소프트웨어 결함허용 기법들의 의존도 모델링

김용규°, 김성수
아주대학교 정보통신대학 정보및컴퓨터공학부

Dependability Modeling of Software Fault Tolerance Techniques

Yongkyu Kim, Sungsoo Kim
Division of Information and Computer Engineering, Ajou University

요 약

신뢰도 높은 소프트웨어 개발의 필요성은 전혀 새로운 것이 아니다. 요즘 들어, 소프트웨어의 크기와 복잡도가 증가함으로 인해 소프트웨어의 결함 때문에 발생하는 시스템 고장이 전체 시스템 고장에서 많은 비중을 차지하고 있다 고 신뢰도를 요구하는 시스템의 소프트웨어는 복구블록, 분산 복구블록, N-버전 프로그래밍, N 자기검사 프로그래밍과 같은 소프트웨어 결함허용 기법들을 사용하고 있다. 이러한 소프트웨어 결함허용 기법들에 대한 연구와 함께 소프트웨어 결함허용 기법들의 의존도 측정에 관한 연구 또한 매우 중요하다 이에 본 논문에서는 마르코프 모델을 사용해서 소프트웨어 결함허용 기법들의 보다 자세한 신뢰도 모델링과 가용도, 안전도 등에 관한 모델링을 제시한다 제안된 모델 분석 결과 같은 수의 대체블록이 있을 때는 분산 복구블록, 복구블록, N 자기검사 프로그래밍, N-버전 프로그래밍 순으로 의존도가 높음을 알 수 있다 또한 소프트웨어 결합허용 기법들의 신뢰도 민감성 분석에서는 복구블록과 분산 복구블록인 경우는 적용검사의 결합발생율에, N-버전 프로그래밍인 경우는 프로그램 버전의 결합발생율에 더 민감한 영향을 받는 것을 알 수 있다.

1 서론

항공산업, 우주산업, 원자력산업 등에 사용되는 시스템과 같이 고 신뢰도를 요구하는 시스템의 소프트웨어는 개발 시 결함을 없애려는 노력이 활발히 진행되고 있다 하지만, 소프트웨어의 크기와 복잡도가 증가함으로 인해 모든 결함 발생의 가능성을 없앨 수는 없기 때문에 이러한 노력과 더불어 결함을 허용하는 소프트웨어 즉 수행 시 어떤 부분에 결함이 발생하더라도 올바른 결과를 산출할 수 있는 소프트웨어를 개발하고자 하는 연구가 진행되고 있다 이제까지 연구된 대표적인 소프트웨어 결함허용 기법들로는 복구블록[1], 분산 복구블록[2], N-버전 프로그래밍[3], N 자기검사 프로그래밍[4] 등이 있다. 이와 더불어 이러한 소프트웨어 결함허용 기법들의 의존도 측정에 대한 연구 또한 매우 중요하다

소프트웨어 결함허용 기법들의 의존도 분석에 관한 연구는 이제까지 신뢰도에 국한되어져 있고 신뢰도에 대한 연구도 상당히 단순화해서 이루어졌기 때문에 정확한 신뢰도 분석이 이루어지지 못했다[5, 6]. [5]에서는 세 가지 대표적인 소프트웨어 결함허용 기법들인 분산 복구블록, N-버전 프로그래밍과 N 자기검사 프로그래밍의 신뢰도와 안전도 분석을 위한 시스템 수준의 모델링을 마르코프 모델과 결함 트리를 사용해서 제안했다 [6]에서는 복구블록, N-버전 프로그래밍의 신뢰도 분석을 위한 모델링을 마르코프 모델을 사용해서 제안하고 두 기지 소프트웨어 결함허용 기법들의 비교도 같이 보여준다

이에 본 논문에서는 마르코프 모델을 이용해서 소프트웨어 결함허용 기법들의 보다 자세한 신뢰도 모델링과 가용도, 안전도 등에 관한 모델링을 제시한다 제 2장에서는 대표적인 소프트웨어 결함허용 기법들인 복구블록, 분산 복구블록, N-버전 프로그래밍, N 자기검사 프로그래밍의 구조를 분석해 본다 제 3장에서는 소프트웨어 결함허용 기법들의 의존도를 분석하기 위한 마르코프 모델을 제안하고, 제 4장에서는 여러 가지 측면에서 의존도를 분석한 결과를 살펴본다 마지막으로 제 5장에서는 본 논문에 대한 결론을 맺고 향후 연구 방향에 대해서 논한다

본 연구는 1999년도 정보통신부 정보통신 우수시범학교 지원사업에 의한 결과임

2 소프트웨어 결함허용 기법들의 구조 분석

2.1 복구 블록 (Recovery Block)

복구블록은 주 처리블록, 여러 개의 대체 처리블록 및 적응검사로 구성되어 있다. 복구블록내의 모든 처리블록은 다르게 설계되어야 하고 그 수행 결과는 동일하거나 수용할 수 있는 근접된 결과이어야 한다 적응검사는 처리블록의 수행 결과를 판정하는 논리식으로 그 결과의 수용여부를 판정한다.

2.2 분산 복구블록 (Distributed Recovery Block)

분산 복구블록은 주 처리블록, 여러 개의 대체 처리블록과 적응검사로 구성되어 있다 분산 복구블록은 대체 처리블록을 동시에 수행하므로 복구 수행속도가 짧기 때문에 시간이 긴박한 응용분야에 사용될 수 있다. 분산 복구블록에서의 적응검사는 계산 결과를 판정하는 적응검사와 수행시간의 경과 여부를 판정하는 적응검사의 복합으로 구현된다.

2.3 N-버전 프로그래밍 (N-Version Programming)

N-버전 프로그래밍은 하드웨어의 NMR(N-Modular Redundancy)과 유사한 개념의 결함허용 기법이다. N-버전 프로그래밍의 기본 개념은 소프트웨어 모듈을 N번 설계하고 코드화 하는 것이고 이 모듈들에 의해 생성된 N개의 결과를 비교하는 것이다

2.4 N 자기검사 프로그래밍 (N Self-Checking Programming)

N 자기검사 프로그래밍은 하드웨어 결합허용의 동적 중복에 해당하는 것으로, 실행 중에 자신의 동적 동작을 검사하는 자기검사 프로그램을 이용하여 결합허용 구조를 만든 것이다

3. 소프트웨어 결함허용 기법들의 의존도 모델

이 장에서는 2장에서 분석한 결합허용 소프트웨어 기법들의 신뢰도, 가용도, 안전도 분석을 위한 효율적인 마르코프 모델을 제안한다.

먼저 이번 장에서 사용되는 기호를 살펴보면 다음과 같다 λ_p는 프

로그램 버전의 결함발생율을, λ_A는 적응검사의 결함발생율을, λ_V는 N-버전 프로그래밍에서 보터의 결함발생율을 의미하고, μ_P, μ_A, μ_V는 각각에 대한 결함복구율을 의미한다. $p_i(t)$는 t시간에 i상태에 있을 확률을 의미하고, $R_{ftsw}(t)$, $A_{ftsw}(t)$, $S_{ftsw}(t)$는 각각의 결함허용 소프트웨어 기법들에 대한 t시간의 신뢰도, 가용도, 안전도를 의미한다.

3.1 신뢰도 분석

신뢰도 모델링을 위해 몇 가지 가정이 필요한데 여기에서 사용하는 가정은 다음과 같다

① 결함복구는 고려하지 않는다
② 한번에 단지 하나의 고장만이 발생한다.
③ 시스템은 완벽한 상태에서 시작한다

복구 블록은 아래 그림 1처럼 주 처리블록이 동작하고 있는 상태를 상태 0으로, 주 처리블록이 고장나고 첫 번째 대체 처리블록이 동작하는 상태를 상태 1로, 계속해서 N-1번째 대체 처리블록이 동작하는 상태를 상태 N-1로 모델링이 이루어진다. 그리고, 모두 고장이 난 상태를 상태 F로 표시할 수 있다 그림 1과 같은 모델을 풀어보면 각각의 상태에서의 확률은 다음과 같이 구해진다.

$$p_i(t) = \lambda_P^i \frac{1}{i!} t^i e^{-(\lambda_P+\lambda_A)t} \quad (i = 0, \ldots, N-1)$$

$$p_F(t) = 1 - \sum_{i=0}^{N-1} \lambda_P^i \frac{1}{i!} t^i e^{-(\lambda_P+\lambda_A)t}$$

복구블록의 신뢰도는 상태 0에서 N-1까지에 있을 확률이므로 다음과 같이 구해진다

$$R_{RB}(t) = \sum_{i=0}^{N-1} \lambda_P^i \frac{1}{i!} t^i e^{-(\lambda_P+\lambda_A)t}$$

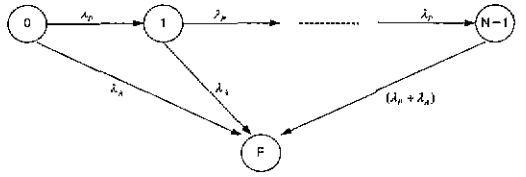

그림 1 복구블록의 신뢰도 모델링

분산 복구블록은 복구블록이 병렬로 수행되는 것과 같은 구조이기 때문에 복구블록으로부터 쉽게 구할 수가 있다 즉, $R_{RB}(t)$를 복구블록의 신뢰도라고 하면, 분산 복구블록의 신뢰도 $R_{DRB}(t)$는 아래 식과 같이 구할 수 있다

$$R_{DRB}(t) = 1.0 - \prod_{i=1}^{N}(1.0 - R_{RB}(t))$$

N-버전 프로그래밍은 아래 그림 2처럼 N개의 버전 모두가 올바르게 동작하고 있는 상태를 상태 N으로, $\lceil \frac{N+1}{2} \rceil$개의 버전이 동작할 때까지 진행되다가 시스템 고장 상태인 상태 F에 도달하게 된다

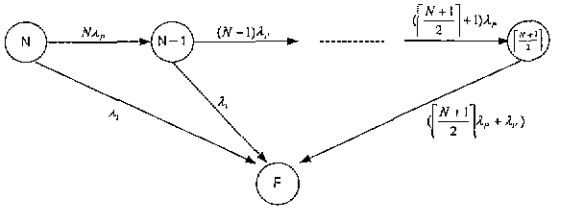

그림 2 N-버전 프로그래밍의 신뢰도 모델링

그림 2와 같은 모델을 풀어보면 각각의 상태에서의 확률은 다음과 같이 구해진다

$$p_N(t) = e^{-(N\lambda_P+\lambda_V)t}$$

$$p_{N-1}(t) = N(e^{-((N-1)\lambda_P+\lambda_V)t} - e^{-(N\lambda_P+\lambda_V)t})$$

$$p_{N-2}(t) = \frac{N(N-1)}{2}(e^{-((N-2)\lambda_P+\lambda_V)t} - 2e^{-((N-1)\lambda_P+\lambda_V)t} + e^{-(N\lambda_P+\lambda_V)t})$$

N 자기검사 프로그래밍은 아래 그림 3처럼 모든 버전이 올바르게 동작하고 있는 상태를 상태 0으로, 하나의 버전이 고장이 나서 두 번째 버전이 동작하는 상태를 상태 1로, 계속해서 마지막 버전이 동작하는 상태를 상태 N-1로 나타내고, 모든 버전이 고장이 난 상태를 상태 F로 표시할 수 있다. 그림 3과 같은 모델을 풀어보면 각각의 상태에서의 확률은 다음과 같이 구해진다

$$p_0(t) = e^{-N(\lambda_P+\lambda_A)t}$$

$$p_1(t) = N(e^{-(N-1)(\lambda_P+\lambda_A)t} - e^{-N(\lambda_P+\lambda_A)t})$$

$$p_2(t) = \frac{N(N-1)}{2}(e^{-(N-2)(\lambda_P+\lambda_A)t} - 2e^{-(N-1)(\lambda_P+\lambda_A)t} + e^{-N(\lambda_P+\lambda_A)t})$$

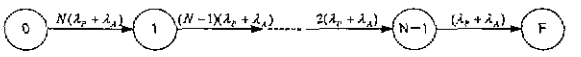

그림 3 N 자기검사 프로그래밍의 신뢰도 모델링

3.2 가용도 분석

가용도 모델링을 위해서는 신뢰도 모델링과는 다른 가정이 필요한데 여기에서 사용하는 가정은 다음과 같다.

① 결함복구를 고려한다
② 한번에 단지 하나의 고장만이 발생한다.
③ 시스템은 완벽한 상태에서 시작한다.

신뢰도 모델링과 다른 점이라면 가정 ①처럼 결함 복구를 고려한다는 점이다. 결함복구를 고려함으로 해서 결함발생율과 함께 결함복구율이 모델링에 추가된다

N 자기검사 프로그래밍의 경우만 살펴보면 아래 그림 4처럼 모델링이 이루어진다. 그림 4와 같은 모델을 풀어보면 각각의 상태에서의 확률은 다음과 같이 구해진다.

$$p_k = \frac{(\frac{\lambda_P+\lambda_A}{\mu_P+\mu_A})^k \binom{N}{k}}{(1+\frac{\lambda_P+\lambda_A}{\mu_P+\mu_A})^N} \qquad k = 0, \ldots, N-1, N(=F)$$

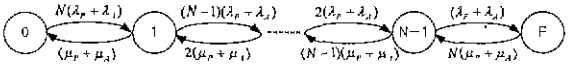

그림 4 N 자기검사 프로그래밍의 가용도 모델링

3.3 안전도 분석

안전도 분석을 위한 마르코프 모델링은 신뢰도 모델링과 유사한 면이 많지만 앞에서 했던 가정과는 다른 가정이 필요한데 여기에서 사용하는 가정은 다음과 같다.

① 결함복구는 고려하지 않는다
② 한번에 단지 하나의 고장만이 발생한다.
③ 시스템은 완벽한 상태에서 시작한다
④ 시스템 고장 발생 시 두 가지 상태 즉, 안전 상태와 불안전 상태로 나누어진다

신뢰도 모델링과 다른 점이라면 가정 ④에 있는 것처럼 시스템 고장 발생 시 두 가지 상태로 나뉘어지고 안전 상태도 시스템이 안전한 상태라고 본다는 점이다 이와 같은 가정을 가지고 모델링을 하게 되면 신뢰도 모델링으로부터 쉽게 모델링을 할 수 있다

4. 소프트웨어 결함허용 기법들의 의존도 분석

이번 장에서는 3장에서 마르코프 모델링을 이용하여 제안했던 각각의 소프트웨어 결함허용 기법들에 대한 의존도를 분석한다

먼저 이번 장에서 사용되는 기호로는 RB(복구블록), DRB(분산 복구블록), NVP(N-버전 프로그래밍), NSCP(N 자기검사 프로그래밍)등이

있으며 각 기호의 뒤에 붙는 숫자는 프로그램 버전의 개수를 나타낸다

그림 5는 시간의 변화에 따른 각 소프트웨어 결함허용 기법들의 신뢰도 변화 그래프이다 프로그램 버전(10^{-5}/시간), 적응검사(10^{-7}/시간), 보터(10^{-7}/시간)의 결함발생율 λ는 고정시키고, 시간을 변화시키면서 신뢰도의 변화를 보여주고 있다. 그림에서 보는 바와 같이 같은 수의 대체블록이 있을 때는 분산 복구블록, 복구블록, N 자기검사 프로그래밍, N-버전 프로그래밍 순으로 신뢰도가 높은 것을 볼 수 있다 그리고, 대체블록의 수가 증가할수록 신뢰도가 증가하는 것을 볼 수 있지만 특이하게 N-버전 프로그래밍의 경우에는 5-버전 프로그래밍이 6-버전 프로그래밍보다 더 높은 신뢰도를 나타내는 것을 볼 수 있다 이 경우는 N-버전 프로그래밍의 다른 경우에도 나타나는 현상으로 홀수개의 버전이 있을 때 더 좋은 신뢰도를 나타낸다

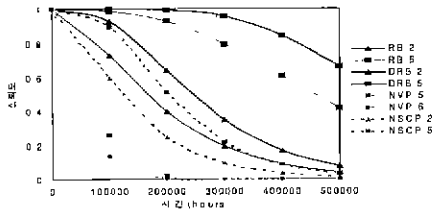

그림 5 소프트웨어 결함허용 기법들의 신뢰도 분석
($\lambda_P = 10^{-5}$/시간, $\lambda_A = \lambda_V = 10^{-7}$/시간)

그림 6은 시간의 변화에 따른 각 소프트웨어 결함허용 기법들의 가용도 변화 그래프이다 프로그램 버전, 적응검사, 보터의 결함발생율 λ는 모두 0.01/시간으로 고정시키고, 결함복구율 μ도 0.05/시간으로 고정시키고, 시간을 변화시키면서 가용도의 변화를 보여주고 있다. 그림에서 보는 바와 같이 같은 수의 대체블록이 있을 때는 신뢰도와 같이 분산 복구블록, 복구블록, N 자기검사 프로그래밍, N-버전 프로그래밍 순으로 가용도가 높은 것을 볼 수 있다 그리고, 대체블록의 수가 증가할수록 가용도가 증가하는 것을 볼 수 있지만 신뢰도에서와 같이 N-버전 프로그래밍의 경우에는 5-버전 프로그래밍이 6-버전 프로그래밍보다 더 높은 가용도를 나타내는 것을 볼 수 있다.

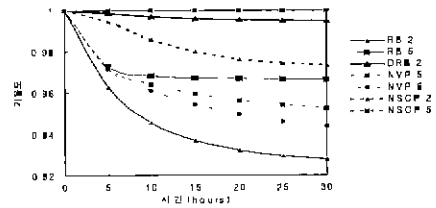

그림 6 소프트웨어 결함허용 기법들의 가용도 분석
(λ = 0.01/시간, μ = 0.05/시간)

시간의 변화에 따른 각 소프트웨어 결함허용 기법들의 안전도 변화는 신뢰도 분석 그래프와 거의 유사한 결과를 보여준다. 다만, 결함커버리지 C의 값에 따라 그 값 이하로는 안전도가 떨어지지는 않는다

그림 7은 복구블록과 N-버전 프로그래밍의 신뢰도가 프로그램 버전과 적응검사 또는 보터의 결함발생율의 변화 중 어떤 것에 더 민감한지를 보여주는 것이다 그림 7에서 RB 2는 프로그램 버전과 적응검사의 결함발생율이 10^{-5}/시간인 경우를 나타내고, RB 2 (P=0.0001)는 적응검사의 결함발생율은 10^{-5}/시간으로 두고 프로그램 버전의 결함발생율만 10^{-4}/시간으로 10% 더 높인 경우를 보여준다 RB 2 (A=0.0001)는 프로그램 버전의 결함발생율은 10^{-5}/시간으로 두고 적응검사의 결함발생율만 10^{-4}/시간으로 10% 더 높인 경우를 보여준다. 그리고, N-버전 프로그래밍에서도 NVP 3은 프로그램 버전과 보터의 결함발생율을 10^{-5}/시간인 경우를 나타내고, NVP 3 (P=0.0001)은 프로그램 버전의 결함발생율만 10^{-4}/시간으로 10% 더 높인 경우이고, NVP 3 (V=0.0001)은 보터의 결함발생율만 10^{-4}/시간으로 10% 더 높인 경우를 보여준다. 그림에서 보는 바와 같이 복구블록의 경우에는 적응검사의 결함발생율에 의해 신뢰도가 더 급격히 감소하고, N-버전 프로그래밍인 경우에는 프로그램 버전의 결함발생율에 의해 신뢰도가 더 급격히 감소한다

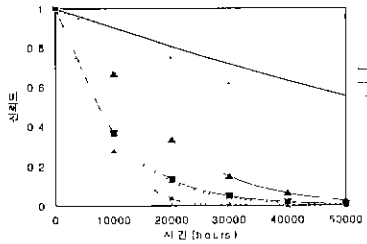

그림 7 소프트웨어 결함허용 기법들의 신뢰도 민감성 분석

5. 결론

소프트웨어의 결함을 다루는 기법에는 첫 번째, 소프트웨어를 만드는 과정에서 V&V (verified and validated) 도구들과 프로시저들을 사용하는 예방(prevention) 방법, 두 번째, 최근의 소프트웨어 재활(software rejuvenation) 방법에 의해 소프트웨어 수행 중에 주기적으로 재 시작하는 회피(avoidance) 방법, 세 번째, 본 논문에서 알아본 결함허용 소프트웨어 구성요소들에 의한 검지와 복구(detection and recovery)방법이 있다.

본 논문에서는 마르코프 모델을 사용하여 대표적인 소프트웨어 결함허용 기법인 복구블록, 분산 복구블록, N-버전 프로그래밍, N 자기검사 프로그래밍의 의존도를 분석하기 위한 모델을 제안했다 이를 위해 먼저 이들 소프트웨어 결함허용 기법들의 구조를 살펴보았고 각각에 대해 신뢰도, 가용도, 안전도 측면에서 모델링이 이루어졌다.

제안된 모델을 기반으로 분석한 결과 같은 수의 대체블록이 있을 때는 분산 복구블록, 복구블록, N 자기검사 프로그래밍, N-버전 프로그래밍 순으로 의존도(신뢰도, 가용도, 안전도)가 높음을 알 수 있다 특이한 점은 N-버전 프로그래밍의 경우에는 홀수 버전인 경우에 하나 더 많은 버전을 가지고 있는 짝수 버전보다 의존도가 더 높음을 볼 수 있다.

또한 각 소프트웨어 결함허용 기법들이 프로그램 버전과 적응검사 또는 보터의 결함발생율의 변화 중 어떤 구성요소에 더 많은 영향을 받는가에 대한 분석도 이루어졌는데, 복구블록과 분산 복구블록인 경우는 적응검사의 결함발생율에, N-버전 프로그래밍인 경우는 프로그램 버전의 결함발생율에 더 민감한 반면 N 자기검사 프로그래밍의 경우는 프로그램 버전과 적응검사에 같은 영향을 받는 것을 알 수 있다

하지만, 이러한 모델링 및 분석 결과들은 모델링 과정에서 어떠한 부분을 모델링에 포함시킬 것인지에 영향을 받기 때문에 이러한 추상화된 모델링을 없애면서 좀더 접근된 결과를 얻기 위해 시뮬레이션 방법을 사용해서 분석하는 연구가 이루어져야 할 것이다

참고 문헌

[1] B Randell, "System Structure for Software Fault Tolerance," IEEE Transactions on Software Engineering, SE-1(2):220-232, June 1975.

[2] K.H. Kim and H O Welch, "Distributed Execution of Recovery Blocks An Approach for Uniform Treatment of Hardware and Software Faults in Real-Time Applications," IEEE Transactions on Computers, 38(5) 626-636, May 1989.

[3] A. Avizienis, "The N-Version Approach to Fault-Tolerant Software," IEEE Transactions on Software Engineering, SE-11(12) 1491-1501, December 1985.

[4] J.C Lapne, J Arlat, C Beounes, and K Kanoun, "Definition and Analysis of Hardware- and Software-Fault-Tolerant Architectures," IEEE Computer, pp 39-51, July 1990

[5] J B Dugan and M R. Lyu, "Dependability Modeling for Fault-Tolerant Software and Systems," Software Fault Tolerance, Wiley Trends in Software Book Series, Wiley, pp 109-138, February, 1995.

[6] J Arlat, K Kanoun, and J C. Lapne, "Dependability Modeling and Evaluation of Software Fault-Tolerant Systems," IEEE Transactions on Computers, Vol 39, No. 4, pp. 504-513, April 1990