중복모줄의 소프트웨어의 신뢰도

최규식
che@konyang.ac.kr
건양대학교 의공학과

reliability of software with redundancy modules

Che Gyu-Shik
Konyang University

요약

내고장성(fault - tolerance)을 향상시키면서 소프트웨어의 신뢰도를 최적화하는 두 가지 접근법으로서 수정복구 블록 스킴과 N-버전 프로그래밍 기법이 있다. 본 연구에서는 중복모줄을 가지고 소프트웨어의 신뢰도를 극대화하기 위한 하나의 알고리즘으로서 수정복구 스킴을 제안한다. 개발소프트웨어의 최적구조를 결정하는 기법을 수립하기 위해 최적화 공정 결과를 적용한다. 주어진 가용자원 범위 내에서 정규실행모줄과 테스트모줄의 신뢰도 및 확률을 계산하여 소프트웨어 시스템의 신뢰도를 극대화하기 위한 소프트웨어의 형상을 고찰한다. 간단한 예로서 한 개의 경우를 예시한다.

1. 서론

소프트웨어 신뢰도는 소프트웨어 품질의 가장 중요한 척도이며, 다분히 고객 지향적이다. 소프트웨어 신뢰도는 소프트웨어가 얼마나 사용자 요건에 맞도록 작동하는가 하는 것을 척도하는 것이다. 소프트웨어 개발자들은 제한된 자원(시간과 비용) 하에서 소프트웨어를 개발해야 하기 때문에 신뢰도와 비용간의 상호 비교분석을 통하여 최적 방안을 찾고자 한다. 소프트웨어 신뢰도와 비용은 두 개의 상충되는 과제로서 한 쪽을 향상시키려면 다른 쪽을 희생시켜야 한다. 따라서, 제한된 자원하에서 신뢰도를 극대화하기 위한 비용-신뢰도 최적화문제가 중요한 과제로 대두된다. 목표신뢰도를 만족시키면서 비용을 최소화하기 위한 연구는 극소수의 연구자에 의해서 수행된 바 있으며, 제한된 자원 하에서 신뢰도를 극대화 시키기 위한 여러 연구가 있었으며, 수많은 노력에도 불구하고 신뢰도 최적화 분야는 아직까지 어떠한 범용적이고도 실제적인 결과를 이끌어내지 못하였다. 기존의 모델/절차는 어떤 특수한 상황 하에서 적용 가능하거나 극히 초보적인 간단한 실제의 경우였을 뿐이다.

본 논문에서는 다중성을 가진 소프트웨어에 대해서 최적구조에 이르도록 하기 위해 기존의 소프트웨어 신뢰도모델의 예측적인 방법을 제시한다. 이 접근법은 구성품으로부터 복잡하고도 내고장성인 소프트웨어의 구조가 구축되는 구성품의 신뢰도 예측, 각각의 구성품을 개발하고 구현하는데 필요한 자원의 양 산출, 최적소프트웨어 구조의 계산을 얻기 위한 기존의 신뢰도모델을 일부 적용한다.

2. 신뢰도 최적화 모델

고찰하고자 하는 소프트웨어 시스템에서 동작하는 모델이 모줄단위로 직렬로 구성된 것으로 가정한다. 각각의 모줄은 확장 복구스킴인 것으로 생각하여 다중성을 적용하여 신뢰도를 극대화하고자 한다. 이러한 가정 하에 소프트웨어 시스템 내의 각 모줄에 대한 최적 다중 레벨을 발견하고자 하는 것이다. 모줄에 대한 개념은 그림 1에 나와 있다.

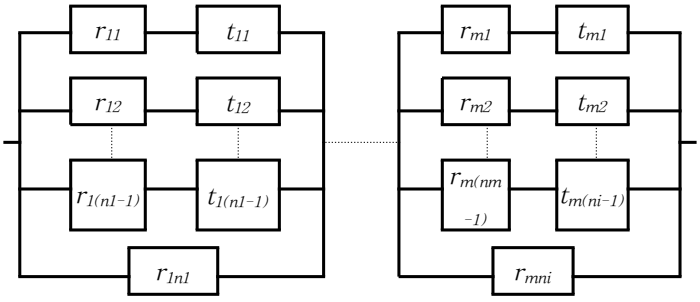

그림 1. 소프트웨어시스템의 모줄

각각의 모줄은 n_i개의 병렬 선로로 구성도며, 하나의 선로는 두 개의 파트로 구성된다. 첫 번째 파트는 모줄 명령을 수행하는 정규모줄이다. 두 번째 파트는 테스트모줄이다. 정규모줄은 소프트웨어의 모줄기능을 수행하며, 테스트모줄은 1) 정규모줄이 에러 없이 기능을 수행했는지를 테스트하고, 2) 정규모줄 수행상에 에러가 발견되면 모줄기능을 처음상태로 복구하고 실행을 다음 모줄로 전환하는 기능을 한다. 테스트 모줄 자체도 장애를 일으킬 수가 있는데 장애를 일으키는 경우는 1) 모줄이 에러가 있어서 잘못 수행되었으나, 에러가 없는 것으로 판단하는 경우와, 2) 모줄의 기능이 정확히 수행되었으나 에러가 발생된 것으로 판단하는 경우이다. 정규모줄이나 테스트모줄이 장애를 발생시켜 j번째 선로의 계산결과가 거절되면, 이 선로가 초기상태로 복구되고 (j+1)번째 선로가 활성화되어 임무를 수행한다. 마지막 n_i번째 선로에는 테스트모줄이 없어서 정규모줄에서 오류가 발생해도 이 선로가 수행된 후 다음 모줄이 동작한다. 따라서, 정상적으로 프로그램이 수행되었다 하더라도 실행결과가 에러일 수도 있다.

정규모줄 수행결과를 테스트모줄이 접수하여 이상 없는 것으로 판단되면 다음 모줄이 활성화되어 가동된다. 소프트웨어 시스템 내의 모든 소프트웨어가 통계적으로 독립이라고 가정한다. 이는 독립적으로 개발된 버전이 상이한 고장동태를 나타내어 결국 상이한 입력에 대해서 고장을 일으킨다는 것을 입증하는 것이다.

모줄내의 선로 각각에 대해서 설계, 전개, 운전하는데 필요한 자원의 양에 대해서 가정한다. 시스템의 총 자원은 모든 모줄의 자원요건의 합이다.

상기와 같은 가정 하에 소프트웨어 신뢰도 최적화 모델을 공식화하기로 하고, 아래와 같은 기호법을 도입한다.

m : 모줄의 수

i=1,…,m : 시스템 내에 있는 모줄의 지수

n_i : i번째 모줄의 선로수

j=1,…,n_i : 모줄내의 선로지수

r_{ij} : i번째 모줄에 있는 j번째 선로의 신뢰도(기능모줄을 참고할 것)

t_{ij} : i번째 모줄에 있는 j번째 선로에서의 테스트모줄 신뢰도

c_{ij} : i번째 모줄에 있는 j번째 선로에 필요한 자원의 양

R_i : i번째 모줄신뢰도

R : 시스템 전체의 신뢰도

C: 가용자원의 양

그림 1에서 선로의 기능 복구시 순서대로 위에서부터 아래로 실행 순서가 진행되는 것으로 가정하여 그 순서를 1, 2, …, j, …, n_i 선로로 하고, i번째 모줄의 신뢰도를 다음과 같이 정의한다.

$$R_i = \sum_{j=1}^{n_i} p_{ij} r_{ij} t_{ij} \quad (1)$$

단, 마지막 선로에는 테스트모줄이 없으므로 $t_{in_i}=1$로 가정한다.

여기서, p_{ij}는 i번째 모줄에 있는 j번째 선로가 실행될 확률을 나타내며, 반복적으로 정의한다.(그림 2 참조)

$$\begin{aligned} p_{i1} &= 1 \\ p_{i2} &= r_{i1}(1-t_{i1}) + (1-r_{i1})t_{i1} \\ p_{i3} &= p_{i2}\{r_{i2}(1-t_{i2}) + (1-r_{i2})t_{i2}\} \\ &\cdots\cdots\cdots\cdots\cdots \\ p_{ij} &= \prod_{k=0}^{j-1}\{r_{ik}(1-t_{ik}) + (1-r_{ik})t_{ik}\} \end{aligned} \quad (2)$$

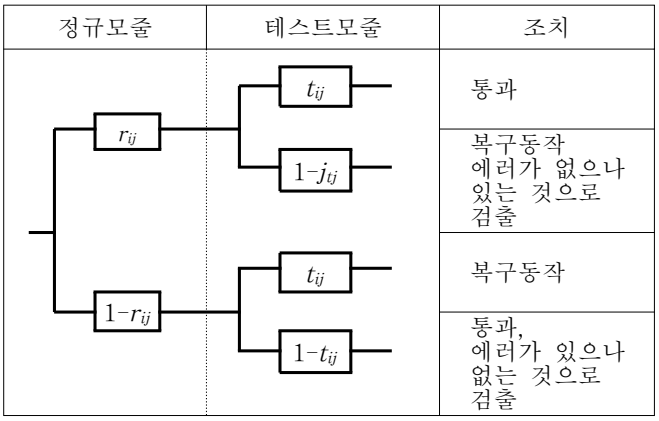

그림 2 모줄기능 수행 신뢰도

그림 2에서 r_{ij}, t_{ij}는 i번째 모줄의 j번째 선로에 있는 정규모줄, 테스트모줄 각자의 실행 성공확률을 나타낸다. 따라서, ($1-r_{ij}$), ($1-t_{ij}$)는 각각 상기 선로의 정규모줄 및 테스트모줄의 기능 실패 확률을 나타낸다.

따라서, 시스템 신뢰도를 다음과 같이 정리한다.

$$R = \prod_{i=1}^{m} \sum_{j=1}^{n_i} p_{ij} r_{ij} t_{ij} \quad (3)$$

그리고, 소프트웨어 최적화 문제는

$$\sum_{i=1}^{m}\sum_{j=1}^{n_i} c_{ij} \le C \qquad (4)$$

인 조건하에서 R를 극대화하는 것이다.

하나의 예로서 두 개의 모듈로 구성된 시스템을 생각해보기로 한다. 구성품의 신뢰도와 비용을 표 1과 같이 가정한다.

표 I 예제시스템의 신뢰도 및 비용

i	r_{ij}	$t_{ij}(j \neq n_i)$	$c_{ij}(j \neq n_i)$	$c_{ij}(j=n_i)$
1	0.90	0.85	12	8
2	0.85	0.85	10	6

좀더 간단히 하기 위해 n_i번째를 제외한 주어진 모듈내의 기능모듈의 신뢰도와 비용이 동일하다고 가정한다. 또한, 가용 자원의 양이 50이라고 가정한다. 표 1에서 제시된 수치를 가지고 (4)로 정의한 비선형프로그래밍 문제를 풀어서 최적시스템 구조를 구한다. 계산 결과는 표 2와 같으며, 가용자원 하에서 n_1 = 2, n_2 =3 인 경우가 최적으로서 이 때의 신뢰도는 93%이고, 자원은 46이다. 그 최적시스템 구조를 그림 3에 나타내었다.

표 2 예제시스템의 신뢰도 및 자원 계산 결과

n_1	1	2	3	4
n_2	4	3	2	1
$\sum_{i=1}^{m}\sum_{j=1}^{n_i} c_{ij}$	44	46	48	50
$\prod_{i=1}^{m}\sum_{j=1}^{n_i} p_{ij} r_{ij} t_{ij}$	0.87	0.93	0.92	0.83

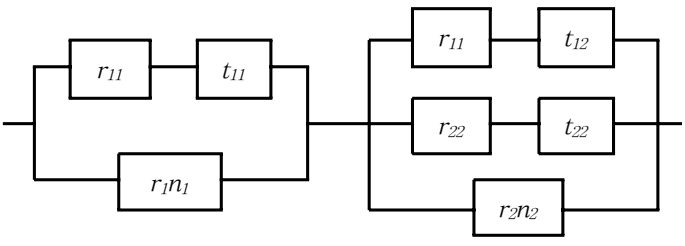

그림 3 예제 시스템의 최적도

3. 결론

소프트웨어 신뢰도를 최적화 하고자 하는 아이디어는 그의 타당성과 실현성을 증명하는 것이었다. 소프트웨어 신뢰도분야에 대해서 좀더 넓은 범위의 탐색 노력을 기울임으로써 그러한 증명을 할 수 있을 것으로 본다. 본 논문에서는 아직까지도 현안문제로 남아 있는 여러 가지 문제를 개략적으로 검토해보았다. 본 논문에서 제시한 모델은 일부 소프트웨어 관련 신뢰도 최적화 문제의 공식화와 해법이 가능하다는 것을 보여주고 있다. 이것들은 소프트웨어 신뢰도 최적화 분야에 내고장(耐故障)을 구현하기 위해 다중개념을 쓸 수 있다는 것을 보여주기도 한다.

참고문헌

[1] Hiroshi Ohtera, Shigeru Yamada, “Optimum Software-Release Time Considering an Error-Detection Phenomenon during Operation”, IEEE Trans. on Reliability, vol.39.No.5, 1990.12

[2] Shigeru Yamada, Hiroshi Ohtera, Hiroyuki Narihisa, “ Software Reliability Growth Models with Testing Efforts”, IEEE Trans. on Reliability, vol.R-35,No.1 1986.4

[3] Shigeru Yamada, Jun Hishitani, Shunji Osaki, “Software-Reliability Growth with a Weibull Test-Effort : A Model & Application”, IEEE Trans. on Reliability, vol.42.No.1, 1993.3

[4] Fevzi Belli, Pior Jedrzejowicz, "An Approach to the Reliability Optimization of Software with Redundancy", IEEE Trans. on Reliability, vol.17.No.3, 1991

[5] B. Littlewood, "How to measure software reliability and how not to" , IEEE Trans. Reliability, vol R-28, 1979 Jun, pp103-110.

[6] B. Littlewood, "Theories of software reliability" , IEEE Trans. Software Engineering, vol SE-6, 1980 Sep, pp465-484.

[7] Jelinski, P.B.Moranda, "Software Reliability Research" , in statistical computer performance evaluation New York, Academic Press, 1972 pp465-484.