2 - C_1)}} \qquad (4.5)$$

으로서 이 관계를 그림으로 나타내면 그림 4와 같다.

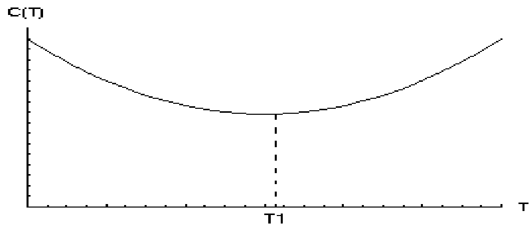

그림 4. 인도시각 - 비용 관계 곡선

$$0 < \frac{1}{\alpha}\log\frac{N - \frac{1+A}{r}\log\frac{C_3}{ar(C_2 - C_1)}}{N + \frac{1+A}{Ar}\log\frac{C_3}{ar(C_2 - C_1)}} < T_{LC}$$

인 범위 즉

1) $\exp\left[-\frac{Ar(e^{\alpha T_{LC}} - 1)}{(1+A)(A+e^{\alpha T_{LC}})}\right] \leq \frac{C_3}{ar(C_2 - C_1)} \leq 1$

에서 유일하고도 유한한 양의 해 $T^* = T_1$이 존재하고 그 값은 (4.5)와 같다.

2) $\frac{C_3}{ar(C_2 - C_1)} > 1$ 이면 $T_1 < 0$이므로 $T^* = T_1 = 0$이다.

3) $\frac{C_3}{ar(C_2 - C_1)} < \exp\left[-\frac{Ar(e^{\alpha T_{LC}} - 1)}{(1+A)(A+e^{\alpha T_{LC}})}\right]$

이면 $T_1 > T_{LC}$이므로 $T^* = T_1 = T_{LC}$이다.

이와 같은 내용을 그림 5에 표시하였다.

$$\exp\left[-\frac{Ar(e^{\alpha T_{LC}} - 1)}{(1+A)(A+e^{\alpha T_{LC}})}\right] \leq \frac{C_3}{ar(C_2 - C_1)} \leq 1$$

이면 비용최저점이 결함시험과 소프트웨어의 전 수명기간 사이에 존재하는 경우이다.

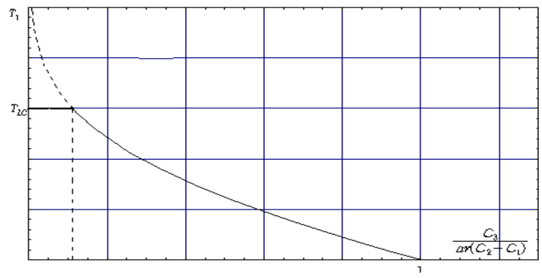

그림 5. 조건별 인도시각

$$\frac{C_3}{ar(C_2 - C_1)} < \exp\left[-\frac{Ar(e^{\alpha T_{LC}} - 1)}{(1+A)(A+e^{\alpha T_{LC}})}\right]$$ 인

조건은 비용이 단조감소하는 경우이다. 이러한 경우는 결함시험을 하면 할수록 총 비용이 감소된다. 이러한 경우는 소프트웨어의 결함시험에 의해서 신뢰도가 높아지고, 따라서 드물게 결함이 검출되어도 그 수정비용이 미미한 경우에 해당된다.

주어진 조건에서 MLE와 LSE를 이용하여 관련 파라미터를 구하고 이 파라미터를 이용하여 방정식(4.5)으로부터 비용 최적 인도시각 T_1을 결정한다.

4.2 신뢰도 기준에 의한 인도 시각

시각 t에서 마지막 고장이 발생한 후 조건 확률함수는 아래와 같으며, 여기서 평균치함수 $m(t)$는 방정식(3.4)와 같다.

$$R = R(x|t) = \exp\{-[(m(t+x) - m(t)]\}$$
$$= \exp\{-a[\exp(-rW^*(t)) - \exp(-rW^*(t+x))]\} \qquad (4.6)$$

위의 방정식으로부터

$$\log R(x|0) = -m(x) = -a(e^{-rW^*(0)} - e^{-rW^*(x)})$$
$$= -a\left\{1 - e^{-r(\frac{N}{1+Ae^{-\alpha x}} - \frac{N}{1+A})}\right\}$$
$$R(x|0) = \exp\left[-a\left(1 - e^{-r(\frac{N}{1+Ae^{-\alpha x}} - \frac{N}{1+A})}\right)\right] \qquad (4.7)$$

과, 또 이 결과로부터

$$e^{-\alpha x} = \frac{1}{A}\left\{\frac{1+A}{1 - \frac{1+A}{rN}log[1 + \frac{1}{a}logR(x|0)]} - 1\right\} \qquad (4.8)$$

을 얻는다. 그리고, (4.6)에 대수를 취한다.

목표신뢰도를 R_0 라하고 위의 식을 정리하면

$$\log R_0 = \log R(x|T) = -a(e^{-W^*(T)} - e^{-W^*(T+x)})$$
$$\log\frac{1}{R_0} = ae^{\frac{rN}{1+A}}\left[e^{-\frac{rN}{1+Ae^{-\alpha T}}} - e^{-\frac{rN}{1+Ae^{-\alpha(T+x)}}}\right] \qquad (4.9)$$

를 만족하는 시각 T_2를 구한다.

여기서, N, A, α, β는 최소자승법으로 구한다.

$$S(N, A, \alpha) = \sum_{k=1}^{n}[W_k - W(t_k)] \qquad (4.10)$$

W_k =(0, t] 기간동안 실제로 소요되는 누적테스트노력이고, $W(t_k)$ =테스트노력함수에 의해서 산출된 누적 테스트노력이다.

방정식(4.9)를 풀어서 원하는 R_0 에 이르는 시간을 결정한다. 방정식(4.9)를 이용하여 목표신뢰도에 이르는 신뢰도를 얻는데 필요한 테스트시간을 얻거나 또는 주어진 시간간격 동안 신뢰도가 목표치에 도달하는지 아닌지를 결정할 수 있다.

주어진 조건에서 MLE와 LSE를 이용하여 관련 파라미터를 구하고 이 파라미터를 이용하여 방정식(4.9)로부터 비용 최적 인도시각 T_2를 결정한다.

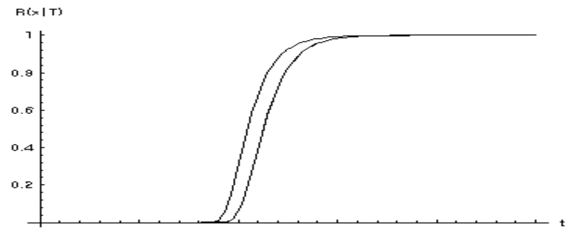

그림 6. 인도시각과 신뢰도와의 관계

5. 실례

5.1 비용

4.1항의 조건 1)에 해당하는 경우로서 방정식(4.5)를 이용하여 그 시각을 구하면 T_I=20.04가 되고 방정식(4.3)을 이용하여 최저비용을 구하면 C=1,548.8이 된다. 이 때의 신뢰도는 방정식(4.6)을 이용하여 이 때의 신뢰도를 구하면 0.5453으로서 목표신뢰도와 거리가 멀다.

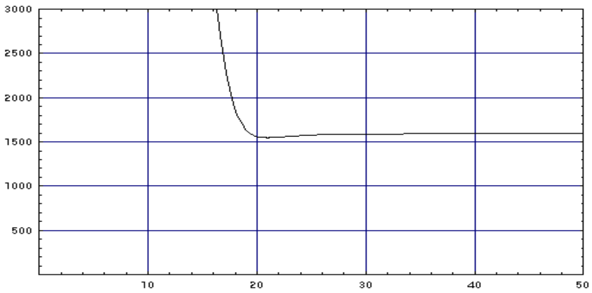

그림 7. 비용곡선

5.2 신뢰도

방정식(4.9)와 방정식(4.10)을 이용하여 목표신뢰도를 만족하는 시각을 구하면 T=26이 된다.
따라서, (4.2)의 조건에 따라 $T^*=T_2$=26이 되어야 하며, 이 때의 총 비용은 1,584.4로서 최저비용보다 35.6만큼 증가된다.
이 예에서는 인도시간이 비용최저에 의해서 결정된 것이 아니고 목표신뢰도를 맞추는 시점에서 결정되었다.

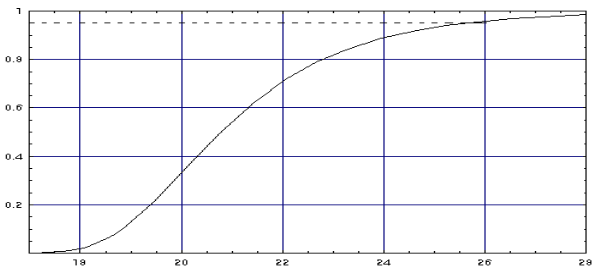

그림 10. 신뢰도 곡선

6. 결 론

개발소프트웨어의 테스트노력함수로서 로지스틱함수를 채택하여 개발소프트웨어의 비용과 신뢰도를 동시에 만족시키는 기법을 연구하고자 하였다.

$\frac{C_3}{ar(C_2-C_1)} > 1$ 이면 비용이 단조증가하는 경우로서 결함시험을 하면 할수록 비용이 증가되어 시험없이 인도하는 것이 최적인 것을 의미한다.

$$\exp\left[-\frac{Ar(e^{\alpha T_{LC}}-1)}{(1+A)(A+e^{\alpha T_{LC}})}\right] \le \frac{C_3}{ar(C_2-C_1)} \le 1$$

이면 비용최저점이 결함시험과 소프트웨어의 전 수명기간 사이에 존재하는 경우이다. 이러한 경우는 본 논문이 추구하고자 하는 이상적인 경우로서 목표신뢰도를 만족시키는 인도시기와 총 비용을 최저로 하는 인도시기 중 큰 값을 취하는 것이 이상적이다.

$\frac{C_3}{ar(C_2-C_1)} < \exp\left[-\frac{Ar(e^{\alpha T_{LC}}-1)}{(1+A)(A+e^{\alpha T_{LC}})}\right]$ 인 조건은 비용이 단조감소하는 경우이다. 이러한 경우는 결함시험을 하면 할수록 총 비용이 감소된다.

또한, 신뢰도 기준에 의한 인도수법을 고찰한다. 일반적으로 소프트웨어 인도시간문제는 소프트웨어 시스템의 신뢰도와 연동되어 있다. 소프트웨어 시스템의 신뢰도가 적정한 수준에 이르면 소프트웨어를 인도한다. 마지막 고장이 발생한 후 조건 확률함수를 이용하여 목표신뢰도를 만족시키는 시간을 결정하는 방정식을 제시하였다. 이 방정식을 풀어서 원하는 R_0 에 이르는 시간을 결정한다.

참고문헌

[1] C. V. Ramamoorthy, F. B. Bastani, "Software reliability - Status and perspectives", IEEE Trans. on Software Eng., vol. SE-8, pp354-371, 1982 Aug.

[2] J. D. Musa, A. Iannino, K. Okumoto, "Software Reliability : Measurement, Prediction, Application", pp230-238, 1987 Mar.

[3] S. Yamada, H. Ohtera, H. Narihisa, "Software reliability growth models with testing - efforts", IEEE Trans. Reliability, vol. R-35, pp19-23, 1986 Apr.

[4] H. Ascher, H. Feigold, "Repairable Systems Reliability : Modeling, Inference, Misconceptions, and Their Causes", 1984, Marcel Dekker

[5] K. Okumoto, A. L. Goel, "Optimum release time for software systems based on reliability and cost criteria", J. System software, vol. 1, pp315-318, 1980.

[6] S. Yamada, S. Osaki, "Cost-reliability optimal release policies for software systems", IEEE Trans. on Reliability, vol. R-34, pp422-424, 1985 Dec.

[7] Rong-Huei Hou, Sy-Yen Kuo, Yi-Ping Chang, "Optimal release policy for hyper-geometric distribution software-reliability growth model", IEEE Trans. on Reliability, vol.45, pp646-651, 1996 Dec.

[8] Chin-Yu Huang, Sy-Yen Kuo, "Analysis of Incorporating Logistic Testing-Effort Function into Software Reliability Modeling", IEEE Trans. on Reliability, vol.51, pp261-270, 2002, Sep.