소프트웨어 신뢰성 평가 도구 분석

김국현, 백종문
한국과학기술원 전산학과
e-mail: {kimegoo, jbaik}@kaist.ac.kr

An Analysis on Software Reliability Evaluation Tools

Gookhyun Kim, Jongmoon Baik
Dept. of Computer Science, KAIST

요 약

소프트웨어가 점점 복잡해지면서 신뢰할 수 있는 소프트웨어의 개발에 대한 필요성이 제기되고 있다. 이에 따라 소프트웨어 개발 업체는 소프트웨어 신뢰성 보장을 위한 많은 활동들을 수행하고 있다. 이 과정에서 소프트웨어 신뢰성 평가는 핵심이 되는 작업 중 하나이며, 다양한 소프트웨어 신뢰성 평가 도구가 개발되어 정확하고 효율적인 신뢰성 평가를 돕고 있다. 소프트웨어 신뢰성 평가 도구는 적용할 수 있는 소프트웨어 개발 단계와 적용 방법에 차이가 있기 때문에 도구들은 적시적소에 적용되어야 한다.

본 논문에서는 CASRE, SMERFS, SREPT, GERT, SRTPRO 와 같은 소프트웨어 신뢰성 평가 도구의 분석을 통해 각 도구들의 특징, 목적, 적용단계 등을 고려하여 사용자가 다양한 도구 중 어떤 도구를 선택해야 하는지 판단하는데 도움을 주고자 한다.

1. 서론

최근 소프트웨어가 대형화되고 복잡해 짐에 따라 소프트웨어 개발 프로세스 관리가 어려워지고 있으며, 신뢰할 수 있는 소프트웨어의 개발이 중요한 문제로 부각되고 있다.

이와 같은 상황에서 소프트웨어 신뢰성이 신뢰할 수 있는 소프트웨어를 개발하는데 중요한 품질 특성으로 부각되고 있다. 소프트웨어 신뢰성이란 특정한 환경에서 특정한 기간 동안 소프트웨어가 고장(Failure) 없이 동작하는 확률을 말한다 [1]. 소프트웨어 신뢰성 목표를 설정하고 개발 과정에서 신뢰성을 평가함으로써 프로젝트가 요구되는 신뢰성 목표를 만족하는지 평가할 수 있다.

소프트웨어 신뢰성은 신뢰성 평가 도구를 이용하여 평가된다. 각 도구들은 고장 데이터 (Failure Data), 소프트웨어 아키텍처 등을 입력 받아 다양한 소프트웨어 신뢰성 모델에 적용한다. 이로써 현재의 소프트웨어 신뢰성의 평가는 물론, 앞으로의 신뢰성이 어떻게 변화될지에 대한 예측이 가능하다. 따라서 신뢰성 목표를 달성하기 위하여 품질 보증 활동을 정확히 수행하고 있는지, 언제까지 소프트웨어 신뢰성을 높이는 작업을 수행해야 하는 지와 같은 다양한 정보를 분석할 수 있다.

소프트웨어 신뢰성 도구들은 신뢰성을 평가하는 방법이 다르기 때문에 도구를 적용할 수 있는 시기나 방법에 있어서 차이가 있다. 따라서 소프트웨어 개발 시 한 개의 도구만을 사용하는 것은 정확한 신뢰성 평가를 어렵게 한다. 결국 소프트웨어 신뢰성 평가 도구들의 입출력과 지원하는 신뢰성 평가 모델, 적용시킬 수 있는 개발 단계를 이해하여 적시적소에 알맞은 도구를 선택하는 것이 중요하다.

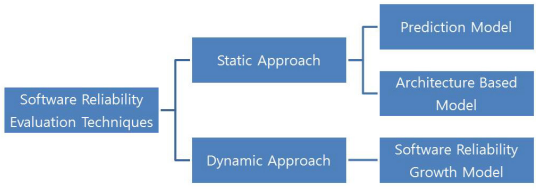
본 논문에서는 지금까지 개발되었던 소프트웨어 신뢰성 평가 도구들을 분석하여 소프트웨어 개발 시 어떤 도구를 어느 시점에 적용할 수 있는지에 대한 판단을 돕는 것을 목표로 한다.

논문 또는 저서는 2010 년 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(한국연구재단-2010-C1AAA001-2010-001014375)

2. 소프트웨어 신뢰성 평가 방법

소프트웨어 신뢰성 평가 도구는 여러 평가 모델을 이용하기 때문에, 소프트웨어 신뢰성 평가 도구의 특성을 이해하기 위해서 각 도구가 이용하는 평가 모델을 충분히 이해해야 한다. 따라서 본 장에서 각 모델에서 사용된 소프트웨어 신뢰성 평가 모델에 대한 기본 지식을 전달하고자 한다.

1970 년 이후로 다양한 소프트웨어 신뢰성 평가 모델이 제안되었다. 소프트웨어 신뢰성을 평가 방법은 크게 정적인 방법과 동적인 방법으로 나눌 수 있다.



(그림 1) 소프트웨어 신뢰성 평가 방법의 분류

정적 방법은 소프트웨어 아키텍처나 다양한 소프트웨어 척도[9-11]를 이용한다. 이로써 소프트웨어 구현 전 신뢰성을 평가할 수 있다. 하지만 동적 모델에 비하여 이용되는 정보의 부족으로 인해 신뢰성의 정확도가 떨어진다는 단점을 가지고 있다.

반면 동적 방법에서는 구현된 소프트웨어 시스템으로 운용 혹은 테스트를 수행하면서 신뢰성을 평가하게 된다. 소프트웨어 신뢰성 성장 모델 [8]은 이러한 방법으로 신뢰성을 평가하는 가장 대표적인 사례다. 이 모델들은 소프트웨어의 실행 중 발생하는 고장 데이터 (Failure Data)를 분석하여 현재의 신뢰성을 추정한다. 또한 테스트 척도 [8]와 같은 자료를 이용하여 신뢰성을 평가하기도 한다. 이처럼 동적 방법은 구현된 소프트웨어 시스템으로부터 신뢰성을 평가하기 때문에 정적 방법보다 정확도가 높지만, 구현 후 신뢰성을 평가하여 시스템을 다시 수정해야 하기 때문에 추가적인 비용이 발생한다.

3. 소프트웨어 신뢰성 도구 분석

소프트웨어 신뢰성 도구 분석에 앞서, 각 도구의 평가를 위한 다음과 같은 기준이 필요하다.

1) 입력과 출력: 사용자의 입장에서 입력과 출력은 매우 중요한 역할을 한다. 현재 이용 가능한 산출물(Artifact)에서 입력이 가능한지 그리고 원하는 결과를 얻을 수 있는지 확인할 필요가 있다. 예를 들어 현재 소프트웨어 아키텍처가 있을 때, 앞으로 설명할 SREPT 의 경우 아키텍처를 입력 받기 때문에 이 단계에서는 SRPET 가 가장 적합한 도구로서 선정될 수 있다.
2) 지원하는 소프트웨어 신뢰성 평가 방식: 도구에서 지원하는 평가 모델을 안다면 현재 프로젝트와 해당 모델의 적합성을 검토할 수 있다. 예를 들어 NHPP 모델을 지원하면 구현 이후 단계부터 신뢰성을 평가해야 하는 프로젝트에 적용해야 할 것이다.
3) 도구를 적용할 수 있는 소프트웨어 개발 단계: 사용자는 도구를 적용할 수 있는 소프트웨어 개발 단계를 파악하여 적용하고자 하는 시점에 적용 가능한지 확인해야 한다
4) 사용성: 사용성이 뛰어난 도구는 신뢰성 평가 작업을 효율적으로 진행시키는데 도움을 주기 때문에 중요하다.

소프트웨어 신뢰성 예측을 위해 다양한 소프트웨어 신뢰성 평가 도구가 개발되었다. 본 논문에서는 그 중 가장 대표적으로 사용되었던 4 개의 도구와, 이 중 일부의 단점을 보완한 1 개의 도구를 분석한다. 그리고 각각의 도구는 소프트웨어 신뢰성 평가 방법을 정확하게 구현하여 각 도구에서 제시하는 신뢰성 수치는 같은 모델 간에 차이가 없다고 가정한다. 또한 각 도구는 최신 버전으로 분석했음을 밝힌다.

3.1 CASRE[3]

CASRE (Computer Aided Software Reliability Estimation)는 소프트웨어 신뢰성 성장 모델을 이용하여 신뢰성을 평가하는 도구이다. 초창기에 개발된 도구로서 다음과 같은 특징을 갖는다:

1) 고장간 시간 (Time Between Failure, TBF) 또는 고장 수 (Failure Counts, FC)데이터 파일을 입력으로 받는다. 하지만 직접 수치를 입력할 수 없기 때문에 사용성이 떨어진다는 단점을 가지고 있다. 입력에 따라서 <표 1>과 같은 결과를 그래프로 얻을 수 있고 프린터 출력도 가능하다.
2) TBF 를 입력 받는 경우, Geometric, Jelinski-Mornada, Littlewood-Verrall, Musa-Basic, Musa-Okumoto, NHPP [8] 와 같은 신뢰성 성장 모델을 적용할 수 있다. FC 를 입력 받는 경우 Generalized Poisson, NHPP, Schneidewind, Shick-Wolverton, Yamada S-Shaped [8] 신뢰성 성장 모델을 적용할 수 있다.
3) CASRE 는 TBF 와 FC 의 값이 입력으로 필요하기 때문에 시스템 테스트, 수용 테스트, 운용 단계에 적용할 수 있다.
4) 데이터 입력은 파일로만 가능하며 직접 입력하지 못하기 때문에 사용성이 떨어진다. 그리고 데이터 윈도우와 결과 그래프 윈도우가 분리되어 있어 다소 산만한 느낌을 준다. 또한 메뉴 구성과 인터페이스는 직관적이지 못하고, 신뢰성에 대한 전문 지식이 없으면 사용하기가 어렵다.

TBF data	***FC data***
	Failure counts
	Test interval lengths
Failure Intensity	Failure Intensity
Times Between Failures	Times between failures
Cumulative Number of Failures	Cumulative number of failures from modeling start point
Reliability	Reliability

<표 1> CASRE 의 입력에 따른 결과

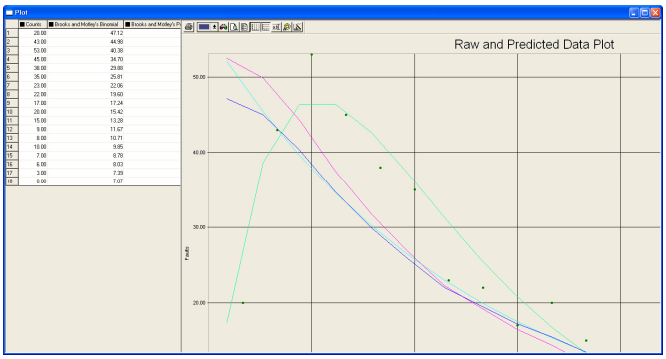
3.2 SMERFS[2]

SMERFS(Statistical Modeling and Estimation of Reliability Functions for Software)는 CASRE 와 마찬가지로 초창기에 개발된 소프트웨어 신뢰성 평가 도구로서 많은 공통점을 가진다. SMERFS 역시 소프트웨어 신뢰성 성장 모델을 이용하여 신뢰성을 평가한다.

1) SMERFS 는 TBF 나 FC 데이터를 입력 받는다. CASRE 와의 차이점은 파일 입력 이외에도 사용자가 직접 수치를 입력할 수 있다는 점이다. 이렇게 입력된 수치와 각 모델을 적용한 결과를 그래프로 출력하고 인쇄가 가능하다.

2) SMERFS 는 TBF 입력의 경우 Geometric, Jelinski-Mornada, Littlewood-Verrall, Musa's Basic, Musa's Logarithmic, NHPP [8] 모델을 지원한다. 또한 FC 의 경우 Brook's and Motely's 모델, Generalized Poisson, NHPP, Schneidewind, Yamada S-shaped [8] 모델을 지원한다.

3) SMERFS 는 CASRE 와 마찬가지로 소프트웨어의 테스트 단계 혹은 운용 단계에서 적용할 수 있다.
4) SMERFS 의 사용성은 CASRE 에 비해 초보자도 쉽게 이용할 수 있고 직관적이다. 하지만 입력된 데이터를 프로그램 상에서 조작하지 못한다는 단점을 가지고 있다.



(그림 2) SMERFS 를 이용한 신뢰성 추정

3.3 SRTPRO[7]

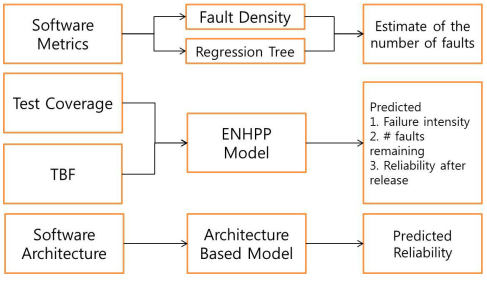
SRTPRO 는 CASRE 와 SMERFS 의 단점을 보완한 신뢰성 평가 도구이다 SRTPRO 는 신뢰성 추정뿐만 아니라 신뢰성 예측을 위한 평가 모델을 제공한다. 또한 CASRE 와 SMERFS 가 도구 내에서 데이터를 변경하지 못했던 단점을 보완하였다.

1) SRTPRO 는 CASRE 나 SMERFS 와 마찬가지로 TBF 나 FC 와 같은 데이터를 입력 받을 수 있다. 뿐만 아니라 예측 모델을 위한 몇 가지 척도를 입력하기도 한다. 이러한 입력을 통해 신뢰성 추정 및 예측에 관한 데이터 수치를 얻을 수 있다. 또한 고장 강도 (Failure Intensity), 누적 고장수 (Cumulative Failure)와 같은 다양한 결과를 그래프로 출력하고, 이 결과를 HTML 문서로 리포트 하거나 인쇄하는 것이 가능하다.
2) SRTPRO 는 TBF 입력의 경우 Musa Okumoto Model, Musa Basic Model, Jelinski Moranda Model, Littlewood and Verrall Linear Model, Littlewood and Verrall Quadratic Model, Non-homogeneous Poisson Model [8]을 지원한다. FC 입력의 경우 NHPP, Schneidewind Model, Yamada S-Shaped Model [8]을 지원한다. 그리고 예측을 위해 RL-TR-92-52 Model, Musa Basic Method, COQUALMO [7]와 같은 모델을 지원한다.
3) SRTPRO 는 신뢰성 예측 모델을 지원하기 때문에 CASRE 와 SMURFS 와 다르게 소프트웨어 개발 주기 초반부터 후반에 이르는 광범위한 주기에 적용할 수 있다는 장점을 가진다.
4) SRTPRO 는 기존의 도구가 가지고 있는 사용상의 불편함을 경감시켰다. 도구 내에서의 데이터 조정, 한 화면에 다양한 추정 결과를 그래프로 볼 수 있는 기능들이 그것이다. 또한 초보자도 쉽게 이용할 수 있는 직관적인 인터페이스를 갖추고 있다.

3.4 SREPT[5]

SREPT (Software Reliability Estimation and Prediction Tool)는 소프트웨어 성장 모델, 척도뿐만 아니라 소프트웨어 아키텍처 기반 신뢰성 평가 모델까지 지원해주는 도구이다. 따라서 이 도구는 특정 개발 프로세스에 한정되지 않고 전 영역에 걸쳐 적용할 수 있다. SREPT 는 다음과 같은 특징을 가진다.

1) 이 도구는 현재의 결함을 추정하기 위해 소프트웨어 척도를 입력 받는다. 그리고 고장 강도 (Failure Intensity), 잔여 결함 수, 출시 후 신뢰성 등을 알기 위해 테스트 커버리지, TBF 등이 입력으로 사용된다. 또한 프로젝트 초반에 신뢰성 예측을 위해 소프트웨어 아키텍처를 입력으로 받는다.
2) SREPT 에서 결함 수를 추정하기 위하여 고장 밀도 (Fault Density)와 회귀 트리 (Regression Tree) 모델이 사용된다. 또한 고장 강도, 잔여 결함 수, 출시 후 신뢰성 등을 예측하기 위해 ENHPP 모델이 사용된다. 마지막으로 아키텍처를 이용하는 아키텍처 기반 신뢰성 모델이 사용된다 [13].
3) SREPT 는 다양한 모델을 지원함으로써 소프트웨어 개발 주기 전반에 걸쳐 적용할 수 있다.
4) SREPT 는 각각의 모델을 적용하는데 있어서 쉽고 직관적인 인터페이스를 제공한다. 하지만 지원하는 모델의 종류가 많아 각각을 적용하기 위해서는 소프트웨어 신뢰성에 대한 전반적인 지식이 필요하다. 따라서 초보자가 이용하기에는 어려움을 겪을 수 있다.



(그림 3) SREPT 의 입력, 모델, 출력[5]

3.5 GERT[6]

GERT("Good Enough" Reliability Tool)는 TDD (Test-Driven Development) 환경에 특화된 소프트웨어 신뢰성 평가 도구다. 지금까지 설명한 다른 도구들과는 달리, Eclipse 의 플러그인 형태로 존재한다.

1) GERT 의 입력은 JUnit 테스팅 프레임워크를 위한 테스트케이스와 소스코드이다. 이러한 입력에 대해 신뢰성 모델에 따른 결과가 화면에 수치로서 표현된다.
2) GERT 는 STREW[12]에서 제시한 척도를 통해 소프트웨어의 신뢰성을 예측한다.
3) GERT 는 테스트 케이스와 소스 코드가 구현되어 있어야 한다. 따라서 구현 이후에 적용이 가능하다.
4) Eclipse IDE 플러그인 형태로 제작되어 있기 때문에 프로그램 개발자는 구현을 하면서 신뢰도의 변화를 바로 확인할 수 있다. 또한 인터페이스 역시 매우 쉽고 직관적으로 구성되어 있다.

GERT 는 사용하기 쉽고 직관적인 인터페이스를 갖추고 있지만 구현 이후에 적용이 가능하다는 점과 지원하는 신뢰성 평가 모델이 부족하다는 단점을 가지고 있다.

지금까지 분석한 소프트웨어 신뢰성 평가 도구들을 정리하면 <표 2>와 같다. CASRE 와 SMERFS 는 입력으로 고장 데이터를 받고, 검증과 유지 보수 단계에 적용이 가능하다. SRTRO 는 고장 데이터와 척도를 입력으로 하고, 소프트웨어 개발 전 단계에 걸쳐 적용이 가능하다. SREPT 는 고장 데이터, 척도, 아키텍처를 입력으로 하여 전 단계에 걸쳐 적용이 가능하다. GERT 는 척도를 입력바다 구현, 검증 그리고 유지보수 단계에 적용이 가능하다.

	CASRE & SMERFS	SRTPRO	SREPT	GERT
Failure Data	√	√	√	
Metrics		√	√	√
Architecture			√	
Requirement		√	√	
Design		√	√	
Implementation		√	√	√
Verification	√	√	√	√
Maintenance	√	√	√	√
Usability	Low	High	Middle	High

<표 2> 소프트웨어 신뢰성 평가 도구 비교

4. 결론

본 논문에서는 입출력, 지원하는 신뢰성 평가 모델, 적용 가능한 개발 주기, 사용성 등을 고려한 몇 가지 대표적인 소프트웨어 신뢰성 평가 도구들의 특징을 분석 하였다. 본 논문에서 제시한 분석 자료와 정확한 소프트웨어 신뢰성 모델에 대한 이해를 바탕으로 실무에서 적절한 소프트웨어 신뢰성 평가 도구를 선정하고 소프트웨어 신뢰성을 평가할 수 있을 것이다.

또한, 본 논문의 분석을 통해 향후 소프트웨어 신뢰성 평가 도구가 나아가야 할 방향을 알 수 있다. 미래의 소프트웨어 신뢰성 평가 도구는 전반적인 소프트웨어 개발 단계에 걸쳐 적용할 수 있도록 다양한 모델이 제시되어야 한다. 또한 사용성을 높여 신뢰성 보증 작업이 보다 더 쉽고 효율적으로 동작할 수 있도록 개선되어야 할 것이다.

참고문헌

[1] ANSI/IEEE, "Standard Glossary of Software Engineering Termonology", STD-729-1991, ANSI/IEEE, 1991

[2] William H. Farr, Oliver D. Smith, "A tool for statistical modeling and estimation of reliability functions for software: SMERFS", Journal of Systems and Software, 1988

[3] Lyu, M.R., Nikora, A. "CASRE: a computer-aided software reliability estimation tool", Computer-Aided Software Engineering, 1992

[4] Naixin Li, Malaiya, Y.K , "ROBUST: a next generation software reliability engineering tool," Software Reliability Engineering, 1995

[5] Trivedi, K.S., "SREPT: a tool for Software Reliability Estimation and Prediction," Dependable Systems and Networks, 2002.

[6] Davidsson, M. et al, "GERT: an empirical reliability estimation and testing feedback tool," Software Reliability Engineering, 2004. ISSRE 2004.

[7] Myungmuk Kang, "A User Friendly Software Reliability Analysis Tool based on Development Process to Iteratively Manage Software Reliability", International Symposium on Software Reliability Engineering, 2009.

[8] W.H.Farr. "A survey of software reliability modeling and estimation", NJaval Surface Weapons Center, Sept. 1983

[9] Linda Rosenberg, et al, "Software Metrics and Reliability," ISSRE 1998

[10] S. Henry and C.Selig. Predicting source-code complexity at the design stage. IEEE 1990

[11] S. Krishnamurthy and A.P. Mathur, "On the Esitmation of Reliability of a Software System Using Reliabilities of its Components", ISSRE 1997

[12] N. Nagappan, Wiolliams, L., Vouk, M.A., "Toward a Metric Suite for Early Software Reliability Assessment", ISSRE 2003

[13] Swapna S. Gokhale; , "Architecture-Based Software Reliability Analysis: Overview and Limitations," Dependable and Secure Computing, 2007