철도신호제어용 소프트웨어의 신뢰도 모델링에 관한 연구

이재호*, 신덕호*, 장선봉**, 안병구**, 지인호**
* 한국철도기술연구원, ** 홍익대학교

A Reliability Modeling of Software for Railway Signalling Systems

Lee Jae-Ho*, Shin Ducko*, Jang Sun-Bong**, An Beong-Ku**, Jee In-Ho**
* Korea Railroad Research Institute, ** Hongik University

Abstract - 안전필수시스템인 철도신호제어시스템의 신뢰성은 하드웨어와 소프트웨어의 신뢰도에 의해서 결정된다. 하드웨어의 신뢰도는 상대적으로 많은 연구와 환경적 시험을 통하여 비교적 용이하게 예측하고 입증할 수 있으나, 소프트웨어의 신뢰도는 반복실험결과에 의해서 추정해야 하므로, 입력 값에 따라서 신뢰도 추정치가 종속된다. 소프트웨어의 입력과 출력의 조합은 거의 Combinatoric으로 되기 때문 모든 경우를 시험하기는 불가능하다. 따라서 단순화된 방법에 의해서 소프트웨어의 신뢰도를 구하는 것이 중요한 문제로 부각되고 있다.

본 연구에서는 소프트웨어의 신뢰도를 예측하는 신뢰도 예측방정식(Reliability Prediction Equation)을 도출하여 신호제어시스템 소프트웨어에 대한 신뢰도 모델링을 수행하고자 한다.

1. 서 론

철도신호분야에서 전자, 통신기술의 발달과 더불어 소프트웨어를 이용한 시스템제어가 급증하고 있다. 신호시스템의 경우 과거에서는 기계식과 전기식 등의 순수 하드웨어 종속적인 방식을 사용하여 고장이나 사고가 발생에 대한 원인을 명확히 발견 혹은 규명할 수 가 있었다. 그러나 컴퓨터 제어를 이용한 시스템은 상당한 유연성(flexibility)을 갖지만 고장이나 사고가 발생할 경우에 원인규명이 용이하지 않고 대규모 인명이나 재산의 손실을 초래하는 철도신호분야에서는 안전필수(safety-critical) 소프트웨어를 주로 사용하고 있다.

안전필수 소프트웨어를 포함한 전체제어시스템의 신뢰도 산정에는 많은 어려움이 있다. 컴퓨터를 이용한 제어시스템의 신뢰성은 하드웨어와 소프트웨어의 신뢰도에 의해서 결정된다. 하드웨어의 신뢰도는 상대적으로 많은 연구와 환경적 시험을 통하여 신뢰도를 구할 수 있으나 소프트웨어의 신뢰도는 반복실험결과에 의해서 얻어질 수 있지만, 시스템 신뢰도를 사용개시 이전에 예측하고 입증하기 위해서는 소프트웨어의 신뢰도를 정량적으로 구하는 것이 중요한 문제로 부각되고 있다.

따라서 본 연구에서는 철도신호제어시스템에 사용되는 소프트웨어의 신뢰도를 예측하는 신뢰도 예측방정식(Reliability Prediction Equation)을 도출하여 신호제어시스템의 신뢰도를 예측하기 위한 신뢰성 모델링을 연구하여, 철도신호제어용 소프트웨어에 대한 신뢰도를 추정하기 위한 방안을 제시하였다.

2.1 및 2.2에서는 소프트웨어 신뢰도의 일반사항을 기술하고, 2.3 및 2.4에서는 소프트웨어 신뢰성 모델링 형태에 대하여 분류하여 각각의 상관관계를 분석하고 소프트웨어 신뢰성 모델을 선정하였다. 또한, 2.5에서는 선정된 모델과 관측된 데이터를 이용하여 시뮬레이션을 통해 관측된 데이터에 대한 소프트웨어 신뢰도를 산정하였다. 마지막으로 3장에서는 결론과 향후 추진사항에 대해 기술한다.

2. 본 론

2.1 개요

소프트웨어 신뢰성은 소프트웨어의 품질을 결정하는 가장 중요한 척도중의 하나이다. 소프트웨어가 광범위하게 사용됨에 따라 소프트웨어의 결함은 시간적, 물질적 손실은 물론 인명의 손실과 대규모 재산피해와 같은 치명적 결과를 가져올 수 있는 기능을 수행하게 되었다. 이러한 환경적 변화에 대응하기 위해 소프트웨어의 결함을 제거하여 고품질의 소프트웨어를 위한 연구가 진행되고 있다.

정량적인 소프트웨어 품질 측정의 한 부류로서 소프트웨어 복잡도(Complexity) 측정이 있다. 소프트웨어 복잡도 측정의 주된 관심은 프로그램의 크기, 제어흐름(Control Flow), 자료 흐름(Data Flow) 등을 바탕으로 한 프로그램의 정적 구조 분석으로 소프트웨어의 사용자 관점을 충분히 반영하지 못하는 단점이 있다. 이에 비하여 소프트웨어 신뢰성 모델(Software Reliability Model)을 이용한 소프트웨어 품질 측정은 소프트웨어 개발, 시험 과정에서 발생하는 소프트웨어 고장(Software Failure)을 근거로 통계적 추정을 실시하므로 복잡도 측정에 의한 품질 측정보다 사용자의 관점을 잘 반영한다. 또한, 개발자의 관점에 있어서도 소프트웨어 신뢰도 모델을 이용한 품질 측정은 동적인 요인에 기반한 품질 측정치를 제공한다. 그러므로 이러한 특성은 소프트웨어 개발 과정에서 소프트웨어의 상태를 파악하고, 품질 목표치를 설정하며, 목표치에 도달하기 위한 계획을 수립하는데 도움을 준다. 따라서 소프트웨어 신뢰성은 소프트웨어의 질을 결정하는 가장 중요한 척도중의 하나이다.

대부분의 소프트웨어 신뢰도 모델들은 동작 프로파일에 기반을 두고 소프트웨어의 동작상태를 관찰해왔다[1,2]. 이런 모델들은 주로 소프트웨어 개발완료 이후 단계에서 소프트웨어가 신뢰도 요구조건을 만족하는지에 대한 여부를 결정하기 위해서 주로 적용 되었다. 하지만, 이러한 통계적 수단을 사용하는 모델들의 단점은 테스트 데이터의 불충분과 소프트웨어의 업그레이드에 대한 적용이 용이하지 않다는 것이다.

소프트웨어 개발단계에서는 주로 프로그램의 오류를 중심으로 프로그래밍과 테스팅(Testing)을 중심으로 품질을 관리한다. 그러나 소프트웨어의 규모가 커지고 복잡해짐에 따라 소프트웨어 신뢰도가 매우 중시되고 있다. 소프트웨어 신뢰도공학(SRE)은 소프트웨어 분야에서 활발한 연구가 진행되는 분야로 발전하였다. 본 논문에서 고려되어진 모델들은, 각 모델들에 대한 가정, 실행을 위한 데이터 요구사항, 형태와 결과의 예측들로 구성된다. 이러한, 모델선정을 통해 얻어지는 예측 값들은 MLE(Maximum Liklihood Estimation)에 근거하며, 본 논문에서는 실패데이터(Failure Data)의 분포형태를 지수(Exponential)구조로 가정한다.

소프트웨어 신뢰성 모델의 개발을 위해 Musa와 Okumoto[3]가 제안한 모델 분류체계를 바탕으로 하면, $M(t)$는 시간구간 t에서의 실패의 임의의 수이고 평균값 함수 $\mu(t)=E[(t)]$로 표현된다. 만약 이때에 $\lim_{t\to\infty}\mu(t)<\infty$라고 하면 유한실패모델(Finite Failure Model)이 된다. 모델의 선정에 앞서 각 모델간의 관계를 이용해야 하므로 먼저 Poisson형태의 중요한 특성에 대하여 고려하여야 한다.

$$\mu(t)=\alpha F_a(t) \qquad (1)$$
$$\lambda(t)=\mu'(t)=\alpha f_a(t)$$

여기서, $R(t)$:신뢰도함수, $\mu(t)$:평균값함수, $\lambda(t)$:실패강도함수, α:상수, $F_a(t)$:각각의 실패 α의 시간에 대한 실패누적분포함수, $f_a(t)$:각각의 실패 α의 시간에 대한 실패확률밀도함수

모델을 간략화하기 위해 소프트웨어의 데이터수집 시점에서 소프트웨어에 존재하는 실패의 수를 N, 그리고, 하나의 실패가 검출되어지면 그것은 즉시 제거되는 것으로 가정하면 다음과 같이 쓸 수 있다.

$$\lambda(t)=N\cdot f_a(t) \qquad (2)$$
$$\mu(t)=N\cdot F_a(t)$$

2.2 모델의 제한과 이슈

수집된 데이터에 대한 모델의 적용에 있어서 다음과 같이 주어진 분석형태에 대한 제한을 주의하여야 한다.

가. 주어진 모델에 대한 가정을 알고 있어야 한다. 만약 선택된 모델이 사건발생의 시간의 차이를 통한 추정을 전제로 하는 경우에, 수집된 데이터의 시간간격이 모두 같은 크기라고 가정되면, 이 모델은 주어진 데이터의 유형에 맞지 않으므로 이 모델을 사용하려고 시도해서는 안 된다.

나. 추정의 한계를 명확히 해야 한다. 만약 소프트웨어가 테스트 되거나 사용되는 환경이 데이터가 수집되어진 환경과 상당히 변화되었다면 예측된 결과의 신뢰수준이 저하된다.

2.3 소프트웨어 신뢰도 측정절차 및 모델링 분석

신뢰도 모델을 통한 신뢰도추정의 절차는 그림 1과 같이 먼저 고장자료를 수집하여 데이터 특성에 적합한 신뢰도 모델을 선택한다. 모델이 선택되면 모델에 필요한 파라메타를 추정하여 이를 모델에 대입하여 모델의 적합성을 확인한다. 다음으로는 실제고장자료로부터 얻어진 모델의 정확성을 평가하기 위하여 적합도 검정을 실시한다. 검정의 결과를 통해 모델의 유도추정치를 획득하고 마지막으로 유도추정치를 바탕으로 의사결정을 하는 단계로 수행된다.

2.3.1 신뢰도 모델링분석

가. Exponential Failure Time Class of Models

Musa와 Okumoto's 분류체계를 사용함으로써 이 그룹은 지수함수적인 형태를 가지는 실패강도함수의 형태로 모든 유한한 실패 모델들로 구성되어 있다. 이 클래스 내의 2항(Binomial) 형태는 실패 당 상수 고장률에 의해 특징지어진다[$z(t)=\phi(t)$]. 검출되어진 i번째 실패 전의 고장률함수는 남아있는 실패들의 함수이다[$N-(i-1)$]. 실패강도함수는 지수함수적인 형태를 가진다[$N\cdot\phi\cdot e^{-\phi t}$]. 이 클래스 내의 Possion형태는 실패당 상수고장률[$z(t)=\phi(t)$]과 실패시간으로[$f_x(x)=\phi\cdot e^{-\phi t}$] 특징지어진다.

- Jelinski-Moranda de-eutrophication model[4]
- Non-homogeneous Possion Process(NHPP) model
- Schneidewind's model
- Musa's basic execution time model

나. Weibull and Gamma Failure Time Class of Model

- S-shaped reliability growth model

다. Infinite Failure Category Model

- Geometric Model
- Musa-Okumoto logarithmic Poisson

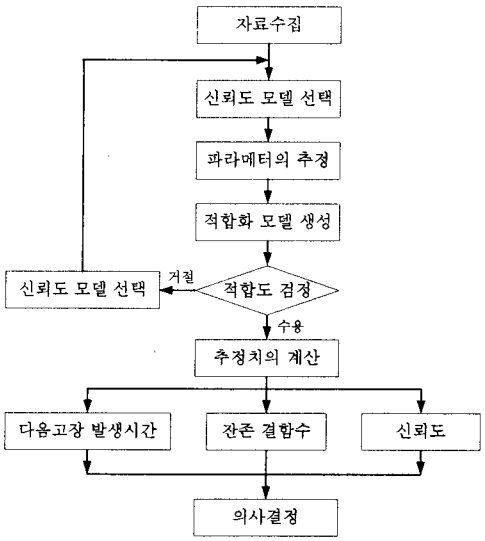

그림 1 소프트웨어 신뢰도를 측정하는 절차

2.3.3 모델들의 비교 상관관계 분석

가. Generalized exponential model class

일반화된 지수적 모델은 수집된 고장데이터가 지수함수의 특성을 가지는 경우에 다수의 모델을 하나의 수식집합으로 일반화시켜 모델처리를 용이하게 하기 위해 Shooman과 Musa에 의해 제안되었다.

표2 일반화된 지수함수모델의 관계

Model	Original Hazard rate function	Parameter equivalences
Generalized form	$z(t)=K[E_0-E_C(t)]$	
Shooman model	$z(t)=[E_0/I_T-\epsilon_{c(t)}]$	$\epsilon_c=E_c/I_T$, $I_T$is the number of instruction s $K=KI_T$
Jelinski-Moranda	$\phi(N-(i-1))$	$\phi=K$, $N=E_0$, $(i-1)=E_c(t)$
Basic execution model	$\beta_0\beta_1[1-\mu(t)/\beta_0]$ where $\mu(t)=\beta_0[1-e^{(-\beta_1 t)}]$	$\beta_0=E_0$, $\beta_1=K$, $\mu(t)=E_c(t)$
Logarithmic Poisson	$\beta_0\beta_1 e^{(-\mu(t)/\beta_0)}$ where $\mu(t)=\beta_0\ln(\beta_1 t+1)$	$\beta_0\beta_1=KE_0$, $E_0-E_c(t)=E_0e^{(-\mu(t)/\beta_0)}$

일반화된 지수함수모델의 기본이론은 실패의 발생비율은 남아있는 실패들의 수에 비례한다는 것이고, 고장률은 실패들 사이의 상수로 존재한다는 것이다. 따라서, 장애가 제거될 때 같은 수만큼 고장률도 감소하게 된다. 표준가정 외에 모델들을 위한 가정은 다음과 같다.

- 고장률은 소프트웨어의 현재 고장내용에 비례한다.
- 고장의 원인이 되는 결함은 즉시 수정되며, 결함을 수정함에 있어 새로운 결함을 발생시키지 않는다. 모델형태는 $z(t)=K[E_0-E_c(t)]$로 표현된다.

나. Exponential order static model class

본 모델은 소프트웨어 신뢰도 성장 프로세스의 실패시간들이 독립적인 총계와 non-homogeneous로 분포화된 지수적 랜덤 변수의 형태에 적합하다. Miller가 본 모델을 개발하였고, Jelinski-Moranda, Goel's NHPP, Logarithmic Poisson등이 모델의 적합성을 입증하였다.

모델간의 상관관계를 통해 모든 상황에 적합한 소프트웨어 신뢰성 모델은 존재하지 않으며, 소프트웨어를 사용하는 분야의 특수성에 따라 모델을 선택해야만, 실제 소프트웨어의 신뢰도에 가까운 추정치를 얻을 수 있음을 알 수 있다. 따라서 적합한 소프트웨어 신뢰도 모델의 선택을 위해서 모델의 특성 및 모델간의 연관관계를 분석하였다.

2.4 소프트웨어 신뢰도 모델

소프트웨어 신뢰성 모델의 분석이 올바르게 수행되고, 적용분야에 대한 고장특성을 고려한 모델간의 연관관계가 분석되면, 신뢰성 모델을 선택하는 방법의 문제가 큰 문제로 대두되게 된다.

본 논문에서는 신뢰도을 추정하기 위한 모델로 Schneidewind 모델을 선택하였다. Schneidewind 모델은 미국 우주왕복선의 기내시스템 소프트웨어의 신뢰도를 추정하기 위해 사용되었던 모델이다.

Schneidewind 모델은 non-homogeneous Poisson process(NHPP)를 따르는 모델 중 한가지이다. NHPP 에서 $m(t)$와 $\lambda(t)$의 관계는 다음과 같다.

$$m(t) = \int_0^t \lambda(s)ds \tag{3}$$

여기서, $m(t)$는 $N(t)$의 기대치인 MVF(mean value function), $\lambda(t)$는 intensity function 이다,

NHPP를 사용하는 Schneidewind 모델의 경우 failure intensity function $\lambda(t)$와 mean value function $m(t)$는 다음과 같이 정의된다.

$$\lambda(t) = \alpha e^{-\beta t} \tag{4}$$
$$m(t) = (\alpha/\beta)[1-e^{-\beta t}]$$

Schneidewind 모델을 적용하기 전 먼저 두개의 파라메터 α와 β을 추정해야 한다. 여기서 α는 $t=0$에서의 초기 실패율이고, β는 상수이다. 이 두 파라메터가 정해지게 되면 4개의 테스트 자원을 얻을 수 있다.

① $[0,t]$시간동안 나타나는 실패의 수 예측
$F(t) = (\alpha/\beta)[1-e^{-\beta t}]$

② 위의 특징을 사용해 $[t_1,t_2]$시간 구간내에서의 실패의 수를 예측
$F(t_1,t_2) = (\alpha/\beta)[1-e^{-\beta t}] - X_{0,t_1}$
X_{0,t_1} : $[0,t_1]$에서의 실패의 수

③ 소프트웨어의 life($t=\infty$)동안 발생할 최대 실패의 수 예측
$F(\infty) = \alpha/\beta$

④ ①,②,③을 이용하여 t에서 시스템에 남아 있을 수 있는 최대 실패의 수 예측
$R(t) = (\alpha/\beta) - X_{0,t_1}$

Schneidewind 모델은 다시 3개의 모델로 나뉘어 진다.

• 모델 1

전체 시간 구간을 n이라고 했을 때, 시간 구간을 동일한 구간으로 나누게 되고 동일한 가중치를 사용하게 된다. 첫 번째 모델을 사용한 경우 아래 식을 이용하여 α, β을 추정하게 된다.

$$\frac{1}{e^{\hat{\beta}}-1} - \frac{n}{e^{\hat{\beta}n}-1} = \sum_{k=0}^{n-1} k\frac{f_{k+1}}{F_n}$$

$$\hat{\alpha} = \frac{\hat{\beta}F_n}{1-e^{-\hat{\beta}n}},\quad F_n = \sum_{i=1}^{n} f_i$$

f_i : 실제 나타나는 실패의 수

• 모델 2

전체 시간구간을 n이라고 했을 때$[1,s-1]$까지 구간은 무시하고, $[s,n]$까지 구간만을 이용해 α와 β을 추정하게 된다. 이 때 α와 β을 추정하기 위한 수식을 다음과 같다.

$$\frac{1}{e^{\hat{\beta}}-1} - \frac{n-s+1}{e^{\hat{\beta}(n-s+1)}-1} = \sum_{k=0}^{n-s} k\frac{f_{k+s}}{F_{s,n}}$$

$$\hat{\alpha} = \frac{\hat{\beta}F_{s,n}}{1-e^{(-\hat{\beta}(n-s+1))}},\quad F_{s,n} = \sum_{i=s}^{n} f$$

• 모델 3

모델3의 경우 모델1과 모델2의 복합 형태로 $[1,s-1]$까지 구간은 누적시켜 첫 번째 나타나는 실패의 데이터처럼 사용하게 되고, $[s,n]$구간은 각각 독립적인 실패의 수로 사용되게 된다. 이 때 α와 β을 추정하기 위한 수식을 다음과 같다.

$$\frac{(s-1)F_{s-1}}{e^{\hat{\beta}(s-1)}-1} + \frac{F_{s,n}}{e^{\hat{\beta}}-1} - \frac{nF_n}{e^{\hat{\beta}n}-1} = \sum_{k=0}^{n-s}(s+k-1)f_{s+k}$$

$$\hat{\alpha} = \frac{\hat{\beta}F_n}{1-e^{-\hat{\beta}n}}$$

2.5 시뮬레이션 및 결과 고찰

2.5.1 입력데이터

자동열차제어 차상장치의 소프트웨어의 신뢰도 분석을 위해 사용된 데이터는 장치의 시운전 기간에서 소프트웨어의 실행 중 발생한 결함의 수를 체크 하여 24시간 단위로 누적된 결함의 수를 입력 데이터로 사용하였다. 2004년 11월 1일부터 2005년 10월 19일까지 총 116회 동안 장치를 가동하여 소프트웨어를 실행하였으며 이 중 4회의 결함의 발견되었다.

2.5.2 모델의 신뢰성테스트

NHPP를 사용하는 Schneidewind 모델 식(4)와 입력 데이터를 근거로 소프트웨어의 신뢰성 검사에 Schneidewind 모델3을 사용하였다. 모델2의 경우 추정시간을 줄여줄 수 있고 s값을 추정하는 명확한 근거가 제시되어 있는 장점이 있어 모델2를 사용하여 에러를 추정하고자 하였으나, 모델2의 경우 $[1,s-1]$구간 동안에 발생하는 에러를 무시하게 되는데 실험데이터의 특성이 처음 1-5회 사이에 에러 데이터의 대부분이 존재하기 때문에 모델2를 사용하게 되면 에러 데이터의 예측에 영향을 미치게 되는 대부분의 에러 데이터를 무시한 후 각 파라메터를 추정하게 되기 때문에 예측에서 많은 오차 성분을 가지게 된다. 따라서 지금과 같은 실험 데이터의 경우에는 모델 2를 사용할 수 없다. 모델2를 사용하려면 s값이 1과 가까운 값에서 최적을 보여야 하지만 s가 1에 가까워지면 모델 1과 동일한 결과를 나타내게 된다. 따라서 현재와 같은 실험데이터의 경우 모델 1과 모델 3를 선택해야 하는데 모델 3의 경우가 보다 정확한 예측을 수행하게 된다.

표1 모델1, 모델2(s=1), 모델3(s=2)

	$\hat{\alpha}$	$\hat{\beta}$	시스템내에 잔존 가능한 에러 수(예측)
모델1	0.6974	0.1744	6.584e-009
모델2	0.6974	0.1744	6.584e-009
모델3	0.6974	0.1744	6.584e-009

Schneidewind 모델 2의 경우 s=1이면 모델 1과 동일

한 모델이 된다. 모델 3에서의 최적의 s값은 모델 2의 s 값보다 1 큰 값이 일반적이다. 따라서 모델 1, s=1인 경우의 모델 2와 s=2인 경우의 모델 3이 같게 된다.

2.5.3 신뢰도 예측 시뮬레이션

입력데이터에 대한 평균, 분산, 표준편차 값을 다음과 같다.

표2 ATC 차상장치의 입력 데이터

입력개수	평균(mean)	분산(var)	표준편차(std)
116	0.0345	0.0510	0.2258

이렇게 얻어진 얻은 실측데이터를 근거로 하여 Maximum Likelihood Estimation 방법을 이용하여 파라메터 $\hat{\alpha}$, $\hat{\beta}$를 추정하게 되면 다음과 같은 추정치를 얻게 된다.

$$\hat{\alpha}= 0.5065,\ \hat{\beta}= 0.1266,\ s = 15 \qquad (5)$$

얻어진 파라메터를 Schneidewind 모델 3에 적용하게 되면 다음과 같은 실험 결과를 얻게 된다. $F(117 \sim \infty)$에서의 결함은 예측 모델을 이용해 추정한 결과이다.

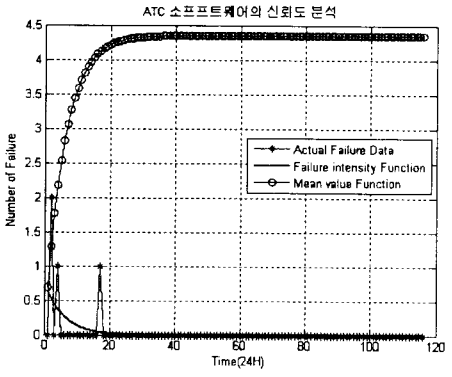

그림 2 ATC 차상 장치의 소프트웨어 신뢰도 분석을 위한 $\lambda(t)$, $\mu(t)$

신뢰도함수 $R(t) = 1 - F(t)$이므로 앞으로 예측될 $F(t \sim \infty)$구간에서의 신뢰도는 $1 - F(t, \infty)$가 된다. 이 수식에 의해 예측되어진 $F(t, \infty)$의 수를 이용하여 신뢰도를 산출하면 표4와 같이 신뢰도는 99.99985%로 예측되었다..

표3 모델3을 이용한 신뢰도 예측

	F(1, 116)	F(117, ∞)
장치가동 시 얻은 결함 수	4	·
Schneidewind 모델 3	4.000006989	1.4705e-006
신뢰도	99.99985%	

3. 결 론

안전필수시스템인 철도신호제어시스템의 신뢰성은 지금까지는 주로 하드웨어에 집중하여 연구가 수행되어왔으나 소프트웨어의 접목이 증가함에 따라 소프트웨어의 신뢰도에 대한 연구가 요구되었다.

따라서 본 연구에서는 철도신호제어시스템 접목되는 소프트웨어의 신뢰성을 추정하기 위한 기초적인 연구로 기존 소프트웨어 신뢰도 모델들을 형태에 따라 분류하고 각 분류에 속해있는 모델들의 소프트웨어 신뢰도 예측방정식을 도출하고 이들 모델들을 비교분석함으로써 신호제어용 소프트웨어에 적용 가능한 소프트웨어 신뢰도 예측방정식을 설정하고 자동열차제어 차상장치의 시운전 데이터를 적용하여 소프트웨어의 신뢰성을 예측을 수행하였다.

향후에는 이러한 방식의 타당성 검정과 효율화 방안에 대한 연구를 지속하여 철도신호제어시스템의 신뢰성예측에 적용될 수 있도록 하여야 할 것이다.

[참 고 문 헌]

[1] Cheung R. C., "A User-Oriented Software Reliability Model," IEEE Transactions On Software Engineering, 6(2), pp.118-125, March 1980.

[2] Farr W., "Software Reliability Modeling Survey," In M.R. Lyu editor, Handbook of Software Reliability Engineering, McGraw-Hill Publishing Company and IEEE Computer Society Press, New York, pp. 71-117, 1996.

[3] Musa, J.D.,Okumoto,K. "A Logarithmic Poisson Execution Time Model for Software Reliability Measurement," Proceeding s Seventh International Conference on Software Engineering, Orlando, Florida, pp. 230-238.

[4] Moranda, P.L., and Jelinski, Z., Final Report on Software Reliability Study, McDonnel Douglass Astronautic Company, MADC Report Number 63921, 1972.

[5] Goel,A.L., and Okumoto, K, "Time-Dependent Error-Detection Rate Model for Software and Other Performance Measures," IEEE Transaction on Reliability, vol.R-28,no.3, August 1979, pp.206-211.

[6] Schneidewind, N.F., "Analysis of Error Processes in Computer Software." Sigplan Note, vol.10, no.6, 1975, pp.337-346.

[7] Yamada,S., Ohba,M., and Osaki, S., "S-Shaped Reliability Growth Modeling for Software Error Detection," IEEE Transactions on Reliability,vol.R-32,no.5, December 1983, pp.475-478.

[8] Moranda, P.B., "Predictions of Software Reliability During Debugging," Proceedings of the Annual Reliability and Maintainability Symposium, Washington D.C., 1975, pp.327-332.