[5B-105]

SRGM과 그의 파라미터 산출 방법

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

SRGM and its Parameter Calculation Method

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

요약

지난 수십년간 많은 SRGM이 제안되었다. 이러한 모델들 중 많은 부분에서 암암리에 소프트웨어 테스트 전 단계를 거쳐서 테스트노력이 상수인 것으로 가정하거나 또는 아예 고려하지도 않았다. 그 후 몇몇 논문을 통하여 테스트노력을 고려한 소프트웨어의 신뢰도 평가가 중요한 인자인 것으로 발표되었다. 이들에 의해 지금까지 제안된 형태를 보면 지수함수형, 레일레이형, 웨이블형, 로지스틱형 테스트노력함수로서 경우에 따라 이 중 하나의 적합한 형태를 사용해왔다.

본 논문에서는 이 네 가지 형태에 대해서 최소자승평가자(LSE)와 최대가능성평가자(MLE)를 써서 신뢰도 성장 파라미터를 구하는 방법에 대해서 고찰하고, 실제의 데이터를 적용하여 각각의 경우에 대한 파라미터를 구하고 이를 이용하여 실제와 비교 연구한다.

1. 서론

소프트웨어 테스트 단계 기간 동안 소프트웨어의 신뢰도는 잠재 소프트웨어 결함을 검출하고 수정하는데에 소요되는 개발자원의 양에 전적으로 의존한다.

야마다 등[]은 역일시간, 테스트 노력량, 테스트 노력에 의하여 검출되는 결함의 수 사이의 관계를 명시적으로 기술하는 새롭고도 단순한 소프트웨어 신뢰도 성장모델을 제안하였다. 이는 그들이 소프트웨어 성장에의 테스트 노력의 영향과 관련된 소프트웨어 신뢰도 성장 모델을 제안한 것이다. 그들은 시간종속적 레일레이와 지수형 곡선을 이용한 테스트노력소요량 거동을 기술한다. 많은 연구가들의 연구에서는 테스트 단계의 테스트 자원의 소모율은 일정한 것으로 가정하거나 또는 그리한 테스트 노력을 고려하지도 않았다. 여러 참고 문헌에서 그 노력지수(실행시간)가 소프트웨어 신뢰도 모델링에서 역일시간보다 더 좋다는 것을 보여주고 있다. 관찰된 신뢰도 성장곡선의 형상이 테스트 노력의 시간분포에 강하게 의존하기 때문이다. 또 여러 참고문헌에서는 역일테스트, 테스트 노력량, 테스트 노력에 의하여 검출되는 결함의 수 사이의 관계를 설명하는 SRGM을 제안하였다.

소프트웨어 신뢰도 모델링 분야에서는 그 노력이 자주 전통적인 지수함수, 레일레이함수 또는 웨이블곡선으로 기술된다. 그러나, 많은 소프트웨어 테스트 환경에서 이러한 3개의 노력함수곡선만으로 소프트웨어 테스트 노력 함수를 기술하는 것은 어려운 일이다. 따라서, 본 논문에서는 로지스틱 테스트 노력까지를 고려한 파라미터 산출법을 연구한다.

2항에서는 문헌에있는 기존의 테스트 노력 함수를 기술하고 추이 곡선을 그렸으며, 3항에서는 4개의 제안된 함수에 대한 파라미터를 구하는 방법을 제시하였다. 여기에 최소 자승 평가자와 최대 가능성 함수를 도입하였다. 4항에서는 실제 현장에서 구한 결함검출 데이터를 이용하여 각각의 파라미터를 구하고 이를 실제 현상과 비교하였다.

2. 테스트 노력 함수

주어진 기간에 맞추어 소프트웨어 시스템을 개발할 때 시간, 자금, 인력과 같은 자원들이 소요된다. 특히, 소프트웨어 테스트에까지 이르는 자원들이 소프트웨어 신뢰도에 상당한 영향을 미친다. 소프트웨어 개발 자원 전체의 약 40-50%가 테스트 단계에서 소요된다. Yamada 등[3]은 테스트 기간중의 테스트 노력과 소프트웨어 개발 노력 모두가 레일레이 곡선이나 웨이블곡선으로 설명될 수 있다는 것을 가정하여 소프트웨어 테스트에 쓰이는 테스트노력의 양을 고려한 소프트웨어 신뢰도 성장 모델을 제시하였다. 그들은 레일레이 곡선의 대안으로서 지수곡선도 제안하였다.

2.1 평균치 함수 및 신뢰도

$R(x|t)$는 시각 t에서 최종적으로 결함을 발견하여 수정한 후 x 유니트의 경과시간동안 새로운 결함이 발견되지 않을 확률이다. 소프트웨어를 개발하여 결함테스트를 하면 할수록 결함을 발견하여 수정하는 빈도가 작아지므로 신뢰도가 성장되며, 결함 수정 후 경과시간이 길어지면 길수록 결함 발견 확률이 높아지기 때문에 소프트웨어의 신뢰도는 낮아진다. 한편, 테스트 단계에서는 얼마나 오랜 시간동안 결함이 발견되지 않느냐가 중요한 것이 아니라, 현 단계에서 소프트웨어 내에서 발견되지 않고 잔존하는 결함의 수가 얼마나 되는가가 더 중요하다.

소프트웨어를 시각 t에서 발행하는 경우, 발견되는 누적 분포는 평균치 함수

$$m(t)=\alpha(1-e^{-\gamma W(t)}) \tag{2.1}$$

이므로, 잔여 분포

$$\bar{a}=a-\alpha(1-e^{-\gamma W(t)})=a\cdot e^{-\gamma W(t)}$$

이다. 이것이 소프트웨어를 발행해서 운전하는 초기 결함수이므로

$$m(t+x)=a\cdot e^{-\gamma W(t)}\cdot(1-e^{-\gamma W(x)})+m(t)$$
$$=\alpha(1-e^{-\gamma W(t)-\gamma W(x)}) \tag{2.2}$$

이므로,

$$R(x|t)=\exp[-m(t+x)+m(t)]$$
$$=\exp[-a\cdot e^{-\gamma W(t)}(1-e^{-\gamma W(x)})] \tag{2.3}$$

이 된다.