결함도입을 고려한 개발 소프트웨어의 최저비용 산출에 관한 연구

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

요약

소프트웨어 결함은 그것을 찾아내는 것도 힘들지만 정확한 해법을 찾는 것도 쉽지 않을 뿐더러, 또 테스트자의 능력 여하에 따라 수정중에 새로운 결함이 도입될 수도 있기 때문에 검출된 결함이 완벽하게 제거되기는 쉽지 않다. 따라서, 결함 제거 효율은 개발중인 소프트웨어의 신뢰도 성장이나 테스트 및 수정비용에 영향을 크게 미친다. 이는 소프트웨어 개발의 모든 과정에서 매우 유용한 척도로서 개발자가 디버깅 효율을 평가하는데 크게 도움이 될 뿐더러, 추가로 소요되는 작업량을 예측할 수 있게 해준다. 그러므로 개발 소프트웨어의 신뢰도와 비용면에서 불완전 디버깅의 영향을 연구하는 것은 매우 중요하다고 할 수 있으며, 이는 최적 인도 시각이나 운영 예산에도 영향을 줄 수 있다.

본 논문에서는 개발중인 소프트웨어를 대상으로 하여 디버깅이 완전하지 않으며, 이 때문에 디버깅 중 새로운 결함이 도입될 수도 있다는 제안하에 보편적으로 사용되는 신뢰도 모델을 대상으로 불완전 디버깅 범위로까지 소프트웨어의 신뢰도와 비용 문제를 확장하여 연구한다.

키워드 : SRGM, 평균치 함수, 결함 검출비, 결함 도입 확률, 목표 신뢰도, 최적 인도 시간

1. 서론

소프트웨어 결함 디버깅은 매우 복잡한 공정이다. 검출된 결함에 대하여 이를 해결하기 위해 여러 가지 테스트(유니트 테스트, 집적 테스트, 시스템 테스트) 과정을 거쳐야 한다. 어떤 경우에는 이러한 테스트를 거치지 않을 수도 있고 때로는 이러한 테스트를 거쳤다 하더라도 그 테스트 환경이 고객의 환경과 동일하지 않아서 결함이 완벽하게 제거되지 않을 수도 있다. 한편, 그러한 과정중에 새로운 결함이 도입될 수도 있다. 소프트웨어 개발팀이 문제의 결함이 최종적으로 제거되기 전에 여러 번 보고된 소프트웨어 결함이라는 것을 발견하는 것도 흔한 일이다. 어떤 결함들은 고객이 현장에서 사용할 때만 나타나는 것도 있다. 그러므로, 결함 제거 효율은 소프트웨어 신뢰도 계산에서 중요한 인자이고 소프트웨어 사업관리에도 중요하다.

또한, 소프트웨어 개발에는 많은 개발자원들이 소요된다. 소프트웨어 테스트 단계 기간 동안에는 소프트웨어의 신뢰도가 내재 결함을 검출 및 수정하는데 소요되는 개발자원의 양에 크게 의존한다. 테스트 노력은 CPU 시간의 양, 실행된 테스트 케이스의 수 등으로 나타낼 수 있다. 때때로 테스트 시간은 실행시간 대신 테스트의 수로 나타낼 수도 있다. 소프트웨어 신뢰도 모델링 분야에서는 소프트웨어의 개발 노력이 자주 전통적인 지수함수, 레일레이함수, 또는 웨이불 곡선으로 표현된다. 또, 실제 테스트/디버그 데이터 집합에 근거하여 실험을 수행하여 다양한 모델간의 예측 능력을 비교해본 결과, 초기 결함의 수를 산출하는 데는 로지스틱 테스트 노력 함수를 가진 SRGM[1]이 이전의 접근법에 비하여 더 적합하다는 것을 보여주고 있다.

본 논문에서는 결함 제거 효율과 결함 개념을 소프트웨어 신뢰도 성장 모델에 도입시키는 방법을 제안한다. 이는 소프트웨어 개발시 디버깅이 완벽하다고 가정하여 소프트웨어의 신뢰도를 평가했던 그동안의 논문들[2,3]과 다른 개념이다.

2. 소프트웨어 신뢰도 모델

2.1 평균치 함수 및 신뢰도

소프트웨어 신뢰도는 규정된 환경 하에서 주어진 시간에 소프트웨어를 결함 없이 운영할 수 있는 확률인 것으로 정의하며, 다음과 같이 조건확률로 표현할 수 있다.

테스트공정이 NHPP를 따르고 테스트 노력이 일정한 경우, NHPP의 표준 이론[4,5]으로부터 평균치 함수를

$$m(t) = a(1 - e^{-bt}) \tag{2.1}$$

로 정의할 때 임의의 $t \geq 0$과 $x > 0$에서

$$\Pr\{N(t+x) - N(t) = k\} = \frac{[m(t+x) - m(t)]^k}{k!} \exp\{-[m(t+x) - m(t)]\} \tag{2.2}$$

이므로, 소프트웨어의 신뢰도는 다음과 같이 표현할 수 있다.

$$R(x|t) \equiv \Pr\{N(t+x) - N(t) = 0\}$$
$$= \exp\{-[m(t+x) - m(t)]\}$$
$$= \exp[-a(1-e^{-bx})e^{-bt}] = \exp[-m(x)e^{-bt}] \quad (2.3)$$

즉, 어느 시각 t부터 $(t+x)$ 시각까지 새로운 결함이 발견되지 않을 확률을 신뢰도로 정의하며, 이는 평균치 함수의 차이를 지수함수의 지수로 취한 형태를 하고 있다.

3. 소프트웨어의 수정 비용

소프트웨어를 주문 받아서 개발한 후, 발행 전 결함을 발견하기 위한 테스트를 수행하여, 신뢰도가 어느 목표치에 도달했을 때 발행하게 된다. 그리고, 발행 후 결함이 발견되면 이를 수정해야 하며, 그 각각의 과정에서 비용이 발생하게 된다. 이러한 비용을 소프트웨어의 수정비용이라 하며, 여기에는 다음과 같은 비용이 발생된다.

테스트 기간중의 결함 수정 비용은 검출된 결함 하나하나를 수정하는데 비용이 발생되므로, 테스트 기간중에 검출되는 총 결함의 수에 결함당 수정비용을 곱한 값이 된다[6].

$$c_1 m_1(T) = c_1 a(1-e^{-b_1 T}) \quad (3.1)$$

운영중에 검출되는 결함의 수정비용은 발행 후 수명이 끝나는 시점까지 발생되는 결함에 대해서 수정하는데 드는 비용이므로

$$c_2\{m_2(T_{LC}) - m_1(T)\} = c_2 a e^{-b_1 T}(1-e^{-b_2(T_{LC}-T)}) \quad (3.2)$$

와 같이 표현된다.

테스트 기간 중에 발생되는 비용은 단위시간당 테스트 비용을 결함 제거 확률과 결함 도입 확률을 고려한 값으로 나누어 전 테스트기간의 시간을 곱한 값이 된다.

참고문헌[2]에서는 이 식의 계수를 $\frac{c_3}{1-p}$라 제안하고 있으나 이는 문제가 있는 것으로 사료된다. 즉, 디버깅 확률이 클수록 테스트 기간중에 발생되는 테스트 비용이 증가되며, 디버깅 확률이 완벽하여 p=1인 경우에는 테스트 비용이 무한적으로 증가한다는 것이다. 이로 미루어 볼 때 이 식이 의미하는 것은 테스트 기간중에 훈련을 통하여 디버깅 확률을 목표 수준까지 높이려 노력할 때 발생되는 비용을 고려한 것으로 사료된다.

따라서, 본고에서는 결함 도입 확률 β까지를 고려하여 그 비용 계수를

$$\frac{c_3}{p-\beta} \quad (3.3)$$

로 제안한다.

이는 디버깅 확률이 높으면 높을수록 테스트가 수월해지며, 따라서 테스트 기간도 단축시키고 테스트 비용도 줄어들 것어 테스트 기간중의 테스트 비용이 감소한다는 개념이 타당성을 갖는다고 본 것이다.

상기와 같은 논리에 의해서 총 비용은 테스트 기간중의 결함 수정 비용, 운영 기간중의 결함 수정 비용, 테스트 기간중의 테스트 비용을 합한 값이 된다.

$$C(T, p, \beta) = c_1 m_1(T) + c_2\{m_2(T_{LC}) - m_1(T)\}$$
$$= -(c_2 - c_1)m_1(T) + c_2 m_2(T_{LC}) + \frac{c_3}{p-\beta} W^*(T) \quad (3.4)$$

이다. 여기서, $m(t)$는 평균치함수이며, b_1, b_2는 각각 개발소프트웨어의 인도전후의 결함검출비이다. 최적 소프트웨어 발행시각은 전체 평균 소프트웨어 비용을 최소로 하는 테스트 시간이다. 각 테스트 노력에 대한 비용을 산출하는 공식과 최적 인도시각을 유도하면 아래와 같다.

가. 테스트 노력이 일정할 경우

- 비용함수

$$C(T, p, \beta) = -\frac{a(c_2 - c_1)}{p-\beta}[1-e^{-(p-\beta)b_1 T}]$$
$$+\frac{ac_2}{p-\beta}[1-e^{-(p-\beta)b_2 T_{LC}}] + \frac{c_3}{p-\beta} T \quad (3.5)$$

- 최적 인도 시각

$C(T, p, \beta)$의 최적값을 구하기 위해 식(3.5)를 T로 편미분하여 정리한다.

$$-a(c_2 - c_1)b - 1e^{-(p-\beta)b_1 T} + \frac{c_3}{p-\beta} = 0 \quad (3.6)$$

특별히 소프트웨어 발행 전후의 결함검출비율이 같을 경우에는 $b_1 = b_2 = b$가 되어 수정비용에 대한 해석이 단순해진다.

이 식을 만족시키는 $T>0$인 범위의 T값을 구하면 $C(T)$에 대한 최소값이 된다.

$$T = \frac{1}{(p-\beta)b} \ln \frac{ab(c_2 - c_1)(1-p+\beta)}{c_3} \quad (3.7a)$$

여기서, $T^*=T_2$는 소프트웨어 인도 전후의 총 수정비용이 최소가 되는 시간이다.

참고로 기존의 문헌에서처럼 p=1, β=0인 경우에는

$$T = \frac{1}{b} \ln \frac{ab(c_2 - c_1)}{c_3} \quad (3.7b)$$

와 같이 표현된다.

나. 지수함수인 경우

- 비용함수

$$C(T,p,\beta) = -\frac{a(c_2-c_1)}{p-\beta}\{1-e^{-(p-\beta)rN(1-e^{-(p-\beta)bT})}\}$$
$$+\frac{ac_2}{p-\beta}\{1-e^{-(p-\beta)rN(1-e^{-(p-\beta)bT_{LC}})}\}$$
$$+\frac{c_3rN}{p-\beta}(1-e^{-(p-\beta)bT}) \quad (3.8)$$

- 최적 인도 시각

$$T = -\frac{1}{(p-\beta)b} \cdot \ln\left[1-\frac{1}{(p-\beta)rN}\ln\frac{ab(c_2-c_1)(p-\beta)}{c_3}\right] \quad (3.9)$$

다. 레일레이 함수인 경우

- 비용함수

$$C(T,p,\beta) = -\frac{a(c_2-c_1)}{p-\beta}\{1-e^{-(p-\beta)rN(1-e^{-(p-\beta)b/2T^2})}\}$$
$$+\frac{ac_2}{p-\beta}\{1-e^{-(p-\beta)rN(1-e^{-(p-\beta)b/2T^2_{LC}})}\}$$
$$+\frac{c_3rN}{p-\beta}(1-e^{-(p-\beta)b/2T^2}) \quad (3.10)$$

- 최적 인도 시각

$$T = \left\{-\frac{2}{(p-\beta)b} \cdot \ln\left[1-\frac{1}{(p-\beta)rN}\ln\frac{ab(c_2-c_1)(p-\beta)}{c_3}\right]\right\}^{1/2} \quad (3.11)$$

라. 로지스틱 함수인 경우

- 비용함수

$$C(T,p,\beta) = -\frac{a(c_2-c_1)}{p-\beta}\left\{1-e^{-(p-\beta)rN\left(\frac{1}{1+Ae^{-\alpha T}}-\frac{1}{1+A}\right)}\right\}$$
$$+\frac{ac_2}{p-\beta}\left\{1-e^{-(p-\beta)rN\left(\frac{1}{1+Ae^{-\alpha T_{LC}}}-\frac{1}{1+A}\right)}\right\}$$
$$+\frac{c_3N}{p-\beta}\left\{\frac{1}{1+Ae^{-\alpha T}}-\frac{1}{1+A}\right\} \quad (3.12)$$

- 최적인도시각

$$T_2 = -\frac{1}{\alpha}\ln\frac{1}{A}\left[\frac{1}{\frac{1}{1+A}+\frac{1}{(p-\beta)rN}\ln\frac{a(c_2-c_1)(p-\beta)}{c_3}}-1\right] \quad (3.13)$$

그림 3.1은 상기와 마찬가지로 참고문헌[2][3]에 의해 제시된 a=142, b=0.1246, x=0.1, p=0.7, β=0.012, c_1=\$200, c_2=\$500, c_3=\$100, T_{LC} = 500인 경우에 대해서 결함 제거 확률과 결함 도입 확률을 고려한 경우와 그렇지 않은 경우의 비용을 비교한 그래프이다. 실선은 결함 제거가 완전한 경우이고 점선은 불완전한 경우이다.

마찬가지로 이 그림에 의하면 결함제거 효율이 완전하여 결함발견 즉시 수정이 완벽하고 결함 수정중 새로운 결함이 도입될 가능성이 없는 경우에는 비용이 최저로 되는 인도시간은 31.9이고 이 때의 비용은 \$32,390이다. 그러나, 결함제거 효율을 가정하여 그 값이 70%라 하고 따라서 결함 도입 확률을 1.2%로 가정한 경우에는 비용을 최저로 하는 시간이 42.0이 되고 비용은 \$49,075로서 결함수정이 완벽한 경우에 비하여 크게 증가된다.

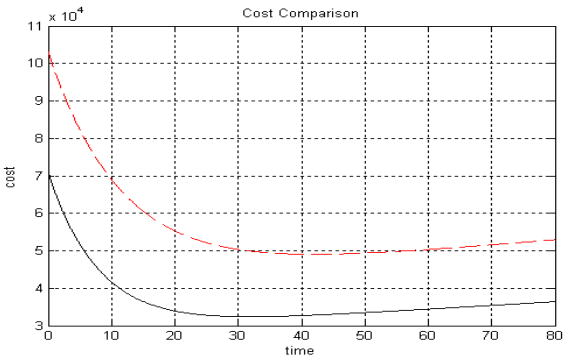

그림 3.1 수정비용 비교

4. 적용 실례

참고문헌[2][3]에서 인용한 데이터는 a=142, b_1=b_2=b=0.1246, R_o=0.95, β=0.012, C_1=\$200, C_2=\$500, C_3=\$100, T_{LC} = 500이며, 이를 근거로 하여 각 결함 제거 확률에 대한 결과를 표 3.1에 나타내었다. 결함도입확률이 0.012로서 일정하다고 가정하고 결함제거효율이 변화함에 따라 최적 발행 시각과 신뢰도, 수정비용이 어떻게 변하는지를 고찰해본다. 결함제거효율이 낮아짐에 따라 목표신뢰도에 이르는 시간은 증가비가 선형적이나 최저 비용에 이르는 시간은 반비례의 관계를 유지하여 목표신뢰도 시간에 비하여 크게 증가하고 이는 검출확률이 낮아짐에 따라 이러한 현상은 급격해진다.

참고로 동일 데이터에 의해서 p=1, β=0인 경우 즉, 결함 검출시 결함제거가 완벽하고 새로운 결함이 도입되지 않는 경우에 대해서 기존의 방식대로 계산해 보면 T_1=28.4, T_2=31.9로서 T^*=31.9이고, $R(x|T)$=0.968, $C(T, p, \beta)$=\$32,390으로서 결함제거효율이 불완전한 것을 감안하고 또한 디버깅중에 결함 도입가능성을 고려한 상기 고찰결과에 비하여 낙관적인 결과가 나온 것을 알 수있다.

표 4.1 각각의 결함 제거 확률에 대한 영향

p	0.9	0.8	0.7	0.6	0.5
T_1	32.0	36.0	41.3	48.3	58.2
T_2	34.8	38.0	42.0	47.0	53.5
T^*	T_2=34.8	T_2=38.0	T_2=42.0	T_1=48.3	T_1=58.2
$R(x\|T)$	0.963	0.960	0.953	0.950	0.950
$C(\$)$	36,921	42,159	49,075	58,618	70,123

5. 결 론

소프트웨어의 개발 단계에서 테스트 및 수정단계를 거칠 때 실제로 결함 제거 효율은 통상 불완전하며, 이러한 사실은 그동안 널리 알려져 있다. 소프트웨어 결함은 그것을 찾아내는 것도 힘들지만 수정중에 새로운 결함이 도입될 수도 있기 때문에 검출된 결함이 완벽하게 제거되기는 어렵다. 따라서, 결함 제거 효율은 개발중인 소프트웨어의 신뢰도 성장이나 테스트 및 수정비용에 영향을 크게 미친다. 이는 소프트웨어 개발의 모든 과정에서 매우 유용한 척도로서 개발자가 디버깅 효율을 평가하는데 크게 도움이 될 뿐더러, 추가로 소요되는 작업량을 예측할 수 있게 해준다. 그러므로 개발 소프트웨어의 SRGM과 비용면에서 불완전 디버깅의 영향을 연구하는 것은 매우 중요하다고 할 수 있으며, 이는 최적 인도 시각이나 운영 예산에도 영향을 줄 수 있다.

본 논문에서는 소프트웨어의 디버깅이 완전하지 않으며, 이 때문에 디버깅중 새로운 결함이 도입될 수도 있다는 제안 하에 보편적으로 사용되는 신뢰도 및 비용모델을 불완전 디버깅 범위로 확장하여 연구하였다. 그간의 기존 논문들을 참고하여 결함 제거 확률과 수정중 결함도입 확률을 고려한 이론을 전개 및 수립하였다. 이러한 알고리즘을 확인 및 실증하기 위해 그간 몇 개의 참고문헌에서 수집된 실제 데이타에 의해서 소프트웨어의 결함 제거 확률을 변화시켰을 때의 각각에 대해서 목표신뢰도에 이르는 시간, 최저 수정 비용에 이르는 시간을 계산하였다. 그리고, 최적시간을 산출하고 이때의 신뢰도 및 총비용을 계산하였다.

본 논문에서는 소프트웨어의 결함 제거가 불완전하다고 가정하고 수정중 결함 도입 확률까지를 고려하여 목표신뢰도와 최저 비용 시간을 구했으나, 여기서 제시된 알고리즘으로서 기존의 결함 제거 확률이 완벽한 경우의 비용을 산출할 수 없어서 이 때는 기존의 방법을 사용할 수밖에 없다는 단점이 있다. 이러한 문제는 추후 연구를 통하여 해결되어야 할 것으로 사료된다.

감사의 글

이 논문은 2005학년도 건양대학교 학술연구비 지원에 의하여 이루어진 것임

참고 문헌

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

[2] Min Xie, Bo Yang, " A study of the effect of imperfect debugging on software development cost", IEEE Trans. on Software Eng., vol.29, no.5, pp471-473, 2003.5

[3] X. Zhang, X. Teng, " considering fault removal efficiency in software reliability assessment", IEEE Trans. on Systems, man, and cybernetics, vol.33, no.1, pp114-120, 2003.1

[4] A.L. Goel, K. Okumoto, "A Markovian model for reliability and other performance measures of software systems", Proc. AFIPS Conf., pp770-774, 1979.6

[5] W. Kremer, "Birth-death and bug counting", IEEE Trans. on Reliability, vol.R-32, no.1, pp37-47, 1983