N개 버전 시스템용 소프트웨어 신뢰도 성장모델

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

SRGM for N-Version Systems

Che Gyu Shik*
*Dept of Information and Communication, Konyang University

요약

본 논문에서는 NHPP에 근거한 N 버전 프로그래밍 시스템의 SRGM을 제안한다. 비록 많은 연구 논문에서 NVP, 시스템 신뢰도에 대해서 연구노력을 기울여 왔지만 그들 대부분이 안정된 신뢰도에 대해서만 고려해 왔다..

테스트 및 디버깅 동안 결함이 발견되면 디버깅 노력은 결함을 제거하는데 집중된다. 소프트웨어가 너무 복잡하므로 이러한 결함을 성공적으로 제거한다는 것이 쉽지 않으며, 또 다른 새로운 결함이 소프트웨어에 도입될 수도 있다. 일반화된 NHPP 모델을 NVP시스템에 적용하여 새로운 NVP-SRGM이 수립된다. 제어시스템에 대한 단순화된 소프트웨어 제어에서 이러한 새로운 소프트웨어 신뢰도 모델을 어떻게 적용하는지를 보여주고 있다. 소프트웨어 신뢰도평가에 s-신뢰도 구간을 준비하였다. 이 소프트웨어 신뢰도 모델은 신뢰도를 평가하는데 쓰일 수가 있어서 NVP 시스템의 성능을 예측하는데 쓰일 수 있다. 일반적인 산업사회에 적용하여 상용화하기 위해서는 내결함 소프트웨어의 신뢰도를 정량화하기 위해 제안된 NVP-SRGM을 충분히 인증하는데 좀더 적용이 필요하다.

NVP 신뢰도 성장 모델링을 하는 이러한 종류의 첫 모델로서 제안된 NVP-SRGM은 독립 신뢰도 모델의 단점을 극복하는데 쓰일 수 있다. 이는 독립적인 모델보다 더욱 더 정확하게 시스템 신뢰도를 예측할 수 있으며, 언제 테스트를 중단해야 하는가를 결정하는 데에도 쓰일 수 있으며, 이는 NVP 시스템 개발 수명주기 단계를 테스트 및 디버깅함에 있어서 핵심 질문사항이다.

1. 서론

내결함(fault-tolerant) 소프트웨어에 대해서 그동안 발표된 많은 논문들은 1970년대 이후로 N-버전 프로그래밍 시스템의 소프트웨어 신뢰도를 모델링하여 평가하는데 집중하였다. 일부 신뢰도 모델들은 소프트웨어 다양성 모델에 초점을 맞추고 이들은 다양화된 소프트웨어에서 세부적으로 종속성을 분석하는데 근거를 두고 있다. 기타 다른 모델들은 우선적으로 특별한 형태의 소프트웨어 시스템에 대한 종속성 척도를 평가하는데 목적을 두고 내결함 소프트웨어 시스템 거동을 모델링하는데 초점을 두었다.

본 논문에서는 NVP 시스템에 대한 새로운 SRGM을 제안하며 테스트 및 디버깅 기간 동안의 오류 제거 효율과 오류 도입율을 고려한다. NVP 소프트웨어 신뢰도 모델을 개발하는데 일반화된 NHPP 모델을 사용한다.

2. 공통 결함과 독립 결함

다양한 소프트웨어 버전이 상이한 규격, 설계, 프로그래밍 팀, 프로그래밍 언어 등을 이용하여 개발되지만 많은 연구가들이 그러한 독립적으로 개발된 소프트웨어들이 예를 들면 독립적으로 고장날 필요는 없다는 것을 밝혀냈다. 어떤 실험에서는 상이한 언어와 설계개념을 이용하는 것이 NVP에서의 신뢰도에 영향을 주지 못한다는 것을 보여주고 있다.

관계 결함과 관계되지 않은 결함을 구분하기가 어려워서 본 논문에서는 이러한 결함분류를 다음과 같이 한다.

- 적어도 2개의 버전이 동일하지만 모두가 잘못된 결과를 나타낼 때는 그 고장이 그들 버전간의 관계된 결함에 의해서 발생된 것이다.
- 적어도 2개 이상의 버전이 유사하지 않으나 잘못된 결과를 나타낼 때는 그 결함이 관계가 없거나 독립적인 소프트웨어 결함에 의해서 발생된 것이다.

단순하게 하기 위하여 본 논문의 나머지 부분에서는 "관계결함"을 "공통결함"으로 칭한다. 그림 1에서는 2VP에서의 공통결함 및 독립결함을 예시하고 있다.

공통고장은 적어도 2개 이상의 소프트웨어 버전 중에서 기능적으로 등가인 모듈에 위치한다. 이는 비록 그것들이 독립적으로 상이한 버전을 개발한다 하더라도 그 프로그래머들이 동일 또는 유사한 실수를 하는 경향이 있기 때문이다. 이러한 결함들은 동일 입력에 대해서 쉽게 활성화되어 그러한 버전들이

동시에 고장나도록 한다. 그리고 공통결함에 의한 이러한고장들은 공통고장이라 부른다.

독립 결함은 상이한 소프트웨어 버전 사이 또는 그 가운데에서 상이한 또는 기능적으로 등가가 아닌 모듈에 위치한다. 그들이 서로간 독립적이기 때문에 그리고 내결함 시스템(fault-tolerant system)에 해가 되지 않으므로 그 결과로 생기는 고장은 통상적으로 결정 메카니즘에서 구분할 수 있다. 그러나, 공통 고장에 비하여 극히 적기는 하지만 예기치 못한 입력이 상이한 버전에서 독립적인 두 개의 결함을 일으킬 확률이 있다. 표 1에서는 공통고장과 동시 독립고장의 차이를 보여준다.

표 1. 공통고장 및 동시독립 고장

구 분	공통고장	동시 독립 고장
결함형태	공통결함	s-독립 결함
출력	통상동일	통상 상이
결함위치	동일	상이
보팅결과	잘못된 해법 선택	정확한 해를 선택할 수 없음

그러므로, 본 논문에서는 NVP 시스템의 소프트웨어 고장이 두 개의 모드를 가지는 것으로 가정한다. 공통고장모드와 s-독립 고장모드이다. NVP 소프트웨어 신뢰도는 이러한 두 개의고장모드 결합이다.

3. NVP-소프트웨어 신뢰도 성장 모델

동일 입력에서 s-독립결함에 의해 적어도 2개의 버전이 고장날 때 동시 s-독립고장이 발생된다. 즉, A, B, C이며 AB, AC,.가 아니다. 그림 2에서는 다양한 결함형태와 그들의 관계를 보여주고 있다.

동시 입력에서 s-독립 결함에 의해서 적어도 2개 이상의 버전이 고장날 때 동시 s-독립고장이 발생된다. 즉, AB, AC, BC가 아니라 A, B, 또는 C이다.

3.1 일반화된 NHPP 신뢰도 모델

가정

1) 소프트웨어 시스템은 소프트웨어 내의 결함에 의해서 실행중에 고장날 수 있다.

2) 소프트웨어 고장 발생은 NHPP를 따른다. NHPP에 기초한 기존의 여러 SRGM은 실제의 소프트웨어 공학에 있어서 매우 성공적인 수단으로 실증되어 왔다.

3) 어느 순간이든 소프트웨어 고장 검출율은 그 시각에 소프트웨어 내에 잔존하는 결함의 수에 비례한다.

4) 소프트웨어 고장이 발생되면 디버깅 노력을 즉시 시작한다. 이 노력으로 확률 p로 상응하는 결함을 즉시 제거하며, $p \gg 1-p$이다. 일부 새로운 결함이 소프트웨어 시스템에 도입될 수 있으며 이 확률은 $\beta(t)$이며, $\beta(t) \ll p$이다. 이 디버깅은 소프트웨어 고장의 각 위치에서 s-독립이다.

가정 #3.1로부터 방정식 (1)을 유도한다.

$$m'(t) = b(t) \cdot [a(t) - p \cdot m(t)],\ a'(t) = \beta(t) \cdot m'(t) \quad (1)$$

m(0)=0, a(0)=a라 하자. m(t)와 a(t)는 (1)의 미분방정식을 풀어서 구한다.

$$m(t) = a \cdot \int_0^t b(u) \cdot \exp\left[- \int_0^u (p - \beta(\tau)) \cdot b(\tau) d\tau \right] du$$

$$a(t) = a \cdot \left[1 + \int_0^t \beta(u) \cdot b(u) \cdot \exp\left[- \int_0^u (p - \beta(\tau)) \cdot b(\tau) d\tau \right] du \right] \quad (2)$$

가정

1) N=3

2) (보팅 전에) 끝내기 위해 빠른 버전이 가장 늦은 버전을 기다려야 함

3) 각 소프트웨어 버전은 소프트웨어 내의 결함에 의해서 실행중에 고장이 발생될 수 있다.

4) 동일 입력에 의해서 두 개 이상의 소프트웨어 버전이 고장을 일으킬 수 있다. 이는 상이한 버전에서 공통 결함에 의하거나 또는 s-독립 결함에 의해서 발생할 수 있다.

5) 상이한 결함(s-독립 결함, 2-버전 공통 결함, 또는 3-버전 공통 결함)에 의해서 발생되는 소프트웨어 고장은 NHPP를 따른다.

6) 어떤 시각에서건 소프트웨어 고장 검출율은 그 시각에 소프트웨어 내에 잔존하고 있는 결함의 수에 비례한다.

7) 이러한 3-버전 어느 것에서 발생되면 디버깅 노력을 즉시 시작한다. 이 노력으로 확률 p_i로 상응하는 결함을 즉시 제거하며, $p_i \gg 1-p_i$이다. (i는 버전 1, 2, 3을 나타낸다.)

8) 각각의 디버깅 노력에 대하여 결함이 성공적으로 제거되었건 아니건 일부 새로운 s-독립 결함이 확률 β로 소프트웨어 시스템에 도입될 수 있으며, $\beta_i \ll p_i$이다. 그러나, 그 시스템에 새로운 공통 결함은 도입되지 않는다.

9) 일부 공통 결함은 제거 노력을 성공적으로 수행하지 못하여 어떤 낮은 공통 결함이나 s-독립 결함 수준으로 감소될 수도 있다.

10) 모든 종류의 결함(A, B, C, AB, AC, BC, ABC)에 대한 오류 검출율은 동일하고 상수이다. 즉 b(t)=b이다.

11) 동시 s-독립 고장은 상이한 버전 사이의 s-독립 결함이 활성화됨에 따라 발생된다. 그리고 Pr{동시 s-독립 고장이 3개의 버전을 포함함}=0이다. 이러한 고장들은 두 개의 버전만을 포함한다.

12) 버전 사이의 어떠한 잔여 s-독립 결함쌍이라도 어떤 입력에 의해서 활성화되는 동일 확률을 갖는다.

13) 어떤 2개의 버전을 포함하는 동시 s-독립 고장의 강도는 그 두 개의 버전에 있는 잔여 s-독립 결함의 쌍에 비례한다.

3.2 시스템 신뢰도

NVP는 그의 2개 버전의 반 이상이 동시에 고장나면 시스템이 고장난다. 보터나 결정 메카니즘은

완벽한 것으로 가정한다. 다른 곳에서 언급한 바와 같이 이러한 고장들은 2개의 범주를 가지고 있다.
- 공통고장모드
- s-독립 고장

NVP 시스템 신뢰도는 이 2개의 고장모드를 결합하여 얻을 수 있다.

1) 공통 고장 모드 : 공통 결함 계수 공정 $N_{AB}(t)$, $N_{AC}(t)$, $N_{BC}(t)$, $N_{ABC}(t)$을 얻은 후에 새로운 공정을 정의한다.

$$N_d(t) \equiv N_{AB}(t) + N_{AC}(t) + N_{BC}(t) + N_{ABC}(t) \quad (3)$$

Nd(t)는 시스템에서 공통 결함에 의해서 발생되는 NVP 소프트웨어 시스템 고장의 수를 계수한다.

$$m_d(t) \equiv m_{AB}(t) + m_{AC}(t) + m_{BC}(t) + m_{ABC}(t) \quad (4)$$

NVP 내결함 소프트웨어(fault-tolerant software) 시스템이 시각 T에서 최근 고장이 난 후 (T, T+x) 기간 동안 고장나지 않을 확률은

$$\Pr\{x \text{ 동안 고장나지 않음} | T\} = \exp[-(m_d(T+x) - m_d(T))] \quad (5)$$

2) s-독립 고장 : 보통 NVP 소프트웨어 시스템 고장은 버전 중의 공통 결함에 의하여 발생한다. 그러나, 그것이 s-독립 결함이므로 3개의 소프트웨어 버전이 동시에 고장날 확률은 매우 낮다.

#3.2의 가정 12와 13으로부터 버전 1과 2 사이의 동시 s-독립 고장에 대한 $h_{\overline{AB}}(t)$는

$$h_{\overline{AB}}(t) = K_{AB} \cdot X_A(t) X_B(t)$$

이고, 버전 1과 2의 s-독립 결함의 수는

$$X_A(t) = a_A(t) - p_1 m_A(t)$$

$$X_B(t) = a_B(t) - p_2 m_B(t)$$

이고, 이제 다음과 같은 값을 가지는 동시 s-독립고장 $N_{\overline{AB}}(t)$에 대한 또 다른 NHPP가 있다.

$$m_{\overline{AB}}(t) = \int_0^t h_{\overline{AB}}(\tau)\, d\tau \quad (6)$$

주어진 발행 시간 T 및 실행시간 x에서 x 시간 동안 동시 s-독립 고장 $\overline{AB}$가 발생하지 않을 확률은

$$\Pr\{x \text{ 동안 } \overline{AB} \text{ 고장나지 않음} | T\} = \exp[-(m_{\overline{AB}}(T+x) - m_{\overline{AB}}(T))] \quad (7)$$

이며, 이와 유사하게 버전 1과 3, 그리고 버전 2와 3 사이에도 다음과 같은 평균치 함수를 가지는 동시 s-독립 고장에 대한 2개의 다른 NHPP가 있다.

$$m_{\overline{AC}}(t) = \int_0^t h_{\overline{AC}}(\tau)\, d\tau \quad (8)$$

$$m_{\overline{BC}}(t) = \int_0^t h_{\overline{BC}}(\tau)\, d\tau \quad (9)$$

주어진 발행시간 T 및 실행 시간 x에서 x 시간 동안 동시 s-독립고장 $\overline{AC}$ 와 $\overline{BC}$가 발생하지 않을 확률은

$$\Pr\{x \text{ 동안 } \overline{AC} \text{ 고장나지 않음} | T\} = \exp[-(m_{\overline{AC}}(T+x) - m_{\overline{AC}}(T))] \quad (10)$$

$$\Pr\{x \text{ 동안 } \overline{BC} \text{ 고장나지 않음} | T\} = \exp[-(m_{\overline{BC}}(T+x) - m_{\overline{BC}}(T))] \quad (11)$$

$$h_{\overline{AC}}(t) = K_{AC} \cdot X_A(t) X_C(t)$$

$$h_{\overline{BC}}(t) = K_{BC} \cdot X_B(t) X_C(t)$$

이다. 이제 새로운 계수공정을 정의한다.

$$N_I(t) = N_{\overline{AB}}(t) + N_{\overline{AC}}(t) + N_{\overline{BC}}(t)$$

$$m_I(t) = m_{\overline{AB}}(t) + m_{\overline{AC}}(t) + m_{\overline{BC}}(t) \quad (12)$$

NVP 시스템이 시각 T에서 최근 고장이 난 후 (T, T+x) 기간 동안 고장나지 않을 확률은

$$\Pr\{x \text{ 동안 동시독립고장이 나지 않음} | T\} = \exp[-(m_I(T+x) - m_I(T))] \quad (13)$$

3) NVP 시스템 신뢰도 :

공통 고장 및 동시 s-독립 고장 확률을 얻은 후에 공통 고장과 동시 s-독립 고장이 상호간에 s-독립이므로

$$R_{NVP}(x|T) = \Pr\{x \text{ 기간동안 공통고장이 없음} | T\} \cdot \Pr\{x \text{ 기간동안 동시 } s-\text{독립고장이 없음} | T\}$$

그러면

$$R_{NVP}(x|T) = \exp[-(m_d(T+x) + m_I(T+x) - m_d(T) - m_I(T))] \quad (14)$$

이다.

4. 파라미터 산출

4.1 미지 파라미터

많은 파라미터들이 NVP-SRGM에 있으며, 그들은 반드시 산출해내야 한다. MLE가 이들을 산출하는데 쓰인다. 문제를 단순화시키기 위해서 결함 검출 효율 p_1, p_2, p_3가 알려져 있다고 가정하자. 아래의 파라미터들은 산출되어야 한다.

b, a_{ABC}, a_{AB}, a_{AC}, a_{BC}, a_A, β_2, β_3, a_B, a_C, β_1, K_{AB}, K_{AC}, K_{BC}

1) 형태 ABC, AB, AC, BC, A, B, C 고장에 대해서 주어진 시간까지의 누적 고장수

2) 주어진 시간까지의 동시 s-독립 고장의 누적수

4.2 MLE

약어 $y_{z,i} \equiv Z$=A, B, C, AB, AC, BC, ABC에 대해서 시간 t_i까지 검출된 형태 Z결함의 누적총수

i=1, 2, …, n_Z이다.

결함형태 A에 대한 가능성함수는

$$L_A = \prod_{i=1}^{n_A} \left[\frac{[m_A(t_i) - m_A(t_{i-1})]^{y_{A,i} - y_{A,(i-1)}}}{(y_{A,i} - y_{A,i-1})!} \right] \cdot \exp[-(m_A(t_i) - m_A(t_{i-1}))] \quad (15)$$

$$\log(L_A) = \sum_{i=0}^{n_A} [(y_{A,i} - y_{A,i-1}) \cdot \log(m_A(t_i) - m_A(t_{i-1})) - (m_A(t_i) - m_A(t_{i-1})) - \log(y_{A,i} - y_{A,i-1})!]] \quad (16)$$

이와 유사하게 다른 결함 형태 B, C, AB, AC, BC, ABC에 대해서도 얻을 수 있다.

5. 각종 케이스

5.1 케이스 1 : s-독립 NVP-SRGM

이러한 2개의 소프트웨어 버전은 상호간에 s-독립이다. 그러므로, 일반화된 소프트웨어 신뢰도 모델을 각 버전에 별도로 적용하여 $R_1(x|t)$와 $R_2(x|t)$를 산출한다. 다음과 같은 방정식을 이용하여 전체 시스템의 신뢰도를 산출한다.

$$R_{ind}(x|t)=1-[1-R_1(x|t)]\cdot[1-R_2(x|t)],$$
$$R_j(x|t)=\exp[-(m_j(t+x)-(m_j(t))],\ j=1,\ 2 \quad (17)$$

그림 1과 2에서는 j=1, 2에 대한 각각의 $m_j(t)$의 적합성을 보여주고 있다.

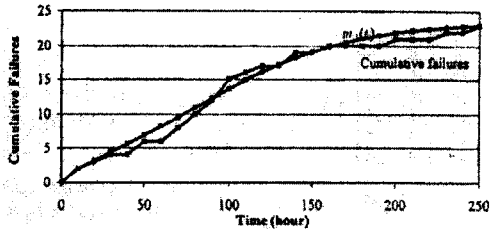

그림 1. 버전 1의 누적 고장수에 대한 $m_1(t)$

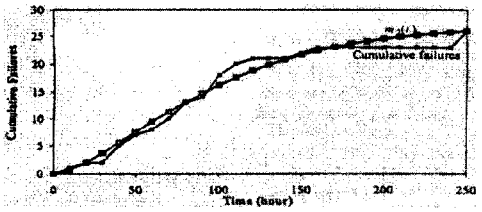

그림 2. 버전 2의 누적 고장수에 대한 $m_2(t)$

5.2 케이스 2 : 종속 NVP-SRGM

동시에 예를 들면 t=8.4, 20, 28, …,, 99.6에 2개의 버전이 동시에 고장 나는 것을 보여주고 있다. 이들은 2 개의 버전 사이에 공통 결함에 의해서거나 또는 관계 없는 결함에 의해서 발생되는 "동시 고장"인 것으로 사료된다. 그러므로, 이 정규화 데이터 집합에는 s-독립이라는 가정이 합당하지 못하다.

이 예에서는 모든 동시 고장이 공통 고장인 것으로 가정한다. 그러므로, 이 2VP 시스템에서의 소프트웨어 결함은 기호법에 따라서 분류할 수 있다.

1)결함형태 AB :
해는 다음과 같다.

$$m_{AB}(t)=\frac{a_{AB}}{p_1p_2}\cdot[1-\exp(-bp_1p_2t)] \quad (18)$$

2) 결함형태 A :
해는 다음과 같다.

$$m_A(t)=C_{A1}+C_{A2}\cdot\exp(-bp_1p_2t)$$
$$+C_{A3}\cdot\exp[-b(p_1-\beta_1)t] \quad (19)$$

3) 결함형태 B :
해는

$$m_B(t)=C_{B1}+C_{B2}\cdot\exp(-bp_1p_2t)$$
$$+C_{B3}\cdot\exp[-b(p_2-\beta_2)t] \quad (20)$$

이다.

4) 일반사항

$L=L_AL_BL_{AB}$

Φ=A, B, AB에 대하여

결함형태 A에 대한 가능성함수는

$$L_\Phi=\prod_{i=1}^{n_\Phi}\left[\frac{[m_\Phi(t_i)-m_\Phi(t_{i-1})]^{y_{\Phi,i}-y_{\Phi,(i-1)}}}{(y_{\Phi,i}-y_{\Phi,i-1})!}\right] \quad (21)$$

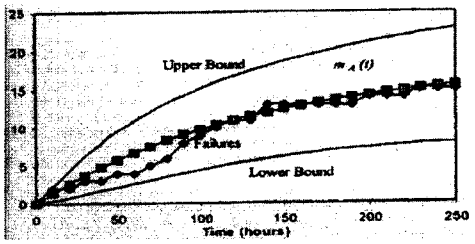

그림 3. 형태 A 고장의 누적에 대한 $m_A(t)$

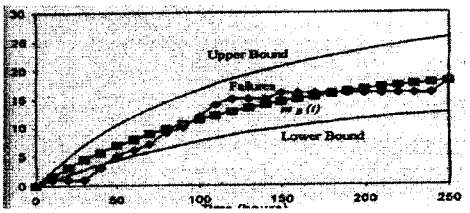

그림 4. 형태 B 고장의 누적에 대한 $m_B(t)$

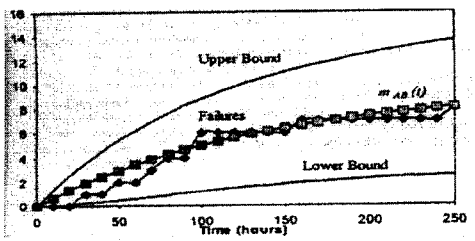

그림 5. 형태 AB 고장의 누적에 대한 $m_{AB}(t)$

감사의 글

본 연구는 한국과학재단
목적기초연구(R01-2000-000-00273-0) 지원사업으로
수행되었음

참고문헌

[1] F. Belli and P. Jedrzejowicz, "Fault-tolerant programs and their reliability", IEEE Trans. Reliability, vol.29,no.2, pp184-192, 1990

[2] L. Chen and A. Avizienis, "N-version programming: A fault tolerance approach to the reliable software", in Proc. 8th Int. Symp.pp3-9, 1978

[3] J. B. Dugan and M. R. Lyu, "System reliability programming application", IEEE Trans. Reliability, vol.43,no.4, pp513-519, Dec. 1994