오류 분류를 이용한 소프트웨어 신뢰도 모델

조 영 식[†] · 이 용 근[††] · 최 형 진[†††] · 양 해 술[††††]

요 약

지금까지 많은 소프트웨어 신뢰도 성장 모델들이 제안되어 왔으나 현상을 완전히 고려한 소프트웨어 신뢰도 모델은 아직 개발되어 있지 못한 실정이다. 따라서 다양한 소프트웨어 신뢰도 모델이 계속 개발되어 모든 현상을 고려할 수 있는 모델의 개발이 필요하다고 볼 수 있다. 이와 같은 목적하에 본 논문에서는 발견되는 오류를 오류의 특성에 따라 3종류로 분류하였고, 또한 각 오류의 분류된 오류의 발견 시점 등을 고려하여 소프트웨어의 신뢰도를 측정할 수 있는 신뢰도 모델을 제안하였다. 그리고 모델의 파라메타의 추정 및 신뢰성 평가를 위해 실측 데이타를 이용하여 고찰하였으며, 본 논문의 적합성을 검증하기 위해 기존의 모델과 비교 분석하였다.

A Study of Software Reliability Model Using Error-Class

Young Sik Cho[†] · Ryong Geun Lee[††] · Hyung Jin Choi[†††] · Hae Sool Yang[††††]

ABSTRACT

The reliability in software has expand in quality and quantity, also its importance and role are increased. But, a study of software reliability is lack of development. this paper software reliability growth models(SRGM) described by NonHomo-geneous Poisson(NHPP) processes. Using actual software error data observed by software testing, the SRGM's are composition of error-class, and error-class by three class. this paper made the reliability-model of software using three error-class. The purpose of this study to increase software productivity and to improve software quality. So to achive these goals we focused on a study of software reliability model using the error-class.

1. 서 론

1970년대부터 소프트웨어 신뢰도에 관한 연구가 꾸준히 진행되어 오고 있지만, 하드웨어와 같이 현상을 완전히 고려한 소프트웨어 신뢰도 모델은 아직 개발되지 못하고 있는 실정이다. 이와 같은 이유는 소프트웨어와 하드웨어의 특성이 근본적으로 차이가 있기 때문이다. 하드웨어의 고장은 마모현상이나 우연한 사건에 의해서 유발되지만 소프트웨어의 고장은 소프트웨어 내부에 잔재하는 오류가 어떤 입력 자료에 의해서 현재화될 때 고장이 발생되는 것이기 때문이다[2, 10, 11, 12].

그러나 오류가 소프트웨어내에 존재한다고 해도 존재하는 오류 모두가 고장을 유발시키는 것은 아니며, 특정한 자료가 입력되지 않으면 오류와 입력된 자료가 서로 사상(mapping)되지 않기 때문에 고장은 영원히 발생되지 않을 수도 있다. 결과적으로 소프트

† 준 회 원: 남서울산업대학교 전자계산학과 강사
†† 종신회원: 경희대학교 전자계산공학과 강사
††† 종신회원: 강원대학교 전자계산학과 조교수
†††† 종신회원: 한국소프트웨어 품질연구소(INSQ)소장
논문접수: 1995년 4월 24일, 심사완료: 1996년 12월 20일

웨어를 완벽하게 테스팅하기 위해서는 모든 가능한 입력에 대해서 소프트웨어를 테스팅해야 하나 현재의 기술 수준으로는 불가능하다[11, 12].

현재 대부분의 소프트웨어 신뢰도 모델들은 임의의 시각까지 테스팅한 결과 발생되는 오류의 개수를 기초로 잔재하는 오류의 수를 예측하는 것이며, 이 예측된 결과에 따라 어느 정도 만족할 만한 수준의 오류가 잔재하는 상태로 소프트웨어를 출고시키고 있는 실정이다. 이러한 인식하에 소프트웨어의 신뢰도와 유용성에 관련된 논문들이 오류 예측의 수학적 모델에 집중적으로 연구되고 있으나, 현상을 완전히 해결하기에는 아직 미흡한 실정이다[2, 3, 4, 12].

소프트웨어 신뢰도 성장 모델은 일반적으로 데이타 종속 모델과 시간 종속 모델의 2가지로 분류되어 왔다. 데이타 종속 모델에는 오류의 삽입 모델과 입력 도메인 모델이 있고, 시간 종속 모델은 오류의 시간적 검출 과정을 수식화하여 그 수식의 계수를 실제의 오류 데이타로부터 결정하여 잔재 오류수를 추정하는 모델이다. 이러한 모델의 대부분은 소프트웨어 내에 잔재하고 있는 오류의 발견/수정이 용이하고, 모든 오류에 대하여 동등하다고 보고 있는 것이 대부분이다. 이 가정은 모델의 단순화에는 적당하지만 실제의 오류 발견 과정에는 적당하지 않다.

따라서 본 논문에서는 모델의 단순화와 오류 발견 과정을 쉽게하기 위해 오류 분류를 이용한 신뢰도 모델을 제안하였다. 본 논문에서 제안하는 모델은 발견되는 오류의 특성에 따라 분류(class)하여 오류 중요도가 가장 높은 오류를 클래스 3오류(class 3 error), 보통 오류를 클래스 2오류(class 2 error), 중요도가 가장 낮은 오류를 클래스 1오류(class 1 error)로 분류하였다[12]. 그리고 분류된 오류는 실측 데이타를 고찰하여 각 분류된 오류의 발견 시점을 추정할 수 있는 기준을 설정하여 신뢰도 모델을 제안하였다. 그리고 모델 파라메타의 추정 및 신뢰도 평가를 위해 실측 데이타를 이용하여 고찰하였으며, 본 논문의 적합성을 평가하기 위해 기존의 모델과 비교 분석하였다.

2. 소프트웨어 신뢰도 모델 분류 및 연구배경

2.1 정의

일반적으로 신뢰도란 주어진 기간 동안 충족하게 작동될 확률이고 소프트웨어 신뢰도란 "주어진 기간 동안 주어진 환경하에서 소프트웨어가 고장없이 주어진 명세서대로 정상 작동될 확률"로 정의한다[1, 2, 3, 11, 12].

이 정의에서 사용되는 확률, 기간, 명세서 및 고장에 대한 정의는 다음과 같다.

(1) 신뢰도는 "확률"이다.

신뢰도를 측정하기 위해 이용되는 것으로 확률론, 추정이론, 확률분포, 마코브 프로세스, 최우추정 등의 개념들이 이용된다.

(2) 신뢰도는 "환경"에 의존한다.

환경은 프로그램의 개발 및 운영환경을 나타내는데 여러형태의 "실행(run)"시의 상대적인 확률을 의미한다. 즉, 동일한 프로그램일지라도 운영환경이 다르면 신뢰도는 각기 다르게 나타난다. 하드웨어에서 환경이란 기온, 부하, 습도, 가속도, 진동, 충격, 소음 등이지만, 소프트웨어에서는 개발운영 환경을 의미하며 요구사항 정의, 계획, 설계, 구현, 테스팅 및 유지보수의 각 단계에 있어서의 환경을 의미한다. 특히, 소프트웨어 신뢰도는 운영환경(예:컴퓨터의 부하, 다른 소프트웨어간의 관계, 운영자의 문제 등) 보다는 개발환경(예:개발자 능력, 좋은 도구의 이용 등)이 신뢰도에 큰 영향을 미치므로 소프트웨어 신뢰도 모델에서 이러한 환경 인자들이 많이 고려된 것일수록 좋은 신뢰도 모델이라할 수 있다.

(3) 신뢰도는 "일정기간" 동안 측정된다.

일정기간을 노출 기간(exposure period)라 하며 CPU 시간, 또는 달력 시간(calendar time) 등을 의미한다.

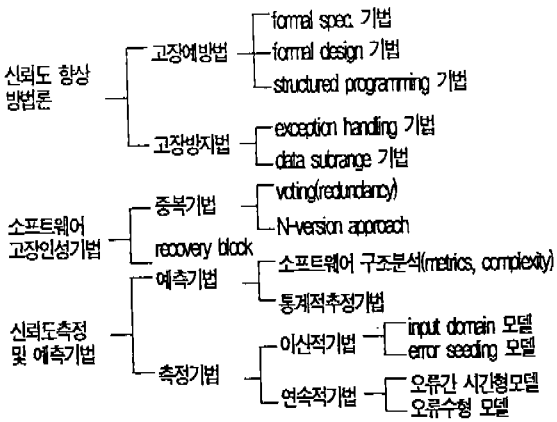
(4) 신뢰도는 "명세서"를 바탕으로 한다.

고장은 명세서와 실제 결과가 다를 때를 의미하므로 신뢰도 예측(측정)을 위해서는 명세서의 결과 값이 어떤 것을 요구하는지 알아야 한다. 이를 위해 소프트웨어의 명세서를 공식화할 필요가 있다. 예를 들면, 수치해석 프로그램의 경우 허용 가능한 오류 범위가 명세서에 기술되어 있으므로 이 범위내의 결과만을 취하면 된다. 즉, 결과는 오류가 있지만 허용 가능한 오류이므로 고장이라 생각하지 않고 신뢰도 추

정시 고장수로 계산하지 않는다. 따라서 명세서에는 모든 입출력 공간과 이들 사이의 사상에 관한 모든 정보가 그 정확도(허용 오류 범위)와 함께 기록되어야 한다.

2.2 신뢰도의 특성 및 분류

소프트웨어 신뢰도는 매우 넓은 의미를 포함하고 있으며, 소프트웨어 공학에서 다루는 모든 방법론과 모델들은 사실 넓은 의미의 소프트웨어 신뢰도 분야라 할 수 있다[2, 3, 10, 12]. 즉, 소프트웨어 공학의 방법론과 모델들은 소프트웨어의 신뢰도를 높이기 위한 방법론이라고도 할 수 있기 때문이다. 소프트웨어 신뢰도 분야는 (그림 1)과 같이 크게 신뢰도의 향상, 측정 및 예측, 그리고 고장 인성으로 분류된다[12].



(그림 1) 소프트웨어의 신뢰도 분류
(Fig. 1) Reliability class of software

1) 신뢰도 향상 방법론

오류(error, fault, failure)를 예방하거나 방지하는 것으로서는 체계화된 방법론을 통해 소프트웨어 내부로 오류가 삽입되는 것을 방지하는 방지(prevention) 방법들과 실행중 오류 발생률을 예측하고 예외상황 처리(exception handling) 기능들을 소프트웨어 내부에 삽입 시키는 회피(avoidance) 방법들이 있다. 이러한 방법의 성능은 오류의 예방률(prevention rate) 또는 방지률(avoidance rate)을 비교척도로 하여 정량적으로 평가할 수 있다.

소프트웨어 공학 분야에서 다루는 대부분의 방법론(methodology)들은 신뢰도 향상을 위한 방법들이라 할 수 있다.

2) 소프트웨어 오류 인성 방법론

소프트웨어 오류 인성 방법론은 소프트웨어 신뢰도 향상을 위한 방법으로서 독립적으로 발전된 분야이며 하드웨어의 오류 인성 기법들을 수정하여 소프트웨어에 적용한 것이다. 오류 인성(fault tolerant)이란 비록 설계시에 오류를 예방하거나 방지하지 못했을지라도 운영시에 오류가 시스템의 파괴를 막거나 피해를 최소한으로 줄이고 긴급히 회복함으로써 신뢰도를 향상시키는 개념이다. 이 방법론들은 운영시 오류의 발견, 시스템의 재구성 및 회복 기법들로 세분화되며, 인성 방법들의 성능평가 척도로서 "오류 인성률" 등을 이용하여 평가할 수 있다.

3) 신뢰도 측정 모델

테스팅중 개발 관리자는 현재의 신뢰도(남은 오류수)와 원하는 신뢰도 수준이 주어질 때 언제까지 테스팅 해야할지를 결정해야 한다. 이를 위해 테스팅 동안 발생된 오류 데이타를 바탕으로 몇 가지 가정을 세우고 통계적 추정방법에 의해 신뢰도를 측정한다.

이러한 측정 방법들은 입력영역 중심 및 오류 삽입 방법들과 같은 이산적 기법들과 오류간 시간형 모델(time between fault model)과 오류수형 모델(fault count model)과 같은 연속적 기법들을 이용한 신뢰도 모델로 분류된다. 각 모델들에 대하여 간략히 살펴보면 다음과 같다.

①입력 영역 중심 모델(input domain based model)

소프트웨어의 개발, 검증, 유지보수 및 신뢰도 측정 단계에서 적용할 수 있는 모델이며, 소프트웨어를 함수로 간주하여 입력 정의역 자료 집합중 오류 입력이 출력 치역으로 사상되는 비율을 소프트웨어의 신뢰도로 보는 방법이다.

모든 가능한 입력 데이타를 테스팅할 수 없으므로 테스팅 데이타 선택기법(test case selection)에 의해 선택된 입력 데이타를 이용하여 신뢰도를 측정하는 방법이다. 이 방법들의 정확성은 테스팅 데이타의 선택에 의존하고 있으므로 테스트 케이스, 발생 알고리즘의 선택이 중요하며 신뢰도의 측정은 별도로 하는

것이 아니라 테스팅 과정과 병행한다.

②오류 삽입 모델(error seeding model)

신뢰도 측정 과정에서 이용할 수 있는 방법으로서 통계적 측정 방법 중 하나인 "tagging" 방법을 응용한 것이며, 소프트웨어내에 오류를 삽입시킨 후 테스팅을 실시하여 삽입된 오류들의 발견수와 전체 오류 발견수와의 비율을 이용하여 신뢰도를 측정하는 방법이다. 이 방법에서는 삽입된 오류가 소프트웨어내에 넓게 분포되어야 하며 테스팅을 실시하는 사람이 오류 삽입에 관한 정보를 알지 못해야만 정확한 신뢰도를 측정할 수 있다는 어려움이 있다.

③오류간 시간형 모델

이 형태의 모델은 초기의 기본적인 소프트웨어 신뢰도 모델로서 소프트웨어의 오류시간 간격이 지수분포를 따르며 서로 독립적이고 분포함수의 모수를 추정할때 오류간 시간형 오류자료가 이용되며, 오류 발생 시간 등을 예측할 수 있는 모델이다.

i-1번째 오류 발생 시간과 i번째 오류 발생 시간 간격은 그 기간동안 소프트웨어 내에 잔재하는 오류수에 비례하는 지수분포를 따른다고 가정한다. 이 모델들은 JM모델의 가정을 축소한 좀더 현실적인 소프트웨어 신뢰도 모델들이다.

④오류수형 모델(fault count model)

이 모델들은 단위시간 동안 구간내의 고장수가 시간에 종속적인 포아송 분포를 따른다고 가정하며, 모델의 모수들은 고장수 형태의 고장자료를 이용해 추정한다. 포아송 분포에서 t시점에서 누적 고장수가 y개일 확률은

$$\text{Prob}\{N(t)=y\}=\frac{m(t)^y}{y!}\times\exp\{-m(t)\}\quad (y=0,\ 1,\ 2,\ldots)$$

$$m(t)=\int_0^t \lambda(t)dt,\ t\geq 0$$

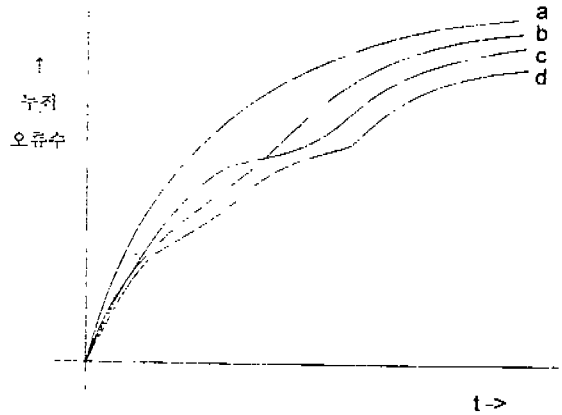
이며, 여기서 N(t)는 누적된 오류수 형태의 자료이고, m(t)는 t시점까지 발견되는 평균 오류수이고, 이를 미분하면 고장율 λ(t)를 얻는다. 그리고 평균치 함수 m(t)는 각 모델에 따라 여러 형태로 변화되어 왔다.

2.3 문제점 분석

종래의 소프트웨어 신뢰도 모델의 대부분은 "테스트 초기에 많은 오류가 발견되고 테스트 시간이 경과함에 따라 발견되는 오류의 수는 감소한다"라는 시간적 오류 발생 현상을 고찰하여 개발되어져 왔다. 이러한 관점하에 "테스트 초기에는 오류 발견률이 높고 테스트 시간이 점차 증가할수록 오류 발견률이 낮아진다"고 가정하여 "지수형 신뢰도 모델", "S자형 신뢰도 모델", "지수-S자형 신뢰도 모델" 등이 개발되어 왔다. 이 모델들의 누적 오류 발생 곡선의 형태를 살펴보면 (그림 2)의 a, b, c와 같은 형태의 곡선을 나타내는 것을 알 수 있다.

그러나 실제 테스트 기간 동안 발생되는 누적 오류 발생 곡선을 고찰해 보면 (그림 2)의 d와 같은 형태로 발생된다고 볼 수 있다. 즉, 테스트 초기에는 지수형 형태의 곡선을 나타내다가 어느 시점 이후부터는 발견되는 오류의 수가 지수형 곡선보다 크고, 다시 지수형 형태와 유사하게 나타나다가 또다시 어느 시점 이후부터 오류의 수가 증가한다는 것을 알 수 있다.

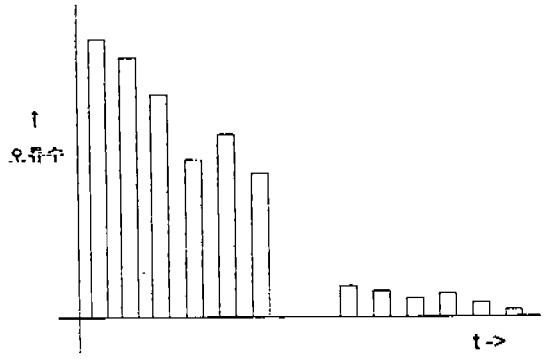
이러한 현상을 좀더 상세히 고찰하기 위해 〈표 1〉과 같은 실측 데이타를 각 시간별로 발견되는 오류수를 그래프로 정리해 보면 (그림 3)과 같다. 〈표 1〉과 (그림 2)에서 나타난 것과 같이 각 단위 CPU시간마다 발견되는 오류의 수가 점차 감소하다가 CPU시간 4-5사이에서는 발견되는 오류의 수가 증가하였고, 다시 점차 감소하다가 CPU시간 7-8사이에서는 발견되는 오류의 수가 증가한다는 것을 알 수 있다. 이후에



(그림 2) 소프트웨어에서 발견되는 누적 오류 성장 곡선
(Fig. 2) The cumulative errors Curve of SRGM's

〈표 1〉 CPU 시간에 의한 오류발견수 및 오류 누적수
〈Table 1〉 Number of detected error and the cumulative error by the CPU time

Hour	Number of Failures	Cumulative Failures	Hour	Number of Failures	Cumulative Failures
0	0	0	13	2	106
1	27	27	14	5	111
2	16	43	15	5	116
3	11	54	16	6	112
4	10	64	17	0	122
5	11	75	18	5	127
6	6	82	19	1	128
7	2	84	20	1	129
8	5	89	21	2	131
9	3	92	22	1	132
10	1	93	23	2	134
11	4	97	24	1	135
12	7	104	25	1	136



(그림 3) CPU시간별 오류 발견수
(Fig. 3) Number of detected error by the CPU time span

도 여러 곳에서 발견되는 오류의 수가 증가되었다가 다시 감소하는 현상을 고찰할 수 있다.

발견되는 오류의 수가 계속적으로 감소하는 것은 동일 종류의 오류가 지수형 형태로 발생하고, 어느 시점 이후부터 오류의 수가 증가하는 것은 새로운(중요한) 종류의 오류가 발생되는 시점으로 생각할 수 있다. 그리고 이러한 새로운 종류의 오류는 모두 3가지로 분류하여 고찰해 볼 수 있다. 이러한 관점하에 3종류의 오류와 발견되는 오류수의 변화시점을 이용한 새로운 소프트웨어 신뢰도 성장 모델의 개발이 가능하다고 생각한다.

그리고 1. 2절에 기술한 "클래스의 발견 난이도를 갖는 소프트웨어 오류를 고려한 지수-S자형 신뢰도 모델"은 실제 오류의 분류를 2종류로 한정하였으나, 이를 세분화시킬 필요가 있고, 또한 오류 발견 형태가 지수형에서 S자형으로 변화되는 시점을 클래스 2 오류의 비율로 산출하고 있기 때문에 정확한 시점이라고 할 수 없다.

따라서 본 논문에서는 위와 같은 문제점을 개선하기 위해 오류의 종류를 3종류로 분류하였고, 오류의 발견 과정을 고찰하여 각 오류의 발견 시점을 정형화하여 소프트웨어 신뢰도 성장 모델을 제안하였다.

세 종류의 오류중 첫번째 오류(클래스 1 오류)는 발견하기 쉬운 "오류로 처리 데이타나 실행환경에 관계없이 오류가 있는 루트를 실행하면 반드시 오류가 발생"하는 오류이다. 그리고 두번째 오류(클래스 2 오류)는 첫번째 오류보다 발견하기 어려운 오류로 "처리환경의 변화나 특이한 데이타의 처리에 의해 오류를 유발하는 오류"이다. 마지막으로 3번째 오류(클래스 3 오류)는 2번째 오류보다 발견하기 어려운 오류로 "다중처리 또는 동일 자원의 액세스 등에 의해서 오류가 발생되는 오류"이다.

그리고 이러한 세 종류의 오류는 테스트 초기부터 모두 발생되는 것이 아니라 테스트 초기에는 클래스 1 오류만 발생하고, 시간별 발견되는 오류의 수가 최초로 변화되는 시점, 즉 오류수가 시간에 따라 감소한다는 종래의 가정에 의해 각 시간별 발견되는 오류의 수는 $r_1 \rangle r_2 \rangle r_3 \cdots \rangle r_n$가 성립되어야 하지만, 실제 데이타를 관찰하면 $r_i \langle r_{i+1}$인 경우가 발생하는 시점 이후부터 클래스 1 오류, 클래스 2 오류가 발생하고, 또 다시 시간별 발견되는 오류의 수가 두번째로 변화되는 시점($r_i \langle r_i + 1$인 경우) 이후부터 클래스 1, 클래스 2, 클래스 3 오류 모두가 발견된다.

3. 오류 분류를 이용한 신뢰도 성장 모델

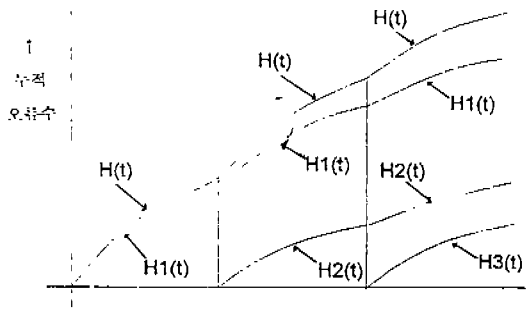
3.1 모델 유도

본 논문에서는 테스트 단계에서 발견되는 오류가 오류의 특성에 따라서 세가지(3 class)로 분류된다고 가정하였다. 그리고 각 오류가 발견되는 시점은 오류 수가 시간에 따라 감소한다는 종래의 가정에 의해 각 시간별 발견되는 오류의 수는 $r_1 \rangle r_2 \rangle r_3 \cdots \rangle r_n$가 성립되어야 하지만 (그림 4)와 같이 실제 데이타를 관찰하면 $r_i \langle r_{i+1}$인 경우가 발생한다. 이러한 현상을 오류의 분류도에 적용하여 각 오류 클래스가 발견되는 시점으로 설정한다. 즉,

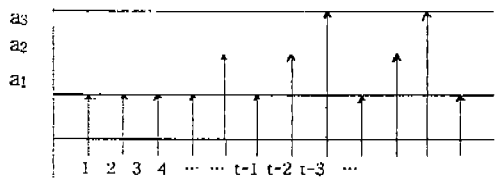
클래스 1오류 : 테스트를 실시하는 시점부터 발생.
클래스 2오류 : 발견되는 오류의 수가 최초로 $r_i \langle r_{i+1}$ 와 같은 현상이 나타 나는 시점부터 발생
클래스 3오류 : 발견되는 오류의 수가 두번째로 $r_i \langle r_{i+1}$ 와 같은 현상이 나타나는 시점부터 발생

본 논문의 소프트웨어 신뢰도 성장 모델의 각 오류 클래스가 시간적으로 발생되는 현상을 그래프로 살펴보면 (그림 5)와 같으며, 소프트웨어 신뢰도 성장 모델을 설정하기 위한 가정들을 정리하면 다음과 같다.

(가정 1) 테스팅시 발견되는 오류의 종류는 3가지이다.
(가정 2) 오류가 발견된 순간부터 그 오류가 수정되기까지의 기간에도 테스팅은 계속 진행된다.
(가정 3) 발견된 오류는 완전히 수정된다. 즉, 오류가 재발생되지 않는다.
(가정 4) 각 오류의 총기대 오류수 ai는 오류 분류도가 작을 수록 크고, 오류 분류도가 클수록 작다. 즉, $a_1 \rangle a_2 \rangle a_3 \rangle 0$
(가정 5) 각 클래스 오류의 오류 발견율 bi는 오류 분류도가 작을수록 크고, 오류 분류도가 클수록 작다. 즉, $b_1 \rangle b_2 \rangle b_3 \rangle 0$
(가정 6) 테스트 초기에는 클래스 1 오류만 발생. 즉, $a_1 \rangle 0$, $b_1 \rangle 0$
(가정 7) 각 시간별 발견되는 오류의 수가 최초로 $r_i \langle r_{i+1}$ 와 같은 현상이 나타날 때 클래스 1, 클래스 2 오류만 발생. 즉 $a_1 \rangle a_2 \rangle 0$, $b_1 \rangle b_2 \rangle 0$
(가정 8) 각 시간별 발견되는 오류의 수가 두번째로 $r_i \langle r_{i+1}$와 같은 현상이 나타날 때 클래스 1, 클래스 2, 클래스 3 오류 모두 발생. 즉, $a_1 \rangle a_2 \rangle a_3 \rangle 0$, $b_1 \rangle b_2 \rangle b_3 \rangle 0$



(그림 4) 각 시간별 오류 발견수
(Fig. 4) Number of detected error by the time span



(그림 5) 테스트 시간에 따라 발견되는 각 오류 클래스
(Fig. 5) Error-class of detected by the testing time

3.2 오류 분류를 이용한 신뢰도 모델

3.1절과 같은 가정하에서 임의의 테스트 시각 t까지 발견되는 오류수 M(t)에 대한 평균치 함수를 H(t)로 하면, 다음과 같은 소프트웨어 신뢰도 성장 모델로 수식화할 수 있다.

$$\Pr[M(t)=m]=\frac{\{H(t)\}^m}{m!}\times\exp[-H(t)]\quad m=0,\ 1,\ 2,\ 3,\ldots$$

$$H(t)=\int_0^t h(t)dt \qquad (2)$$

(식 2)의 평균치 함수 H(t)는 임의의 테스트 시각 t까지 발견되는 오류의 기대치를 나타낸다. 또한 h(t)은 강도함수(intensity function)라 하며 순간 오류 발견율을 나타낸다.

3.2.1 각 오류 클래스의 평균치 함수

이때, 각 오류 클래스가 발견되는 평균치 함수를 정의하면 다음과 같다.

(1) 클래스 1 오류

앞절에서 기술한 것과 같이 클래스 1 오류는 테스팅시 가장 발견하기 쉬운 오류로 처리 데이타나 실행환경에 관계없이 오류가 있는 루트를 실행하면 반드시 고장이 발생되는 오류이다. 그리고 클래스 1 오류는 테스트가 진행되는 초기부터 발생한다. 클래스 1 오류의 오류 발생형태는 지수형 곡선을 나타낸다고 가정한다. 이때 평균치 함수 Hi(t)에 대한 잔재 오류수에 대하여 오류 발견율이 테스트 기간을 통하여 일정하다고 가정하여 평균치 함수를 정의하면 다음과 같다.

$$\frac{dH_1(t)}{dt} = b_1 \times [a_1 - H_1(t)] \quad (a_1 > 0,\ b_1 > 0) \tag{3}$$

a_1 : 테스트 시작전에 소프트웨어내에 잔재하는 클래스 1 오류의 총기대 오류수

b_1 : 클래스 1 오류의 오류 1개당 발견율

(식 3)를 $H_1(t)$에 대하여 정리하면 (식 4)와 같다.

$$H_1(t) = a_1 \times [1 - \exp(-b_1 \times t)] \tag{4}$$

(2) 클래스 2 오류

소프트웨어를 테스팅하는 경우 테스트 입력뿐만 아니라 컴퓨터의 내부상황까지도 의존하여 소프트웨어 고장이 발생되는 경우가 있다. 예를 들면, 실행환경의 변화나 특정 데이타의 처리에 의해 고장이 발생되는 오류가 클래스 2 오류에 해당된다고 할 수 있다. 이러한 클래스 2 오류의 평균치 함수 $H_2(t)$에 관한 잔재 오류수에 대하여 오류 발견률이 테스팅 기간을 통하여 일정하고, 발견되는 누적 오류의 수가 지수형 형태로 발생한다고 가정하여 평균치 함수를 정의하면 다음과 같다.

$$\frac{dH_2(t)}{dt} = b_2 \times [a_2 - H_2(t)] \quad (a_1 > a_2 > 0,\ b_1 > b_2 > 0) \tag{5}$$

a_2 : 테스트 시작전에 소프트웨어내에 잔재하는 클래스 2 오류의 총기대 오류수

b_2 : 클래스 2 오류의 오류 발견율

(식 5)를 $H_2(t)$에 대하여 정리하면 다음과 같다.

$$H_2(t) = a_2 \times [1 - \exp(b_2 \times t)] \tag{6}$$

(3) 클래스 3 오류

소프트웨어 테스팅시 다중 병렬처리나 동일자원의 다중 액세스에 의한 병목현상 등의 오류를 유발하는 오류이다. 따라서 클래스 3 오류의 평균치 함수 $H_3(t)$에 관한 잔재 오류수에 대하여 오류 발견율이 테스트 기간을 통하여 일정하고, 발견되는 누적 오류의 수가 지수형 형태로 발생한다고 가정하여 평균치 함수를 정리하면 다음과 같다.

$$\frac{dH_3(t)}{dt} = b_3 \times [a_3 - H_3(t)] \quad (a_1 > a_2 > a_3 > 0,\ b_1 > b_2 > b_3 > 0) \tag{7}$$

a_3 : 테스트 시작전에 소프트웨어내에 잔재하는 클래스 3 오류의 총기대 오류수

b_3 : 클래스 3 오류의 오류 발견율

(식 7)을 H3(t)에 대하여 정리하면 다음과 같다.

$$H_3(t) = a_3 \times [1 - \exp(-b_3 \times t)] \tag{8}$$

3.2.2 소프트웨어 신뢰도 모델

앞절에서 클래스 1 오류, 클래스 2 오류 그리고 클래스 3 오류의 평균치 함수 $H_i(t)(i = 1, 2, 3)$에 대하여 기술하였다. 그리고 3종류의 오류를 하나의 확률로 처리하기 위해 각 평균치 함수를 하나의 평균치 함수로 혼합할 경우 계수 과정 $\{M(t), t \geq 0\}$는 3종류의 오류 $\{M_1(t), t \geq 0\}$, $\{M_2(t), t \geq 0\}$, $\{M_3(t), t \geq 0\}$를 하나로 혼합한 것으로, NHPP의 성질에 의해 $\{M(t), t \geq 0\}$와 같은 계수과정을 같는 평균치 함수를 H(t)로 하면,

$$\Pr[M(t) = m] = \frac{\{H(t)\}_m}{m!} \times \exp[-H_i(t)] \qquad m = 0, 1, 2, 3, \ldots \tag{9}$$

$$\begin{aligned} H(t) &= H_1(t) + H_2(t) + H_3(t) \\ &= a_1 \times (1 - \exp[-b_1 \times t]) + a_2 \times (1 - \exp[-b_2 \times t]) \\ &\quad + a_3 \times (1 - \exp[-b_3 \times t]) \end{aligned} \tag{10}$$

(1) 발견되는 시간별 오류의 수가 $r_i < r_{i+1}$ 이전까지

테스트 초기에 발생되는 각 시간별 오류의 수가 $r_i < r_{i+1}$인 경우 클래스 2, 클래스 3 오류는 발생되지 않고, 단지 클래스 1 오류만이 발생된다. 그리고 클래스 1 오류는 지수형 형태로 오류가 발견된다고 가정한다.

$$H(t) = H_1(t) = a_1 \times (1 - \exp(b_1 \times t)) \quad (11)$$

(2) 최초의 $r_i < r_{i+1}$부터 다음 오류 발견수가 $r_i < r_{i+1}$까지

소프트웨어 테스트가 진행됨에 따라 각 시간별 발견되는 오류의 수가 점차 감소하다가 최초로 $r_i < r_{i-1}$되는 시점부터 다음의 오류 발견수 $r_i < r_{i+1}$가 되는 시점까지 클래스 1, 클래스 2 오류가 발생된다. 그리고 클래스 1, 클래스 2 오류 모두 지수형 형태로 오류가 발생된다고 가정한다.

$$\begin{aligned} H(t) &= H_1(t) + H_2(t) \\ &= a_1 \times (1 - \exp[-b_1 \times t]) + a_2 \times (1 - \exp[-b_2 \times t]) \end{aligned} \quad (12)$$

(3) 발견되는 시간별 오류의 수가 두번째 $r_i < r_{i+1}$ 이상부터

소프트웨어 테스팅이 진행됨에 따라 각 시간별 발견되는 오류의 수가 점차 감소하다가 두번째로 $r_i < r_{i+1}$되는 시간부터 클래스 1, 클래스 2, 클래스 3 오류가 모두 발생된다. 그리고 클래스 1, 클래스 2, 클래스 3 오류 모두 지수형 형태로 오류가 발생된다고 가정한다.

$$\begin{aligned} H(t) &= H_1(t) + H_2(t) + H_3(t) \\ &= a_1 \times (1 - \exp(-b_1 \times t)) + a_2 \times (1 - \exp(-b_2 \times t)) \\ &\quad + a_3 \times (1 - \exp(-b_3 \times t)) \end{aligned} \quad (13)$$

3.2.3 기대 잔재 오류수 및 신뢰도 측정

평균치 함수로부터 임의의 테스트 시각 t에서 소프트웨어내의 기대잔재 오류수를 도출할 수 있다. 기대잔재 오류수를 측정하는 수식은 다음과 같다.

$$\begin{aligned} r(t) &= a - M(t) \\ &= a - (H_1(t) + H_2(t) + H_3(t)) \\ &= a - \{a_1 \times (1 - \exp(-b_1 \times t)) + a_2 \times (1 - \exp[-b_2 \times t]) \\ &\quad + a_3 \times (1 - \exp[-b_3 \times t])\} \end{aligned} \quad (14)$$

단, a_1: 최초로 오류가 발견되는 순간부터
a_2: 최초의 $r_i < r_{i+1}$인 경우부터
a_3: 두번째의 $r_i < r_{i+1}$인 경우부터

그리고 $M(t) = x_i$ 즉, 임의의 시각 t에서 x_i개의 오류가 발견되었다는 조건하에서 $(t, t+x](x \geq 0)$에서 오류가 발견되지 않을 확률의 신뢰도 수식으로 정리하면 다음과 같다.

$$\begin{aligned} R(x/t) = \exp[-\{&(a_1 \times (1 - \exp(-b_1 \times (t+x))) \\ &+ a_2 \times (1 - \exp(-b_2 \times (t+x))) \\ &+ a_3 \times (1 - \exp(-b_3 \times (t+x))) \\ &- (a_1 \times (1 - \exp(-b_1 \times t)) + a_2 \times (1 - \exp(-b_2 \times t)) \\ &+ a_3 \times (1 - \exp(-b_3 \times t)))\}] \end{aligned} \quad (15)$$

단, a_1: 최초로 오류가 발견되는 시점부터
a_2: 최초로 $r_i < r_{i+1}$인 경우가 발생되는 시점부터
a_3: 두번째로 $r_i < r_{i+1}$인 경우가 발생되는 시점부터

4. 파라메타 추정

실제 테스트 단계에서 (식 7)에 정의되어 있는 오류분류를 이용한 소프트웨어 신뢰도 성장 모델에 기초하여 소프트웨어의 신뢰도를 평가하는 경우에는 테스트 단계에서 발생되는 오류 데이타를 이용하여 모델 파라메타를 추정하여야 한다. 이때 파라메타는 (식 10)의 수식에 포함되어 있는 a_i, b_i $(i = 1, 2, 3)$로 본 논문에서는 최우법을 이용하여 파라메타를 추정한다.

일정한 테스트 시각 $t(t = 1, 2, 3, \ldots, n; 0 < 1 < 2 < \ldots < t)$까지 발견된 누적 오류수의 실측치를 $X_j(j = 1, 2, 3, \cdots n)$로 하면 실측 데이타에 대한 우도함수 L은 다음과 같은 수식으로 정리된다.

$$\begin{aligned} L &= \Pr[M(t_1) = y_1, M(t_2) = y_2, M(t_3) = y_3, \cdots\cdots, M(t_n) = y_n] \\ &= \prod_{k=1}^{n} \frac{\{H(t_k) - H(t_{k-1})\}^{y_k - y_{k-1}}}{(y_k - y_{k-1})!} \times \exp[-H(t_k) - H(t_{k-1}) \end{aligned}$$

단, $t_0 = 0$, $y_0 = 0$ (16)

그리고 (식 16)의 우도함수에 자연로그를 취하여 lnL을 구하면 다음과 같다.

$$\begin{aligned} \ln L = \sum_{k=1}^{n} &[(y_k - y_{k-1}) \ln [H(t_k) - H(t_{k-1})] \\ &- (H(t_k) - H(t_{k-1})) - \ln(y_k - y_{k-1})!] \end{aligned} \quad (17)$$

이때, 분류된 오류의 발견 시점에 따라 추정되어야 할 파라메타의 수가 변화되기 때문에 분류된 오류의 발견 시점에 따라서 파라메타를 추정해야 한다. 발견되는 시간별 오류의 수가 최초로 $r_i < r_{i+1}$가 성립되기 이전까지는 파라메타 a_1, b_1을 추정하고, 각 시간별 발견되는 오류의 수가 최초로 $r_i < r_{i+1}$인 경우부터 두번째의 $r_i < r_{i+1}$인 경우까지는 a_i, b_i(i=1, 2)의 파라메타를 추정해야 한다. 그리고 각 시간별 발견되는 오류의 수가 $r_i < r_{i+1}$와 같은 현상이 두번째로 발생되는 시간부터는 a_i, b_i(i=1, 2, 3)의 파라메타를 추정한다.

5. 수치 적용

본 논문에서 제안한 모델은 수치적인 적용 예를 보이기 위해 문헌 2에 수록된 데이타를 이용하였다. 이 데이타는 단위시간 즉, 실행(CPU)시간 1시간마다 발견된 오류의 수를 나타낸 것으로 〈표 2〉와 같으며, 평가 시점은 실행시간 4, 7, 25에서 본 논문의 모델과 Goel-Okumoto의 지수형 모델을 이용하였다.

〈표 2〉 실행시간동안의 오류 발견수 및 오류 누적수
〈Table 2〉 Number of detected error and cumulative error by the execution during time

Hour	Number of Failures	Cumulative Failures	Hour	Number of Failures	Cumulative Failures
0	0	0	13	2	106
1	27	27	14	5	111
2	16	43	15	5	116
3	11	54	16	6	122
4	10	64	17	0	122
5	11	75	18	5	127
6	6	82	19	1	128
7	2	84	20	1	129
8	5	89	21	2	131
9	3	92	22	1	132
10	1	93	23	2	134
11	4	97	24	1	135
12	7	104	25	1	136

각 시점에서 파라메타를 추정해 보면 오류가 발견되는 시점부터 $r_i < r_{i+1}$ 이전까지 본 논문에서 제안하는 모델은 $a_1=149$ $b_1=0.15$, 지수형 모델은 $a_1=149$, $b_1=0.15$로 동일하다. 그리고 최초의 $r_i < r_{i+1}$부터 다음 오류 발견수가 $r_i < r_{i+1}$까지 본 논문에서 제안하는 모델은 $a_1=146$, $b_1=0.152$, $a_2=1$, $b_2=0.033$ 지수형 모델은 $a_1=146$, $b_1=0.152$이다. 또한, 시간별 오류수가 두번째 $r_i < r_{i+1}$ 이후부터 본 논문에서 제안하는 모델은 $a_1=138$, $b_1=0.13$, $a_2=7$, $b_2=0.041$, $a_3=2$ $b_3=0.002$이고, 지수형 모델에서는 $a_1=141$ $b_1=0.125$의 파라메타가 추정되었다.

각 시점별 파라메타에 의한 평균치 함수를 살펴보면 다음과 같다.

1) 오류가 발견되는 시점부터 $r_i < r_{i+1}$ 이전까지

$$H(t)=149\times(1-\exp(-0.150\times t))$$

2) 최초의 $r_i < r_{i+1}$부터 다음 오류 발견수가 $r_i < r_{i+1}$까지

$$H(t)=146\times(1-\exp(-0.152\times t)+1\times(1-\exp(-0.033\times t))$$

3) 시간별 오류수가 두번째 $r_i < r_{i+1}$ 이후부터

$$H(t)=138\times(1-\exp(-0.130\times t))+7\times(1-\exp(-0.041\times t))2\times(1-\exp(-0.002\times t))$$

그리고, 추정된 평균치 함수 $H(n_i)$를 이용하여, 기대 잔재 오류수($r(n_i)$)를 다음과 같이 추정한다.

$$r(n_i)=147-\{138\times(1-\exp(-0.130\times t))+7\times(1-\exp(-0.041\times t))2\times(1-\exp(-0.002\times t))\} \quad (18)$$

기대 잔재 오류수를 이용하여 신뢰도($RG(n_i)$)를 구하면 〈표 3〉과 같은 신뢰도를 얻을 수 있다.

$$R(n,h)=\exp(-(\{138\times(1-\exp(-0.130\times(t+h)))+7\times(1-\exp(-0.041\times(t+h)))+2\times(1-\exp(-0.002\times(t+h)))\}-\{138\times(1-\exp(-0.130\times t))+7\times(1-\exp(-0.041\times t))\}) \quad (19)$$

〈표 3〉 테스트 데이타에 대한 신뢰도($RG(n_i)$)
〈Table 3〉 Reliability about the test data

t	0	1	2	3	4	5	6	7	8	9	10	11	12
H(t)	0	16	31	44	55	66	75	83	90	96	101	106	110
$\hat{RG}()$	0	0	0	0	0.0	0.0	0.0001	0.0003	0.0009	0.002	0.004	0.008	0.014

t	13	14	15	16	17	18	19	20	21	22	23	24	25
H(t)	114	118	120	123	125	127	129	131	132	133	134	135	136
$\hat{RG}()$	0.024	0.038	0.055	0.078	0.10	0.13	0.17	0.21	0.25	0.29	0.33	0.38	0.42

6. 결 론

소프트웨어 개발 단계중 테스트 단계에서 발견되는 소프트웨어 오류 데이타를 이용하여 소프트웨어의 신뢰도를 평가할 때에는 오류 데이타의 발생 형태/경향에 따라서 오류 데이타를 해석하는 것이 매우 중요하다.

본 논문에서는 종래의 신뢰도 모델들에 대하여 조사 검토하였고, 또한 소프트웨어의 오류 발생 형태/경향을 고찰하여 「테스트 초기에는 오류발견이 쉽고 테스트 시간이 경과함에 따라서 오류발견이 어렵다」는 것을 고찰하였다. 그리고 본 논문에서 고려한 가정과 유사한 모델을 검토하여 이 모델에서 개선되어야 할 문제점을 분석하여 개선된 소프트웨어 신뢰도 모델을 제안하였다. 또한 종래의 지수형 신뢰도 모델과 비교해 본 결과 본 논문에서 제안하는 모델이 적합함을 알 수 있었다.

향후 연구과제로서는 본 모델에서 설정한 가정들을 개선한 새로운 모델의 개발이 필요하다. 그리고 오류 데이타를 이용한 소프트웨어 신뢰도 모델뿐만 아니라 소프트웨어를 분석하는 신뢰도 모델의 개발이 필요하며, 또한 하드웨어 오류와 소프트웨어 오류를 동시에 고려한 신뢰도 모델의 개발이 필요하다.

참 고 문 헌

[1] Abdalla A. Abdel-Ghaly, "Evaluation of Competing Software Reliability Prediction", IEEE Trans. SE. Vol. SE-12, No. 9,pp. 950-967, 1986.

[2] AMRIT L. Goel, "Software Reliabilty Models: Assumption, Limitations, and applicability", IEEE Trans. SE. Vol. SE-11, No. 12, pp. 1411-1423, 1985.

[3] John C. Knight and Nancy G. Leveson, "An Experimental Evaluation of the Assumption f Independence in Multiversion Programming", IEEE Trans. Vol. SE-12, No. 1, pp. 96-109, 1986.

[4] Koch, H. S. and Kubat, P.:Optimal Release Time of Computer Software, IEEE Trans. SE., Vol. SE-9, No. 3, pp. 323-327, 1983.

[5] Norman F. Schneidewind, Fellow, "Software Reliability Model with Optimal Selection of Failure Data", IEEE Trans. SE., Vol. SE-19, No. 11, pp. 1095-1104. 1993.

[6] Ramamoorth, C. V. and Bastani, F. B.:Software Reliablity Status and Perspectives, IEEE Trans. SE., Vol. SE-8, No. 4, pp. 351-371, 1982

[7] Yamada, S. and Osaki, S:Software Relibility Growth Modeling:Models and Applications, IEEE Trans. SE.,Vol. SE-11, No. 12, pp. 1431-1437, 1985.

[8] "ソフトウェア 信賴性 モテルと評價", 日本電子情報通信學會, Vol. 73, No. 5, pp. 461-466, 1990.

[9] "應用ソフトウェア 信賴性 技術", 日本電子情報通信學會, Vol. 73, No. 5, pp. 49 2-496, 1990.

[10] "運用段階におけるソフトウェア信賴性評價に關する考察", 電子情 報通信學會, vol. J72-D-I, No. 11, pp. 797-803,1989.

[11] 中川農, 竹中市郎, "エラ-の 複雜度に 基づく ソフトウェア 信賴度モデル" vol. J74-D-I, No. 6,

pp. 379-385, 1989.

[12] 木村光宏, 山田茂, "2クラスの發見難易度をもつソフトウェア エラ-を考慮た 指數-S字形 信賴度成長モデルの 考察", 日本情報處 理學會論文誌, Vol. 33, No. 11, pp. 1446-1452, 1992.

[13] 우치수, 김갑수, "내장 소프트웨어에서의 신뢰도와 고장률 예측", 한국정보과학회 논문지, Vol. 21, 1호, pp. 117-127, 1994

조 영 식

1992년 강원대학교 자연과학대학 전자계산학과 졸업(학사)

1995년 강원대학교 전자계산학과 소프트웨어공학 전공(석사)

1994년 한림전문대학 전산정보처리과 강사

1996년~현재 남서울신업대학교 전자계산학과 강사

1995년~현재 강원대학교 대학원 전자계산학과 박사과정

관심분야: 소프트웨어공학(특히, S/W 품질보증과 품질평가, 소프트웨어 신뢰도, 소프트웨어 재사용)

이 용 근

1988년 강원대학교 자연과학대학 전자계산학과 졸업(학사)

1994년 강원대학교 전자계산학과 소프트웨어공학 전공(석사)

1989년~1992년 강원대학교 전자계산학과 조교

1995년~현재 강원대학교 대학원 전자계산학과 박사과정

1996년~현재 경희대학교 전자계산학과 강사

관심분야: 소프트웨어공학(특히, S/W 품질보증과 품질평가, 객체지향프로그래밍, 객체지향분석과 설계방법)

최 형 진

1982년 영남대학교 물리학과 졸업(학사)

1987년 日本 동경공업대학 정보공학과(공학석사)

1990년 日本 동경공업대학 정보공학과 인공지능 전공(공학박사)

1990년~91년 한국전자통신 연구소 선임연구원

1991년~현재 강원대학교 전자계산학과 조교수

관심분야: 인공지능, 화상처리, 패턴인식, 컴퓨터 비젼, 컴퓨터 그래픽 등

양 해 술

1975년 홍익대학교 공과대학 전기공학과 졸업(학사)

1978년 성균관대학교 정보처리학과 정보처리전공(석사)

1991년 日本 오사카대학교 기초공학부 대학원 정보공학과 소프트웨어공학 전공(공학박사)

1975년~1979년 육군중앙경리단 전자계산실 시스템 분석장교

1986년~1987년 日本 오사카대학교 객원연구원

1993년~1994년 한국정보과학회 학회지 편집부위원장

1980년~1995년 강원대학교 전자계산학과 교수

1994년~1995년 한국정보처리학회 논문지 편집위원장

1994년~현재 한국산업표준원(IIS) 이사

1995년~현재 한국소프트웨어품질연구소(INSQ) 소장

관심분야: 소프트웨어 공학(특히, S/W 품질보증과 품질평가 SA/SD, OOA/OOP, CASE, SI), 소프트웨어 프로젝트관리