Journal of the Korea Academia-Industrial cooperation Society
Vol. 21, No. 6 pp. 178-186, 2020

https://doi.org/10.5762/KAIS.2020.21.6.178
ISSN 1975-4701 / eISSN 2288-4688

국내 무기체계 분야의 소프트웨어 신뢰성 추정 모델 적용 사례

박다운
국방과학연구소 정보화기술실

An Application of Software Reliability Estimation Model on Weapon System

Da-Un Bak
Information System & Common Technology Office, Agency for Defense Development

요 약 국내 무기체계 연구개발과정에서 소프트웨어 신뢰성이 중요한 요소로 여겨지고 있다. 그래서 무기체계 소프트웨어 연구개발 절차에는 소프트웨어 신뢰성 향상을 위한 활동들이 포함되어 있다. 하지만 개발절차에 포함된 활동은 소스코드 정적 및 동적 분석으로 국제 표준에서 요구하는 활동과 다소 차이가 존재한다. 소프트웨어 신뢰성 관련 국제 표준인 IEEE std 1633-2016에서는 소프트웨어 신뢰성 확보를 위한 프로세스를 정의하고 있으며, 이들 가운데 소프트웨어 신뢰성 추정을 필수 활동이라고 이야기하고 있다. 소프트웨어 신뢰성 추정은 시험단계의 결함을 기반으로 통계 모델을 활용해 현재 시점의 소프트웨어 신뢰성을 추정하는 활동이다. 추정한 모델을 기반으로 소프트웨어 고장률을 추정할 수 있으며, 목표 고장률과의 비교를 통해 시험 종료 여부를 결정할 수 있다. 따라서 본 연구에서는 무기체계 소프트웨어 개발과정에 소프트웨어 신뢰성 추정 모델을 적용하였다. 그 결과 목표한 소프트웨어 신뢰성을 달성하기 위해 지속적인 시험이 진행되었으며, 정량적인 소프트웨어의 신뢰성을 확인 할 수 있었다. 본 연구를 기반으로 무기체계 소프트웨어의 개발 과정에서 국제 표준에서 제시한 소프트웨어 신뢰성 공학 프로세스를 반영하는 노력이 지속적으로 이루어지기를 기대한다.

Abstract In the domain of Korean weapon system development, issues about software reliability have become crucial factors when developing a weapon system. There is a process required for weapon system software development and management that includes certain activities required to improve the reliability of software. However, these activities are biased toward static and dynamic analyses of source code and do not include activities necessarily required by the international standard. IEEE std. 1633-2016 defines a process for software reliability engineering and describes software reliability estimation as an essential activity in the process. Software reliability estimation means that collecting defective data during the test and estimating software reliability by using the statistical model. Based on the estimated model, developers could estimate the failure rate and make comparisons with the objective failure rate to determine termination of the test. In this study, we collected defective data and applied reliability estimation models to analyze software reliability in the development of a weapon system. To achieve objective software reliability, we continuously tested our software and quantitatively calculated software reliability. Through the research, we hope that efforts to include activities described by the international standard will be carried out in the domain of Korean weapon system development.

Keywords : Software Reliability Engineering, Korean Weapon System Software Development, Reliability Estimation Model, Software Reliability during Testing, Release Decision

*Corresponding Author : Da-Un Bak(Agency for Defense Development)
email: dwpark90@gmail.com
Received March 2, 2020 Revised March 25, 2020
Accepted June 5, 2020 Published June 30, 2020

1. 서론

소프트웨어 고장은 마모에 의해 발생하는 하드웨어 고장과는 달리 코드에 내재 되어있는 잠재적 결함에 의해 발생한다[1]. 즉 코드의 수정이 없는 한 개발된 소프트웨어의 신뢰성은 사용과 무관하게 동일하다고 볼 수 있다. 따라서 배포 전에 반복적인 시험을 통해 소프트웨어 결함을 식별하고 수정해야 한다.

하지만 소프트웨어 신뢰성 향상을 위해 개발기간을 무제한으로 계획할 수 없다. 언젠가는 소프트웨어 배포 여부를 결정해야 한다. 이와 같은 결정 시에 개발과정에서 추정한 소프트웨어 신뢰성을 활용할 수 있다[2]. 소프트웨어 신뢰성은 특정 시점과 환경에서 소프트웨어가 결함 없이 정상적인 동작을 수행할 확률을 의미하며, 개발 환경의 특성 또는 통계 모델을 이용해 계산될 수 있다. 신뢰성과 관련된 연구는 소프트웨어 신뢰성 공학 분야에서 활발히 진행되고 있으며, 최근에 소프트웨어로 구현되는 기능이 복잡하고 거대해짐에 따라 많은 관심을 받고 있다[1][2].

국내 무기체계 소프트웨어 연구개발에서도 소프트웨어 신뢰성에 대한 이슈가 점차 부각되고 있다. 우선 무기체계 소프트웨어 개발 과정은 방위사업청에서 제정한 무기체계 소프트웨어 개발 및 관리 매뉴얼을 따른다[3]. 해당 매뉴얼에는 개발 프로세스와 소프트웨어 품질 확보를 위한 개발 단계별 활동이 정의되어있다. 개발단계별 활동에는 소프트웨어 신뢰성 확보를 위한 활동도 포함되어 있다.

하지만 매뉴얼에 제시된 소프트웨어 신뢰성 관련 활동들은 국제 표준에서 요구하는 활동과 차이가 있다. 먼저 매뉴얼에서 제시하고 있는 활동은 코드의 정적 및 동적 분석을 강조하고 있다[3]. 반면에 국제 표준에서는 개발 전 주기 동안 소프트웨어 신뢰성 관련 활동을 수행하고, 특히 시험 단계에서 발견된 결함의 추이를 필수적으로 분석할 것을 요구하고 있다[2]. 이를 위해 표준에서는 여러 통계 모델들을 제시하고 있으며, 분석의 결과는 소프트웨어 배포 여부를 결정하는 기준으로 사용된다.

따라서 본 연구에서는 무기체계 소프트웨어 개발 과정에 소프트웨어 신뢰성 추정과 관련된 활동을 시범 적용해보고 그 결과를 서술하고자 한다. 무기체계 소프트웨어 개발과정에 국제 표준에서 제시한 신뢰성 추정 활동이 추가된다면 보다 고품질의 소프트웨어 개발이 가능할 것이라 기대된다.

본 논문의 장절은 다음과 같다. 1장에서는 연구에 대한 개략적인 설명을 하였으며, 2장에서는 국제 표준에서 정의한 소프트웨어 신뢰성 공학 절차와 국내 무기체계 분야의 소프트웨어 신뢰성 관련 활동에 대해 정리한다. 다음으로 3장에서는 무기체계 분야에 소프트웨어 추정 모델 적용 방안을 제시하고, 4장에서 소프트웨어 신뢰성 추정 모델 시범 적용 결과를 서술한다. 마지막으로 5장에서 결론을 서술한다.

2. 소프트웨어 신뢰성 공학

2.1 소프트웨어 신뢰성 공학 프로세스

IEEE std. 1633-2016은 소프트웨어 신뢰성과 관련된 대표적인 국제 표준으로 소프트웨어 신뢰성 공학의 프로세스와 단계 별 세부 활동들을 명시하고 있다[2].

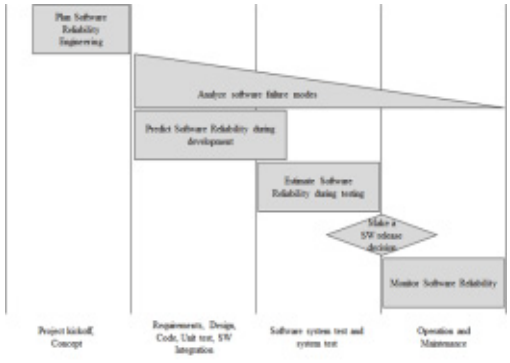

Fig. 1. Software reliability engineering process in IEEE std.1633-2016 [2]

소프트웨어 신뢰성공학 프로세스는 Fig. 1과 같이 6단계로 구분된다. 첫 번째는 계획 단계로, 개발 환경 및 프로젝트 자원의 가용 수준을 기반으로 소프트웨어 신뢰성 적용 범위를 결정한다. 예를 들어 개발되는 소프트웨어의 물자표(Bill of material)를 작성해야 한다. 이를 통해 소프트웨어의 구조와 하위 연관 관계를 파악할 수 있다. 다음으로 발생할 수 있는 소프트웨어 실패에 대한 분석을 진행한다. 이때 소프트웨어 결함의 주요 원인 분석과 시스템 영향도 분석 등이 수행된다. 3번째 단계는 신뢰성 예측을 수행해야 한다. 신뢰성 예측은 테스트 이전 단계에 수집한 정보(예: 개발환경, 개발언어 등)를 이용하여 소프트웨어의 결함 밀도를 예측하는 활동이다. 다음 단계는 소프트웨어 신뢰성 추정으로, 시험 단계에서 발생한 실제 소프트웨어 결함을 기반으로 통계적 모델을 활용해

현재 시점의 소프트웨어 고장률을 추정하는 활동이다. 5번째 단계에서는 앞서 획득한 신뢰성 예측과 추정 결과를 기반으로 소프트웨어 배포 시점에 대한 판단을 수행해야 한다. 마지막 6번째 단계에서는 실제 운영 단계에서 획득한 고장률과 개발과정에서 예측 및 추정한 고장률 간의 비교 등을 수행해야 한다.

2.2 소프트웨어 신뢰성 추정

국제 표준에서는 프로세스의 4번째 단계인 신뢰성 추정을 신뢰성 공학의 필수 활동들 가운데 하나라고 이야기하고 있다[2]. 그 이유는 추정된 모델 기반으로 소프트웨어 고장률을 추정할 수 있기 때문이다. 그리고 소프트웨어 고장률은 이전 단계에서 수립한 목표 고장률과의 비교를 통해 소프트웨어 배포 결정의 근거가 된다. 만일 추정된 고장률이 목표 수준보다 낮은 경우 개발된 소프트웨어가 목표 수준의 신뢰성을 확보했다고 판단하고 시험을 종료 할 수 있다. 하지만 그 반대의 경우에는 소프트웨어 신뢰성 향상을 위해 추가적인 시험을 수행해야 한다.

모델 추정을 위해서는 시험단계에서 발견된 소프트웨어 결함이 사용된다. 여러 해 동안의 연구를 통해 분석된 소프트웨어 결함의 발견 추세는 Fig. 2와 같다[2]. 일반적으로 시험 초기에 계속적으로 증가하다가, 특정시점에 최고점을 달성한 후 감소하는 흐름을 가진다. 시험 초기에는 소프트웨어가 안정화되지 않았기 때문에 결함이 지속적으로 발견되지만, 어느 정도 시간이 경과한 후에는 내재되어 있는 결함이 감소하여 추가적으로 식별되는 결함이 감소하는 것이다.

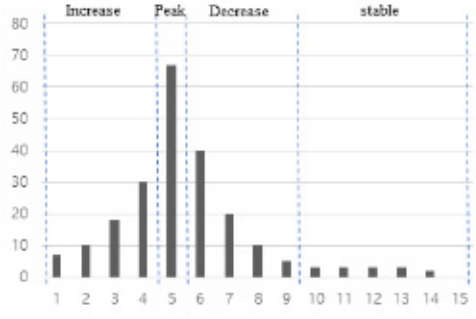

Fig. 2. Distribution of software defect during test phase [2]

Fig. 2와 같은 추세를 기반으로 표준에서는 Table 1과 같이 여러 신뢰성 추정 모델들을 제시하고 있다. 제시된 모델들은 공통적으로 몇 가지 가정을 하고 있는데 대표적으로 발견된 결함은 즉시 제거되며 제거된 결함으로 인한 새로운 결함은 발생하지 않는다는 것이다[2]. 반면에 각 모델들의 결함 발생 추이에 대한 가정은 서로 상이하다. 예를 들어 Goel-Okumoto 모델은 결함이 시간의 경과에 따라 직선형으로 감소한다고 가정하는 반면[6], Log-Logistic 모델은 결함의 감소 추세를 비선형으로 가정하고 있다[9]. 또한, 모델에 따라 추정을 위해 필요한 데이터가 상이하다. 대부분의 모델은 단위 시간 당 발견된 결함수를 기반으로 추정할 수 있지만, 일부 모델들은 결함이 발견된 정확한 시간과 같이 더 자세한 데이터를 요구한다. 따라서 추정 모델 적용 시에는 시험 단계에서 발견된 소프트웨어 결함의 추이를 보고 적절한 가정을 내포하고 있는 모델을 적용해야 하며, 획득이 불가능한 데이터를 요구하는 모델은 제외해야 한다.

Table 1. Software reliability estimation models suggested in IEEE 1633-2016 [2]

Categories	Models
Increasing fault rate	Weibull [4]
Peak	Shooman Constant Defect Removal Rate Model [5]
Decreasing fault rate	Shooman Constant Defect Removal Rate Model [5] Goel-Okumoto [6] Musa Basic Model [7] Jelinski-Moranda [8] Shooman Linearly Decreasing Model [9] Duane [10] Musa-Okumoto [11] Shooman Exponentially Decreasing Model [9] Log-Logistic [12] Geometric [13]
Increasing and then decreasing	Yamada (delayed) S-shaped [14] Weibull [4]

2.3 무기체계 분야 신뢰성 확보 활동 현황

무기체계 내에 탑재되는 소프트웨어의 연구 및 개발 시에는 무기체계 소프트웨어 개발 및 관리 매뉴얼을 준수해야한다[3]. 해당 매뉴얼은 방위사업청에서 제정한 것으로 무기체계 소프트웨어 개발 절차와 개발 단계별 세부 업무에 대한 지침을 제공하고 있다. 매뉴얼에서는 다양한 활동을 요구하고 있는데, 그 중의 하나가 소프트웨어 신뢰성 향상을 위한 신뢰성 시험이다.

신뢰성 시험은 소프트웨어 통합 시험 단계에서 요구되는 활동으로 소프트웨어가 잠재적으로 유발할 수 있는 결함을 식별하는 시험이다. 신뢰성 시험은 정적시험과 동적시험으로 구분된다. 우선 정적 시험은 소프트웨어를 실행하지 않은 상태에서 잠재적인 결함을 검출하는 것으로

코딩규칙 검증, 취약점점검 및 소스코드 메트릭을 포함하고 있다. 다음으로 동적시험은 소프트웨어를 실제 하드웨어에 탑재한 상태에서 시험절차에 따라 코드 실행률을 점검하는 것을 의미한다.

무기체계 소프트웨어 개발과정에서 요구하는 신뢰성 관련 활동들은 국제 표준에서 중요시 하고 있는 활동들과는 차이가 있다. 우선 무기체계 소프트웨어 개발과정에서는 코드 기반의 정적, 동적시험을 강조하고 있다. 하지만 국제표준에서는 매뉴얼에서 제시한 활동을 개발 프로세스 단계를 넘어가기 위한 전이 조건 정도로 간주하고 있으며 [2], 소프트웨어 신뢰성 향상을 위해서는 신뢰성 추정을 요구하고 있다.

이와 같이 매뉴얼에서 신뢰성 추정과 관련된 활동을 필수적으로 요구하고 있지 않다보니, 무기체계 분야에서는 신뢰성 추정과 관련된 연구가 일부 제한적으로 수행되고 있다[15]. 해당 연구에서는 국방 분야의 소프트웨어 신뢰성 확보를 위한 프로세스 부재를 지적하며, 신뢰성 공학 프로세스 적용 사례를 제시하였다. 그리고 프로세스 적용의 일환으로 한 가지 추정 모델을 이용해 소프트웨어 신뢰성을 추정하였다.

3. 무기체계 분야 소프트웨어 신뢰성 추정모델 적용 계획

3.1 신뢰성 추정 모델 적용 절차

본 연구에서는 국제 표준에서 명시하고 있는 신뢰성 추정 모델을 무기체계 소프트웨어 개발과정에 시범 적용하고자 한다. 신뢰성 추정 모델 적용 절차는 Fig. 3과 같다. 먼저 시험 종료와 소프트웨어 배포 시점 결정의 기준이 되는 목표 고장률을 결정해야 한다. 목표 고장률 설정 방법은 유사 소프트웨어 존재 여부에 따라 달라질 수 있다. 만일 기 배포된 유사 소프트웨어가 존재하는 경우 해당 소프트웨어의 고장률을 기반으로 목표 고장률을 결정할 수 있다. 반면에 유사 소프트웨어가 존재하지 않는 경우에는 설계 및 개발 단계에서 예측한 소프트웨어의 신뢰성을 활용할 수 있다[2].

다음으로 소프트웨어 통합 시험 단계에서 발생한 결함을 수집해야 한다. 결함 수집 시에는 결함의 내용, 시간 등이 기록되어야 한다. 그리고 수집된 결함은 추후 모델 적용을 위해 운영시간을 정규화 한 후 단위시간 당 결함 수로 변환되어야 한다.

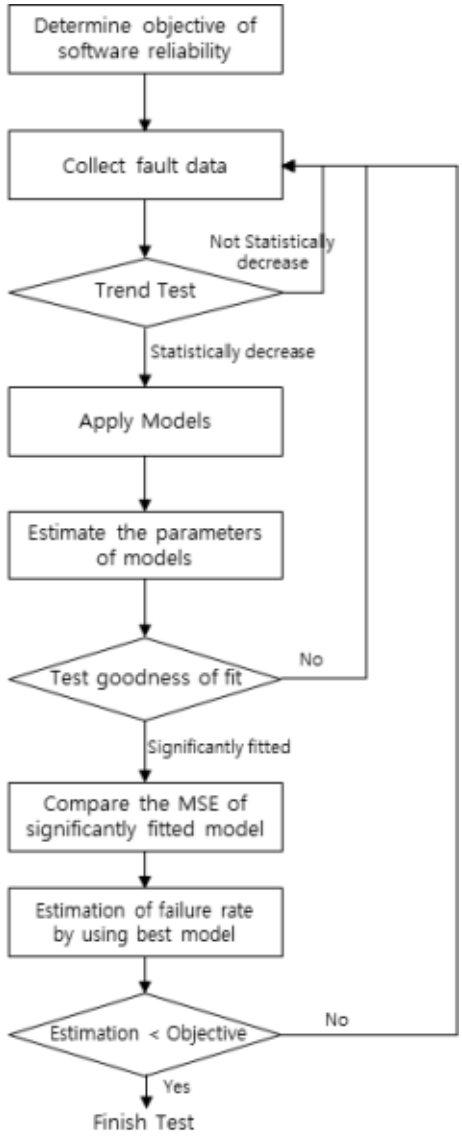

Fig. 3. Application procedure for software reliability estimation model

결함이 수집된 이후에는 결함 추이에 대한 분석을 진행해야 한다. 모델 적용에 앞서 결함의 추이 분석이 필요한 이유는 결함의 추이와 모델의 가정이 상이하다면 추정 결과가 의미가 있다고 보기 힘들기 때문이다[16]. 결함 추이 분석을 위해 graphical test, analytical test가 존재하는데, 본 연구에는 analytical test에서 가장 널리 활용되는 라플라스 시험을 사용하고자 한다[16].

라플라스 시험의 결과는 음수 일 때 결함의 추세가 감소(신뢰성 증가)한다고 보고, 양수 일 때 결함의 추세가

증가(신뢰성 감소) 한다고 본다[16]. 또한, 결과 값은 +2와 -2 값을 기준으로 유의수준 5%이내에서 추이가 통계적으로 유의미하다고 본다. 따라서 라플라스 시험의 결과가 -2이하일 때부터 결함이 통계적으로 유의미하게 감소하고 있다고 보고 앞서 소프트웨어 결함 발견률이 시간에 따라 감소한다는 가정을 내포하고 있는 모델들을 적용할 수 있다.

다음으로 라플라스 시험 결과가 기준을 충족한다면, 단위 시간 당 결함 수를 활용하여 모델의 파라미터 값을 추정 할 수 있다. 파라미터 추정 방법으로는 LSE(Least Square Method), MLE(Maximum Likelihood Estimation)가 보편적으로 사용된다[1][17]. 우선 LSE 방법은 모델의 예측값과 실제 결함의 차이를 제곱하여 더한 값을 최소화하는 파라미터를 추정하는 방법이다. 반면에 MLE는 각 모델에서 가정하는 분포와 실제 결과의 분포가 일치할 확률을 가장 높게 하는 파라미터를 추정하는 방법이다. 기존 연구들에 따르면 MLE의 추정 결과가 편향 될 가능성이 높다고 하여 본 연구에서는 파라미터 추정을 위해 LSE 방법을 사용하고자 한다[17].

모델의 파라미터 추정 후에는 모델의 적합성(Goodness of fit)을 검정해야 한다. 모델의 적합성 검정이란 추정한 모델이 결함의 분포를 잘 설명하는지 확인하는 활동이다. 카이제곱 검정을 통해 추정된 모델의 통계적 적합성을 검정할 수 있다 [18]. 만일 특정 유의 수준의 카이제곱 값보다 결과 값이 작은 경우, 해당 모델이 현재의 결함 분포를 통계적으로 유의미하게 표현한다고 볼 수 있다. 기준이 되는 카이제곱 값 선정을 위해 본 연구에서는 유의수준을 95%로 설정하였다. 만일 적합한 모델이 존재하지 않는 경우에는 추가적인 시험을 통해 결함을 수집해야 한다.

다음으로 통계적으로 유의미한 분포들 중에 가장 결함의 분포를 잘 설명하는 모델을 선정해야 한다. 모델간의 비교는 MSE(Mean Square Error)값 비교를 통해 가능하다 [19]. MSE는 LSE방법을 이용해 추정된 모델들의 우위를 비교할 때 사용되는 대표적인 방법이다. 가장 낮은 MSE 값을 가지는 모델이 현재의 결함을 가장 잘 설명한다고 볼 수 있다 [19]. 따라서 카이제곱검정 결과 복수의 모델이 통계적으로 유의미한 경우, MSE값 비교를 통해 가장 적합한 모델을 결정할 수 있다.

마지막으로 가장 분포를 잘 설명하는 모델을 활용하여 현재시점의 고장률을 계산 할 수 있다 [2]. 만일 추정된 고장률이 목표 고장률 보다 낮은 경우, 개발자는 시험을 종료하고 소프트웨어를 배포할 수 있다.

3.2 신뢰성 추정 모델 선정

IEEE-1633에서 여러 추정 모델을 제시하고 있지만, 본 연구에서는 결함을 추정하는 모델로 아래 Table 2와 같이 6가지 모델을 선정하였다. 모델 선정 시에 다음과 같은 사항들이 고려되었다. 우선 시간이 지날수록 결함수가 증가하는 모델이나 특정 시점에 결함이 최고치로 발견되는 형태를 구현한 모델은 결함의 추세는 알 수 있으나 잔존 결함을 추정하는 것이 불가능하다[2]. 따라서 본 연구에서는 시간에 따라 결함이 감소하는 모델과 결함 증가 후 감소하는 모델들을 추정 모델로 고려하였다.

Table 2. List of candidate model for software reliability estimation

	Model	Assumption about failure rate	Input Data
1	Jelinski-Moranda	Linearly Decreasing	Failure Count Data
2	Goel-Okumoto	Linearly Decreasing	
3	Yamada(delayed) S-shaped	Increasing and then decreasing	
4	Inflection-S	Increasing and then decreasing	
5	Log-Logistic	Non-Linearly Decreasing	
6	Weibull	Increasing and then decreasing	

다음으로 모델 추정 시에 단위 시간당 발견된 결함수를 기반으로 추정할 수 있는가를 고려하였다. 대부분의 모델들은 단위시간 당 발생한 결함 수를 이용하여 모델의 파라미터를 추정할 수 있지만, 일부 모델들은 보다 정교한 데이터를 요구한다. 예를 들어, Duane, Geometric 모델에서는 결함이 발견된 정확한 시간을 필요로 한다 [10][13]. 따라서 데이터 획득의 용이성을 고려하여 해당 모델들을 제외하였다.

마지막으로 표준에서는 감소 추이를 선형(Linearly)과 비선형(Non-Linearly)으로 구분하고 있다. 본 연구에서는 선형과 비선형 모두 분석 대상에 포함될 수 있도록 고려하였다. 다만 Inflection-S 모델은 표준에서 명시되지 않았지만 여러 연구들에서 검증된 모델로 시범 적용의 다양성을 위해 포함시켰다[20].

4. 소프트웨어 신뢰성 추정모델 적용 결과

4.1 소프트웨어 신뢰성 모델 적용 사업 특성

소프트웨어 신뢰성 추정 모델의 적용 대상 과제는 별도의 하드웨어 없이 윈도우 기반으로 운영되는 소프트웨어를 개발하는 사업이다. 해당 소프트웨어는 총 7개의 모듈로 구성되어 있으며, 모듈에 따라 C++/C#/Java 등으로 구현되었다. 또한, 개발 코드 라인 수는 약 200만 라인이다.

4.2 소프트웨어 신뢰성 모델 적용 결과

우선 소프트웨어 신뢰성 목표는 유사 소프트웨어의 고장률을 참고하여 결정하였다. 12명의 국방 분야소프트웨어 신뢰성 업무 관련 관계자들에게 적용 대상 소프트웨어의 운용개념을 설명한 후, 설문을 통해 총 11개의 유사 소프트웨어를 식별하였다.

그리고 식별된 소프트웨어의 유사도와 고장률, 목표 고장률을 델파이 기법 기반으로 추정하였다. 델파이 기법은 설문을 통해 정보를 수집하고, 수집된 결과를 참여자들에게 공유한다[21]. 그리고 참여자들 간의 합의점이 도출될 때까지 조사를 진행한다.

델파이 조사에는 유사 소프트웨어의 유사도를 묻는 순서형 질문과 유사소프트웨어를 사용했을 때 사용기간과 고장률을 묻는 수치형 질문이 포함되었다. 그리고 수렴된 유사도와 고장률의 평균, 표준편차, 중간 값을 설문 참여자에게 제공한 후 개발 중인 소프트웨어의 목표 결함률을 조사하였다. 그 결과 해당 소프트웨어의 목표 고장률은 0.012(고장/시간)으로 설정되었다.

다음으로 시험단계에서 발생한 결함 정보를 수집하였다. 일관성 있는 결함 데이터 수집을 위해 개발 조직 내에 시험 관련 담당자를 별도로 지정하였다. 식별된 결함은 모델 적용을 위해 하루 8시간 당 결함수로 정리되었다. 그 결과는 다음 Fig. 4와 Table 3과 같다. 발견된 결함은 최대한 바로 수정되었으며, 시간이 경과함에 따라 결함 수가 줄어드는 것을 확인 할 수 있다.

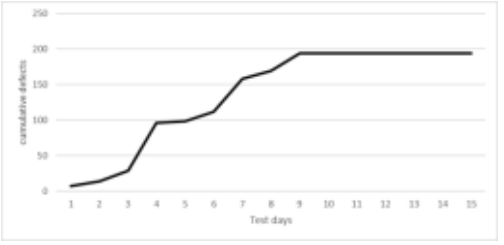

Fig. 4. Distribution of software defect during test

Table 3. Data for software defect during test

Test Round	Number of Failure	Number of Cumulative Failure
1	7	7
2	7	14
3	15	29
4	67	96
5	2	98
6	14	112
7	46	158
8	11	169
9	25	194
10	0	194
11	0	194
12	0	194
13	0	194
14	0	194
15	0	194

다음으로 신뢰성 모델의 파라미터 추정에 앞서 라플라스 시험으로 결함의 추이를 분석하였다. 그 결과 Fig. 5와 같이 초기에는 결함이 증가하다가, 11회차부터 라플라스 시험의 값이 -2이하로 내려갔다. 앞서 3장에서 언급했듯이 결과가 -2이하인 경우 결함의 추이가 통계적으로 유의미하게 감소하고 있는 것으로 볼 수 있다. 따라서 본 연구에서는 11번째 라운드부터 6개의 추정 모델들의 파라미터값을 추정하였다. 그리고 추정 시에는 자체 연구 개발한 프로그램을 활용하였다[22].

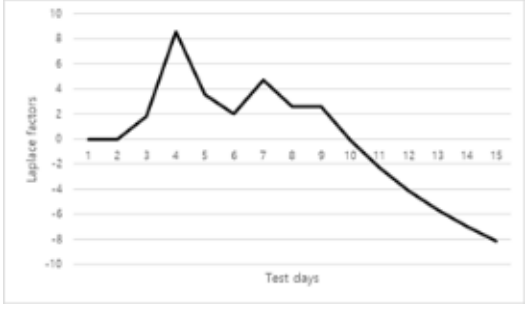

Fig. 5. Result of Laplace Test

다음으로 추정한 모델들의 적합도 검증을 수행하였다. 카이제곱 검정의 유의 수준은 95%로 설정했으며, 그 결과는 Table 4와 같다. 모델 적합도 검증 결과가 95% 유의수준의 카이제곱값(Critical value)보다 작은 경우, 해당 모델이 현재의 결함분포를 통계적으로 유의미하게 표현한다고 볼 수 있다. 그 결과 11번째, 12번째 테스트 라운드에서는 Log-Logstic 모델만이 적합한 모델인 것으로 확인되었으며, 13, 14, 15번째에는 Inflection-S 모

델과 Weibull 모델이 추가적으로 적합한 것으로 확인되었다.

Table 4. Result of the chi-square test for each model

Test Round	Jelinski-Moranda	Goel-Okumoto	Yamada(delayed) S-shaped	Inflection-S	Log-Logistic	Weibull	Critical value (95%)
11	1994.8	115.20	35.80	23.00	**18.20**	18.60	18.31
12	135.60	3312.9	42.60	22.20	**18.50**	3141.1	19.68
13	161.30	170.40	48.90	21.72	**19.00**	**18.20**	21.03
14	203.83	194.73	54.41	**21.40**	**19.45**	3380.4	22.36
15	267.66	216.42	59.09	**21.18**	**19.90**	**18.15**	21.40

다음으로 통계적으로 유의미한 모델들 가운데 가장 최적의 모델을 식별하기 위해 각 모델들의 MSE 값을 비교하였다. 먼저 11번째와 12번째에는 Log-Logstic 모델만이 적합성 검증을 통과하여 별도로 MSE 분석을 수행 할 필요가 없었다. 다음으로 복수의 모델이 적합한 것으로 나타난 13, 14, 15번째 시험 라운드의 MSE 결과 값은 Table 5와 같다. 각 모델들의 MSE값을 비교한 결과 13번째, 14번째, 15번째 시험 차수에서 각각 Weibull, Inflection-S, Weibull 분포가 가장 적합한 모델인 것으로 확인되었다.

Table 5. Result of MSE for each model

Test Round	Models	MSE
13	Log-Logistic	114.02
	Weibull	102.72
14	Log-Logistic	111.86
	Inflection-S	105.17
15	Log-Logistic	108.50
	Inflection-S	98.99
	Weibull	90.29

마지막으로 가장 적합한 모델을 기반으로 소프트웨어의 고장률을 Table 6과 같이 추정하였다. 우선 14번째 라운드 이전까지는 목표 고장률(0.012) 보다 높은 고장률이 추정되어 시험을 종료하지 못했다. 반면에 15번째 Weibull 모델의 소프트웨어 고장률이 0.007로 추정되어 목표 고장률보다 낮은 것으로 확인되었다. 따라서 15번째 라운드에서 소프트웨어 통합시험을 마무리하였다.

Table 6. Result of Estimated current failure rates

Test round	Models	Estimated Current Failure Rate (failure/hour)
11	Log-Logistic	0.488
12	Log-Logistic	0.5541
13	Weibull	0.0719
14	Inflection-S	0.0605
15	Weibull	0.007

또한, Weibull 모델을 기반으로 라운드별 추정 데이터와 실제 데이터를 비교한 것은 Fig. 6과 같다. 두 개의 데이터가 완벽히 일치하지 않지만, 추정 데이터의 분포와 결함의 분포가 유사한 것으로 보인다.

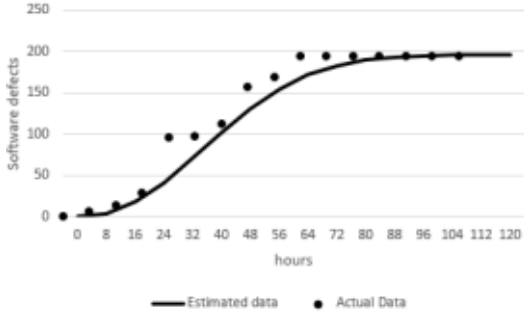

Fig. 6. the fitted data based on Weibull model with the actual data

5. 결론

본 연구에서는 소프트웨어 신뢰성 추정 모델을 무기체계 소프트웨어 연구개발에 적용하였다. 시범 적용 결과, 신뢰성 추정모델은 시험 종료 판단의 객관적 기준이 되었다. 기존 매뉴얼에 명시된 활동으로는 개발된 소프트웨어가 충분한 검증을 거쳤는지 판단하기 어려워 시험 종료 여부를 결정하는 것이 쉽지 않았다. 하지만 신뢰성 추정 모델을 적용함으로써, 시험 종료 기준을 명확하게 설정할 수 있었다. 이와 더불어 목표 고장률 달성을 위해 보다 완성도 있는 소프트웨어 통합 시험을 수행할 수 있었다.

무기체계 내 소프트웨어의 중요성이 증가함에 따라, 소프트웨어 신뢰성 확보가 주요 이슈로 대두되고 있다. 기존 매뉴얼에서 제시하고 있는 활동들만으로는 보다 객관적인 신뢰성 지표를 요구하고 있는 여러 이해관계자의

기대치를 충족하기 힘들 뿐만 아니라, 국제적인 흐름을 좇아가지 못하는 상황이다. 따라서 국내 무기체계 소프트웨어 개발 절차도 국제 표준에서 요구하는 활동들을 반영하는 노력이 필요하다고 판단된다. 본 연구에서는 신뢰성 추정 모델을 적용하는 것에 그쳤지만, 다음 연구에서는 신뢰성 예측 모델 적용, 운용 시점에서의 신뢰성 공학 관련 활동들을 수행할 예정이다. 그리고 그 결과를 기반으로 무기체계 소프트웨어 개발 과정에 적용할 수 있는 적절한 수준의 신뢰성공학 활동이 식별되기를 기대한다.

References

[1] Lyu, M.R., Handbook of Software Reliability Engineering, p.850, McGraw-Hill. 1996.

[2] IEEE, IEEE Recommended Practice on Software reliability, IEEE Reliability Society, IEEE std. 1633-2016, 2016.
DOI: https://doi.org/10.1109/IEEESTD.2017.7827907

[3] DAPA(Defense Acquisition Program Administration), "Weapon System Software Development and Management Manual", DAPA, 2018.

[4] GQ Kenny, "Estimating defects in commercial software during operational use", *IEEE Transactions on Reliability*, vol. 42, no. 1, pp. 107-115, 1993.
DOI: https://doi.org/10.1109/24.210280

[5] M. L. Shooman et al., "Reliability of Shuttle Mission Control Center Software", Proceedings of the Annual Reliability and Maintainability Symposium, pp. 125-135, 1983.

[6] Goel, B., and Okumoto, K., "Time-dependent error-detection rate for software reliability and other performance measures", *IEEE Transactions on Reliability*, vol. 28, no. 3, pp. 206-211, 1979.
DOI: https://dx.doi.org/10.1109/TR.1979.5220566

[7] Musa, J. D., B. Iannino, and K. Okumoto, Software Reliability: Measurement, Prediction, Application, p.621, New York: McGraw-Hill, 1987.

[8] Jelinski, Z., and Moranda, P., "Software Reliability Research", *Statistical Computer Performance Evaluation*, New York: Academic Press, pp. 465-484, 1972.
DOI:
https://dx.doi.org/10.1016/B978-0-12-266950-7.50028-1

[9] Shooman, M. L., Reliability of Computer Systems and Networks, Fault Tolerance, Analysis, and Design, p.560, New York: McGraw-Hill, 2002.

[10] Duane, J. T., "Learning curve approach to reliability monitoring", *IEEE Transactions on Aerospace*, vol. 2, no. 2, pp. 563-566, April 1964.
DOI: https://dx.doi.org/10.1109/TA.1964.4319640

[11] Musa, J. D., and Okumoto, K., "A logarithmic Poisson execution time model for software reliability measurement", *Proceedings of the Seventh International Conference on Software Engineering*, Orlando, FL, USA, pp. 230-238, Mar. 1984.

[12] Gokhale, S., and K. Trivedi, "Log-logistic software reliability growth model", *Proceedings IEEE High-Assurance Systems Engineering Symposium*, Washington, DC, USA, pp. 34-41, 1998.
DOI: https://dx.doi.org/10.1109/HASE.1998.731593

[13] Moranda, P., "Event-altered rate models for general reliability analysis," *IEEE Transactions on Reliability*, vol. 28, no. 5, pp. 376-381, Dec. 1979.
DOI: https://dx.doi.org/10.1109/TR.1979.5220648

[14] Yamada, S., M. Ohba, and S. Osaki, "S-shaped reliability growth modeling for software error detection," *IEEE Transactions on Reliability*, vol. 32, no. 5, pp. 475-478, Dec. 1983.
DOI: https://dx.doi.org/10.1109/TR.1983.5221735

[15] Kichang Kim et al, "A Case Study on Application for Software Reliability Model to Improve Reliability of the Weapon Systm", *Journal of Korean Institute of information scientists and engineers*, vol.38, no.8, pp.405-418, 2011.

[16] Chin-Yu Huang et al., "An Assessment of Testing-Effort Dependent Software Reliability Growth Models", *IEEE Transactions on Reliability*, vol.56, no.2, pp.198-211, 2007.
DOI: https://doi.org/10.1109/TR.2007.895301

[17] J. D. Musa, Software Reliability Engineering: More Reliable Software, Faster Development and Testing ,p.632, Author-House, 2004.

[18] Omar H. Alhazmi et al., "Application of Vulnerability Discovery Models to Major Operating Systems", *IEEE Transactions on Reliability*, vol.57, no.1, pp.14-22, 2008.
DOI: https://doi.org/10.1109/TR.2008.916872

[19] Huang, ,C. Y. Lyu, M. R., "Estimation and Analysis of Some Generalized Multiple Change-Point Software Reliability Models", *IEEE Transactions on Reliability*, vol.60, no.2, pp.498-514, 2011.
DOI: https://doi.org/10.1109/TR.2011.2134350

[20] Ohba, Mitsuru. "Inflection S-shaped software reliability growth model.", *Stochastic models in reliability theory*. Springer, pp.144-162, 1984.
DOI: https://doi.org/10.1007/978-3-642-45587-2_10

[21] Gordon, Theodore Jay. "The delphi method." *Futures research methodology*, 2(3), pp.1-30,1994.

[22] Samjoon Park et al, "SIRIUS:Systematic Investigation for Reliability Improvement Upon Software", *IEEE 28th International Symposium on Software Reliability Engineering Workshops (ISSREW)*, toulouse, France, 2017.

박 다 운(Da-Un Bak) [정회원]

- 2015년 2월 : 한국과학기술원 경영공학과 (경영공학석사)
- 2015년 3월 ~ 현재 : 국방과학연구소 연구원

〈관심분야〉
소프트웨어 신뢰성, 소프트웨어 신뢰성 공학