비-동질 안정 프로세스 기반 임베디드 시스템 소프트웨어의 신뢰성 특성에 관한 연구

(A study on the Reliability System Software based on NHPP(Non-Homogeneous Poisson Process)

한상섭* 백영구** 이근석*** 전현덕*** 류호중*** 이기서****
Han, Sang-Seop Baek, Young-Gu Lee, Keun -Seog Jeon, Hyun-Duck Ryu, Ho-Jung Lee Key-Seo

ABSTRACT

In this paper, we apply NHPP model example to s/w process in order to get to know s/w reliability. The test is constructed by a test zig of commercial product loaded real embedded sytstem s/w. It is established to s/w reliability prediction and estimation of real-time embedded system s/w. It is computed the prediction value of cumulative failures, the failure intensity, the reliability and the estimation value of MTTF, Failure Rate.

To the more realization of high reliability in the real-time embedded system s/w, if the embedded system s/w is ensured to the test coverage and constructed to stable s/w process & operating system, we can improve the performance and the reliability characteristic of the real-time embedded system s/w.

1. 서론

초기의 소프트웨어 신뢰성은 1970년대 후반 John Musa(AT&T Bell LAB) Okumoto, Shooman 등에 의해 그 기본 모델링이 정의됐으며 신뢰성 함수는 결함률(Failure Rate), 고장 평균 시간(MTTF : Mean Time To Failure)등을 도출하기 위해 하드웨어에 의존한 접근이 대부분이었다. 이후 시스템의 복잡성(Complexity)이 증대되면서 하드웨어적인 일부 측면을 소프트웨어가 처리함으로서 하드웨어 프로세서 운영체제의 급변화로 인해 기존의 소프트웨어 신뢰성 측정 및 모델링 매개변수들이 더 이상 적용이 불가능하게 되어, 80년대에 이르러 소프트웨어의 신뢰성은 큰 전환기를 맞게 된다. 이전까지 양적(Quantitative)측정을 통해 소프트웨어 신뢰성을 모델링 해오던 방식이 80년대 후반부터 90년대에 들어서면서 보다 다양화되고 복잡성을 지니면서 보다 Human-dependent하다는 특성으로 인해 등장하게 된 소프트웨어 프로세서를 통해 소프트웨어 신뢰성 매개변수를 도출하게 된다.

이후 소프트웨어 신뢰성은 이제는 소프트웨어 프로세서만으로는 비가시적이고 잠재적인 오류(Error, Fault, Failure)들을 검출할 수 없음을 인식하여 소프트웨어 테스팅 기술의 발전으로의 윤곽을 드러내기 시작했다.

오늘날의 소프트웨어 신뢰성은 소프트웨어 프로세스를 기반으로 구성 소프트웨어 시스템에 대한 정형화(Formalized)를 구현하고 해당 소프트웨어 시스템의 테스팅 시간동안 결함률(Failure Rate) 및 결함 강도(Failure Intensity) 그리고 결함 노출 비율(Fault Exposure Rate)등의 매개변수를 수집하여 소프트웨어 신뢰성을 모델링 할 수 있게 되었다.

이를 바탕으로 본 논문에서는 실시간 기반 내장형 시스템 소프트웨어 신뢰성 구현을 위해 소프트웨어 프로세스 및 테스팅 데이터를 바탕으로 소프트웨어 신뢰성 모델을 모델링하고 평가하여, 소프트웨어가 하드웨어에 비해 평균 고장 간격 시간이 매우 짧고 신뢰성 수치 또한 운영체제

* 광운대학교 제어계측공학과 석사, 정회원
** 광운대학교 제어계측공학과 석사과정, 비회원
*** (주) 대우 엔지니어링, 비회원
**** 광운대학교 제어계측공학과 정교수, 정회원

종속적이며, 이보다 높은 실시간 기반 내장형 시스템 소프트웨어의 신뢰성을 구현하기 위해서는 안정된 소프트웨어 프로세스와 운영체제를 구축하고 보다 높은 테스팅 커버리지를 확보해야 한다는 점에 대해서 논하고자 한다.

2.1 소프트웨어 신뢰성의 정의

소프트웨어 신뢰성의 정의를 IEEE Std.에서는 다음과 같이 정의하고 있다. "소프트웨어가 주어진 조건과 시간동안 시스템 고장 발생을 일으키지 않을 확률" [Ensuring Software Reliability : Ann Marie Neufelder 1993 published by Marcel Dekker]. 1970년대 후반부터 소프트웨어 신뢰성 모델을 연구해온 Dr. John MUSA (AT&T Bell Lab.)는 다음과 같이 정의한다. " 주어진 소프트웨어 시스템이 주어진 시간 주기에서 에러 없이 작동할 확률" 또한 Dr. Martin Shooman (New York Polytechnical University)은, "주어진 시간과 환경에서 컴퓨터 프로그램의 고장에 대해 자유롭게 동작할 확률"로 정의하고 있다.

이는 하드웨어의 신뢰성의 정의와 유사하지만, 특히 주어진 시간과 환경에서 불규칙한 변수가 발생할 수 있으며, 운영환경과 데이터이동에 대한 영향 등을 고려하여 소프트웨어의 잠재적인 설계 상의 에러를 검출할 수 있어야 한다.

2.2 하드웨어 신뢰성과 소프트웨어 신뢰성의 비교

일반적인 하드웨어 신뢰성과 소프트웨어 신뢰성 곡선은 다음과 같다.

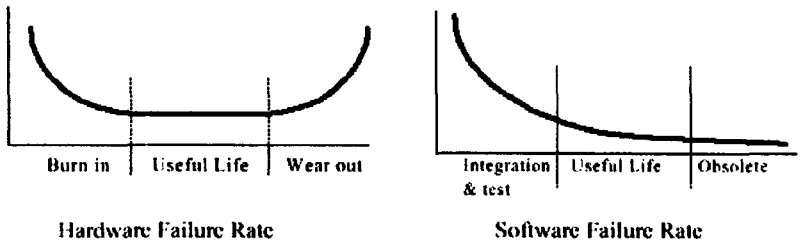


그림 [1]. 하드웨어와 소프트웨어 신뢰성 곡선 비교

다음 표는 하드웨어와 소프트웨어의 신뢰성영역에 대한 비교를 나타낸 것이다.

표 [1]. 하드웨어와 소프트웨어의 신뢰성 영역에서의 비교

Reliability		
항목	Hardware phase	Software phase
고장원인	설계, 제품 및 유지 보수	설계 결함
고장발생	고장은 시스템 또는 해당 제품이 소모(Wear)되거나 에너지와 관련된 현상에 의해 발생하며 잠재적인 증상이 현실적으로 나타난다	Wear-out 현상과는 관계가 없으며 고장 및 에러에 대한 어떠한 경고적인 현상 발생을 취합하기가 어렵다.
고장분석	고장 진단이나 분석을 위한 기구 및 형상을 재현할 수 있다	고장 분석을 위한 소프트웨어적인 방어가 어렵다.
시간관련	시간함수에 밀접하게 관련 운영시간이 늘어날수록 고장률이 떨어지는 것을 목표	반드시 시간에 의존하지는 않음 수행 논리 경로에 따라 고장 발생이 에러들을 수반해서 발생 신뢰성 성장은 에러들을 검출하거나 정정하는 활동을 통해 구현가능
환경영향	외부 환경적 조건에 많은 영향	외부 환경적 영향에는 상관없이 동작 내부 환경적 영향에 민감하게 반응 (충분한 메모리 할당+적절한 수행 속도등)

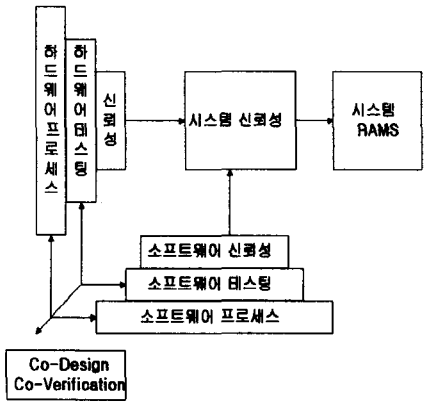
	Reliability	
항목	**Hardware phase**	**Software phase**
예견	물리적인 기반으로 부터 이론상 예견가능	환경 stress 요소, 설계의 지식으로부터 예견 불가
신뢰성향상	여분기법을 통해 향상	상이점(diversity)을 통해 향상
인터페이스	가시적	개념적
설계	표준 구성요소들을 사용	표준 구성요소의 사용은 아님

하드웨어와 소프트웨어 신뢰성에 관하여 John Musa의 말을 빌어 사용한다면 다음과 같다. "Software Reliability is determined by environments and conditional variables" 여기서 말하는 environments가 의미하는 것은 소프트웨어가 운영하는 환경을 뜻하며 이에 따라 개발 프로그램의 성능과 효율이 엄청난 차이를 가져올 수 있다는 것을 의미한다. 따라서 내장형 시스템 소프트웨어의 경우 신뢰도 함수 매개변수에 운영환경에 대한 매개변수를 꼭 적용해야 할 것이다. 이 운영환경에 대한 프로그램의 처리율이 해당 프로그램의 성능을 나타내며 이 성능매개변수는 해당 소프트웨어의 신뢰도 지표로서 사용 할 수 있게 된다.

소프트웨어의 신뢰성 성장모델(Software Reliability Growth Model)에 정확한 매개변수 표현을 통해서 어떻게 모델링 하느냐가 소프트웨어 신뢰성 구현의 열쇠라고 할 수 있다.

2.3 영역 및 역할

소프트웨어의 신뢰성은 다음과 같은 영역과 역할을 수행한다.



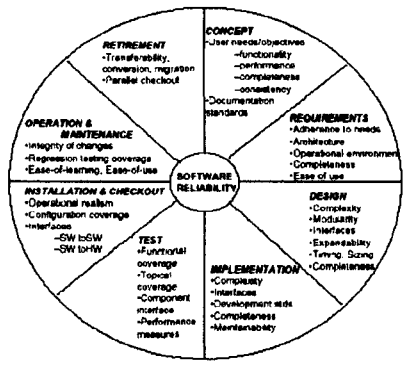


그림 [2]. 소프트웨어의 신뢰성 영역

이러한 영역과 비추어, 소프트웨어 신뢰성을 구현하기 위해서 갖추어야 할 선결조건에 대해서 IEEE/ANSI/AIAA 에서는 다음과 같이 밝히고 있다.

표 [2]. 세계 표준 소프트웨어 신뢰성 공학 구현 절차

소프트웨어 신뢰성 공학 프로세스 (Software Reliability Engineering Process)		
단계	IEEE/ANSI/AIAA	내 용
Step 1	Identify the Application	요구기능 정의, I/O 정의, Data 구조 및 흐름 정의
	Specify the Requirement	요구기능 명세화
	Allocate the Requirement	요구기능 할당
Step 2	Define Failure	발생 고장 정의
	Characterize the Operational Environment	운영 환경 특성
	Select Tests	테스팅 방법 선택
	Select models	신뢰성 측정 모델 선택
	Collect data	데이터 수집

소프트웨어 신뢰성 공학 프로세스 (Software Reliability Engineering Process)		
단계	IEEE/ANSI/AIAA	내 용
Step 3	Determine parameters	매개변수 결정
Step 4	Validate model	모델 검증
	Perform the Appropriate Analysis	분석(Prediction and Estimation)
Step 5	Safety	Software Safety Assurance

본 논문에서는 표준 SREP의 Step1,2에 대한 자료를 바탕으로 Step3와 4를 연구하였으며, 이러한 영역에서의 활동을 통해 얻어진 데이터를 기준으로 일반적인 소프트웨어의 신뢰성 모델을 분석하고 실시간 기반 임베디드 시스템 소프트웨어의 신뢰성 성장모델(SRGM : Software Reliability Growth Model)에 대한 예견과 평가 모델을 수립하고자 한다.

2.4 소프트웨어 프로세스 모델의 분류

소프트웨어 신뢰성모델은 초기 연구된 소프트웨어 또는 시스템 운영시간과 관련된 Finite 양적(Quantitative) 모델(Musa & Lyu 연구), 최근 연구되고 있는 시간과 관련되지 않는 Infinite 양적 모델, 소프트웨어를 통한 Qualitative 평가 모델로서 크게 3가지 형태로 분류 할 수 있으며,

표 [3]. 소프트웨어 신뢰성 모델 분류

소프트웨어 프로세스 모델의 분류	소프트웨어 특성에 따른 분류	실시간 기반 임베디드 시스템 소프트웨어	시스템 운영시간과 관련된 양적모델
			시스템 운영시간과 관련이 없는 양적모델
		응용프로그램 및 기타 Visualized program	소프트웨어 프로세스를 통한 질적 모델
	임베디드 시스템 소프트웨어 신뢰성 모델의 분류	Time domain	Wall Clock
			Execution Time
		고장 주기에 따른 분류	Finite
			Infinite
		프로세스형태에 따른 분류	Poisson
			Binomial
		Poisson	NHPP
			HPP

본 논문에서는 소프트웨어 신뢰성을 기초로 하여 시간 영역(Time Domain)에 따라 Prediction과 Estimation[1]으로 모델을 분류, 적용하고 NHPP-기반 프로세스를 기준으로 제한된(Finite) 상태와 무제한(Infinite) 상태를 구분하여 모델을 적용하고자 한다.

2.5 소프트웨어 신뢰성 기초 해석

일반적인 소프트웨어 신뢰성 모델 구현을 위한 매개변수의 확률적 표현은 식(1)-소프트웨어 결함 함수-과 같다.

$$F(t) = P[T \le t] = \int_0^t f(x)dx \qquad \text{- 식 (1)}$$

T : 물리적인 시스템의 고장 시간이나 유효기간

t : 소프트웨어가 요구하는 고장 시간

1) Prediction 과 Estimation : 소프트웨어 신뢰성 공학에서 Prediction은 개발 주기(Development Cycle) 내에서 얻어지는 해당 시스템 소프트웨어의 신뢰성 예견 함수 값이며 Estimation은 개발된 시스템 소프트웨어를 생명주기(Life-Cycle) 내에서 요구되어지는 시스템 운영 주기(System Operation Cycle)에서 수집된 소프트웨어 신뢰성 통계 데이터를 말한다.

결함함수 F(t)에 대한 시스템 존속 확률은 식(2)-해당 소프트웨어의 신뢰성 함수-가 된다.

$$R(t)=P[T>t]=1-F(t)=1-\int_0^t f(x)dx=\int_t^\infty f(x)dx \qquad \text{- 식 (2)}$$

소프트웨어 신뢰성은 하드웨어와는 달리 시스템 실행주기의 모든 시간 간격에 대한 신뢰성을 표현하기에는 운영환경의 다양성과 실행 상태에 대한 비가시적인 어려움으로 인해 특정 고장 간격에 대한 신뢰성 함수만을 확률로 표현 할 수 있다. 주어진 시간 간격 [t_1, t_2]에서 고장이 발생할 확률을 고장률(Failure Rate)이라고 한다. 단, 시간 t_1이전에는 고장이 발생하지 않는 것을 전제로 한다.

$$\frac{P[t_1\le t\le t_2|T>t_1]}{t_2-t_1}=\frac{P[t_1\le t\le t_2]}{(t_2-t_1)P[T>t_1]}=\frac{F(t_2)-F(t_1)}{(t_2-t_1)R(t_1)} \qquad \text{- 식 (3)}$$

또한, 시간 간격이 0으로 근접할 때 위험률(Hazard Rate)은 식(4)-위험률 함수-와 같다.

$$Z(t)=\lim_{\Delta t\to 0}\frac{F(t+\Delta t)-F(t)}{\Delta t R(t)}=\frac{f(t)}{R(t)} \qquad \text{- 식 (4)}$$

$Z(t)$는 주어진 시간 t 까지 시스템이 존속되어질 때 순간 고장률(Instantaneous Failure Rate)이다. 따라서, 고장 평균률을 λ 라고 했을 때 다음과 같은 소프트웨어 신뢰성 기초 측정 매개변수((식)5,6,7)를 나타낼 수 있다.

$$\mu(t)=\lambda\times t \qquad \text{- 식 (5)}$$

$$MTTF(\text{Mean Time To Failure})=\frac{1}{\lambda}\ \Leftrightarrow\ \lambda=\frac{1}{MTTF} \qquad \text{- 식 (6)}$$

$$R(t)=e^{-\mu(t)} \qquad \text{- 식 (7)}$$

실시간 기반 내장형 시스템 소프트웨어의 신뢰성 모델 수립을 위해 앞에서는 일반적인 신뢰성 매개변수에 대한 해석을 했다. 이러한 일반적인 신뢰성 모델 매개변수를 바탕으로 실시간 기반 내장형 소프트웨어에 대한 신뢰성 모델 매개변수를 해석 하고자 한다.

2.6 실시간 기반 임베디드 시스템 소프트웨어의 신뢰성 모델 접근

다양한 운영환경과 불규칙적이고 비가시적인 실행 형태를 갖는 특징을 통해 가정(1)에서는 시간 t에서 일어날 수 있는 불규칙한 고장수를 가정하고 정의(1)에서는 가정(1)에 대한 기대값을 통해 평균 고장 함수를 정의 할 수 있다.

고장들의 불규칙 수(Random Variable of Failures) = M(t) - 가정 (1)

평균 고장 함수(Mean Value Function) = $\mu(t)$ - 가정 (2)

$$\mu(t)=E[M(t)] \qquad \text{- 정의 (1)}$$

초기 소프트웨어 신뢰성 엔지니어들은 제한 시간 간격의 모델링에 접근하였고 점차적으로 무제한 시간 간격의 모델링에 많은 관심을 기울이게 되었다. 모델링 수립을 위한 측정 시간 주기에 대해 민감한 이유는 구현하려는 소프트웨어가 어느 정도의 시간주기를 갖고 신뢰 할 수 있느냐가 소프트웨어 신뢰성 모델링을 하는 가장 큰 이유이기 때문이다.

또한 실시간 기반 내장형 시스템의 소프트웨어 신뢰성 모델은 매개변수 형태가 매우 중요하기 때문에 Poisson과 binomial형태로 분류하여 모델링 접근하는 것도 매우 중요하다고 할 수 있다.

비-동질 안정 프로세스(Non-Homogeneous Poisson Process)는 1979년 Amrit Goel and Kazu Okumoto에 의해서 처음으로 제안되어졌으며 독립적인 Poisson 불규칙 변수를 시간 단위에 대한 결함들의 수로 표현한 것이다. 대부분의 실시간 기반 내장형 시스템 소프트웨어가 비 동질적이기

때문에 본 논문에서도 NHPP모델[2] 기반의 소프트웨어 신뢰성 성장 모델(Software Reliability Growth Model : SRGM)을 구현한다.

소프트웨어의 고장이나 결함 강도는 누적결함이나 검출되어지는 에러들의 변화율을 나타낸다. 소프트웨어 신뢰성 성장모델은 고장간격의 테스팅 시간동안 수집되어지는 자료를 사용하여 설계되어지며 대부분의 모델은 아래의 정의 (2),(3)를 가지고 모델링 할 수 있다. 가정 (1),(2)와 정의 (1),(2),(3)로부터 NHPP모델 기반 Exponential 과 Logarithmic 소프트웨어 신뢰성 모델을 해석하고자 한다.

Mean value function : μ_t - 정의 (2)

Failure intensity function : λ_t - 정의 (3)

2.6.1 The Exponential Model

대부분의 소프트웨어 신뢰성 성장 모델은 지수함수 모델로 나타나며, 비-동질 안전 프로세스 모델에 기반을 둔 통계적 모델로서, 1975년 MUSA는 CPU 실행 시간에 따른 소프트웨어 신뢰성 성장 모델에 대한 정확한 예측을 할 수 있는 모델을 선보였다. 일반적으로 지수함수 소프트웨어 신뢰성 모델은 다음과 같은 수식을 갖는다.

$$\mu_t = \beta_0[1 - e^{\beta_1 t}] : \text{Mean Value Function} \quad \text{- 식 (8)}$$

$$\lambda_t = \beta_0 \beta_1 e^{-\beta_1 t} : \text{Failure Intensity Function} \quad \text{- 식 (9)}$$

(β_0 = 초기 소프트웨어 전체 결함 수)

(β_1 = 프로그램의 결함 위험률)

2.6.2 The Logarithmic Model

1984년 Musa, Okumoto에 의해서 제안되어진 Logarithmic model은 exponential model과 유사하며 가장 큰 차이점은 고장 강도(failure intensity)가 지수적으로 줄어드는 것이다. 1992년 Malaiya, Karunanithi, and Verma 는 logarithmic model이 다른 어떤 SRGMs(Software Reliability Growth Models)보다 일반적이고 정확하다고 평가했다.

$$\mu_t = \beta_0 \ln(1 + \beta_1 t) \quad \text{- 식 (10)}$$

$$\lambda_t = \frac{\beta_0 \beta_1}{1 + \beta_1 t} \quad \text{- 식 (11)}$$

2.7 소프트웨어 신뢰성 측정 방법

소프트웨어 신뢰성 측정 방법에는 평가(Estimation)와 예견(Prediction)이 있다. 평가는 시스템개발 및 생명주기 전체에 걸쳐 수집되는 고장 데이터를 통계적 추론 기술을 응용하여 소프트웨어 신뢰성을 결정하는 활동을 말하며 과거로부터 현재 시점에 이르기까지 신뢰성 데이터 및 모델링 상태를 말하며, 예견은 소프트웨어 개발 단계에서 수집되는 고장 데이터의 사용가능성에 따라 사용 가능한 경우 시스템 테스팅이나 동작 단계에서 사용 불가능할 경우 설계나 코딩 단계에서 제품특성에 따른 매개변수들을 설정하여 소프트웨어 신뢰성 모델에 예견을 하게 된다. 따라서 개발단계 소프트웨어 신뢰성 예견과 시스템 생명주기 전체에 걸친 신뢰성 지표를 통해 비교 분석이 가능하다.

---

2) NHPP(Non-Homogeneous Poisson Process) Model : 1979년 Amrit Goel and Kazu Okumoto에 의해서 처음으로 제안되어졌으며 독립적인 Poisson 불규칙 변수를 시간 단위에 대한 결함들의 수로 표현한 것이다.

3. 실제 실험 및 결과 고찰

3.1 소프트웨어 프로세스의 적용

소프트웨어 프로세스(Software Process)를 구현하기 위해서는 많은 인프라(Infrastructure)가 필요하다. 대부분이 관리적인 측면에서 구현되며 현재 국내 소프트웨어 프로세스의 수준은 평균적으로 상업용은 CMM 및 SPICE 레벨 II 정도의 수준이고 기간 산업이나 기타 임계-안전(Safety-Critical) 시스템의 경우 레벨 III - IV 정도의 수준을 갖고 있다.

3.2 소프트웨어 테스팅(User Interface Testing)

소프트웨어 테스팅은 크게 두 부분으로 나눌 수 있다.

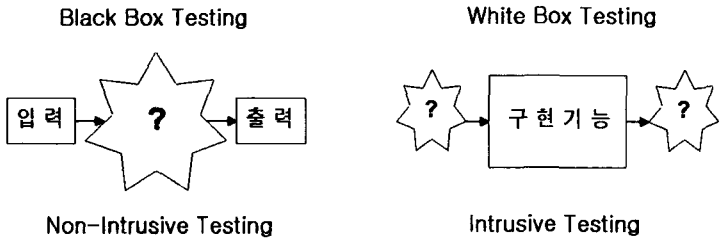


그림 [3]. 내장형 시스템 소프트웨어의 테스팅 방법

두 테스팅의 차이는 그림 [3]에서 보듯이 입출력과 기능 구현 부분에서 서로 정반대의 상태를 갖고 있으며 직접 하드웨어를 침입(Intrusive)하지 않고 논리적인 부분만을 테스팅 하는 Black Box Testing과 소프트웨어의 논리적인 부분과 함께 직접 하드웨어의 제어 메모리 부분을 검출하는 White Box Testing으로 분류할 수 있으며 본 논문에서는 무선 휴대 전화기의 Black Box Testing을 위한 User Interface Zig를 구성하여 쌍방간 Air On 상태에서의 Menu Screen 테스팅 및 기능 테스팅에 대한 Non-Intrusive 테스팅을 구현하였다.

그림 [4]. 무선 휴대폰의 Black Box Testing Zig(User Interface)

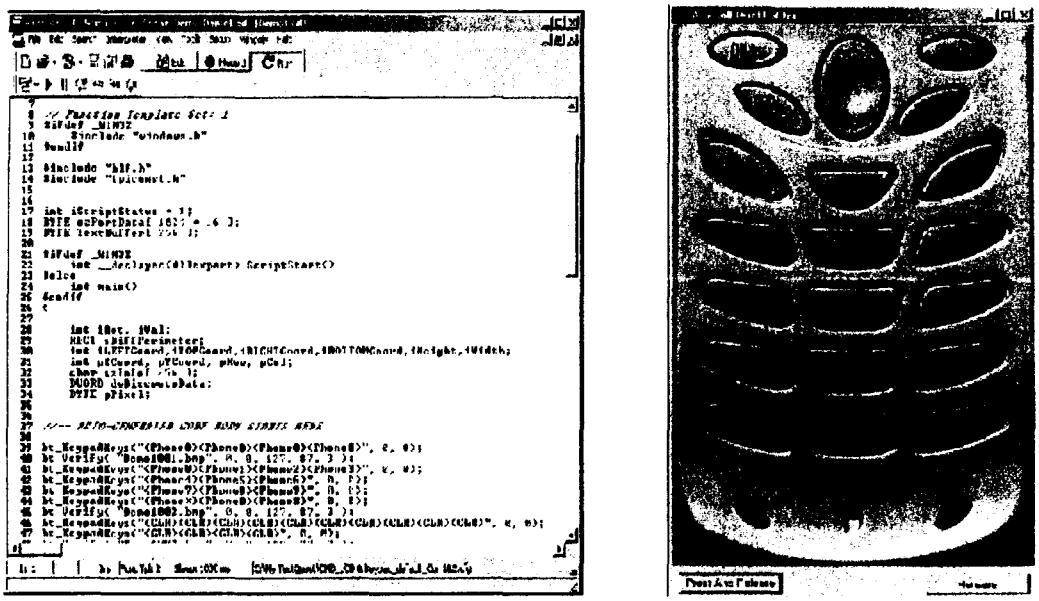


그림 [5]. 테스팅 프로그램 및 제어판

3.3 내장형 시스템 소프트웨어 신뢰성 연산

구현된 소프트웨어 신뢰성 프로그램은 표 [4]와 같은 데이터 구조로 되어 있으며 그림 [6]과 같이 NHPP 모델 데이터를 기반으로 3.2절의 테스팅 실례의 고장 주기, 결함률, Defect Density, 테스팅 주기 신뢰성 함수식으로 구현됐다. 그림 [6]은 요구되는 신뢰성 구간을 시간으로 표현한 것이며 그림 [7]은 경과되는 에러 수에 따라 신뢰성 함수식(Exponential & Logarithmic)으로 표현한 것이다. 단, severity는 고-안정, 고-난위도 소프트웨어가 아니기 때문에 1로 셋팅(Setting) 되었다.

표 [4]. 소프트웨어 신뢰성 Prediction 프로그램 데이터 형태

기 준	데이터 구조
고장기반형태	<고장수><고장 함수식><Severity 분류>
시간기반형태	<간격수><고장간격><임의시간간격><누적 결함수><Severity 분류>

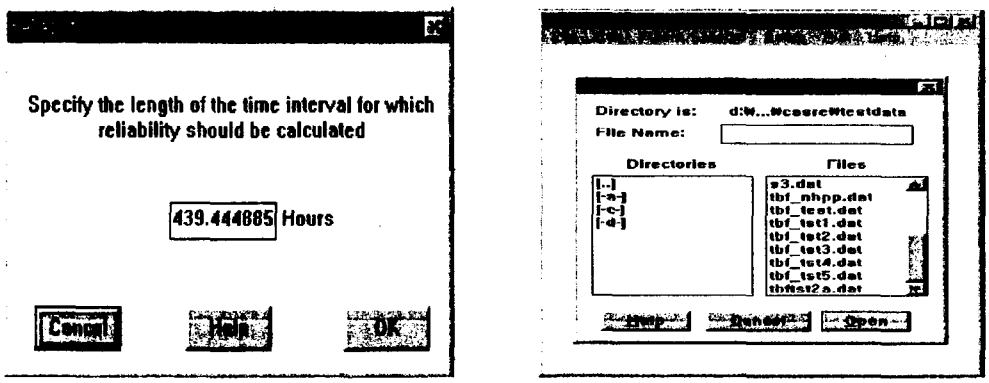
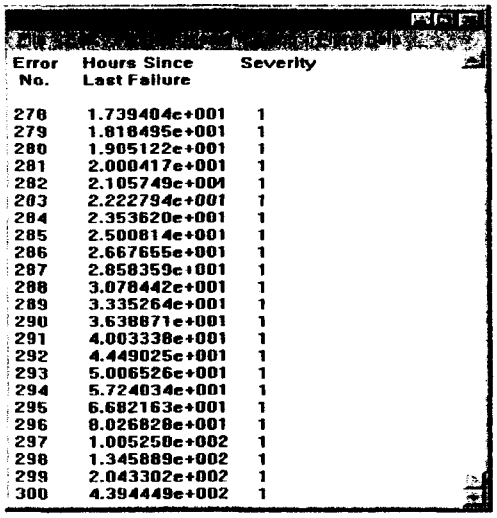


그림 [6]. NHPP 모델 소프트웨어 신뢰성 Prediction 초기화면



Error No.	Hours Since Last Failure	Severity
278	1.739404e+001	1
279	1.818495e+001	1
280	1.905122e+001	1
281	2.000417e+001	1
282	2.105749e+001	1
283	2.222794e+001	1
284	2.353620e+001	1
285	2.500814e+001	1
286	2.667655e+001	1
287	2.858359e+001	1
288	3.078442e+001	1
289	3.335264e+001	1
290	3.638871e+001	1
291	4.003338e+001	1
292	4.449025e+001	1
293	5.006526e+001	1
294	5.724034e+001	1
295	6.682163e+001	1
296	8.026828e+001	1
297	1.005250e+002	1
298	1.345889e+002	1
299	2.043302e+002	1
300	4.394449e+002	1

그림 [7]. 에러에 따른 고장 함수 식

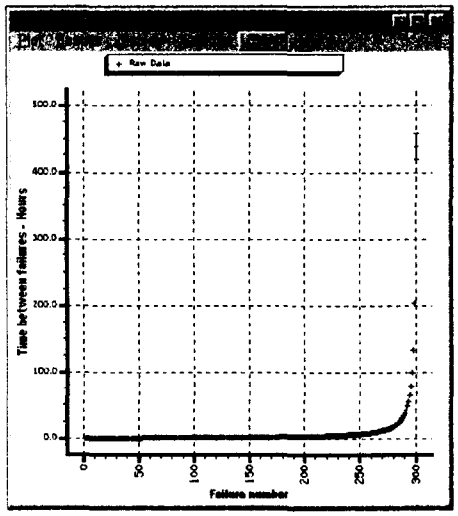


그림 [8]. Time Between Failure

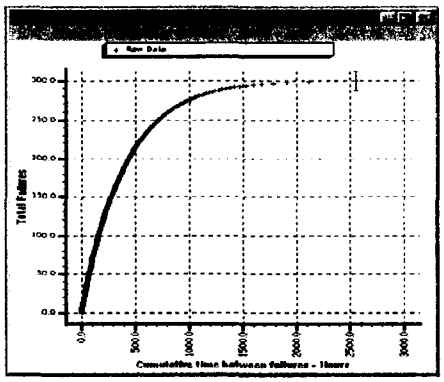


그림 [9]. Failure Intensity

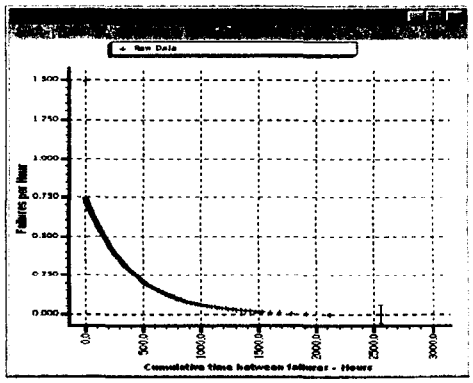


그림 [10]. Cumulative Failure

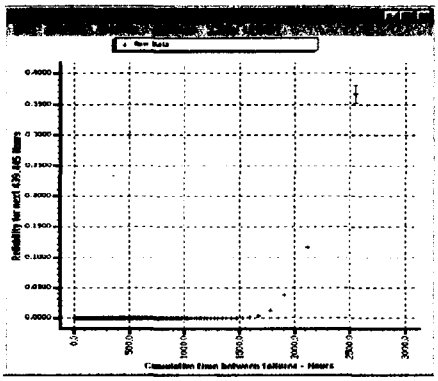


그림 [11]. Reliability

위 프로그램에서 수행한 소프트웨어 신뢰성 수치를 직접 Prediction하면 다음과 같이 할 수 있다. 아래 예제는 신호 시스템의 소프트웨어를 Ada로 프로그래밍하고 하드웨어 프로세스의 속도를 기준으로 소프트웨어 신뢰성 수치를 연산한 것이다.

예제) 20.000 라인의 Ada 프로그램을 CPU 실행 속도 2 MIPS로 실행하면 어느 정도의 신뢰성을 보장 할 수 있을까(단, t=40,000 sec)?

H_S = 2 MIPS = 2,000,000 명령어/초

$F_R = 4.2\times 10^{-7}$

$F_L = \frac{6}{1000}\times 20,000 = 120$faults

O_{ins} = 20,000 * 4.5 = 90,000 명령어

$\lambda_0 = \frac{2,000,000\times 4.2\times 10^{-7}\times 120}{90,000} = 0.00112$ failures/CPU Second

$\beta = B\frac{\lambda_0}{F_L} = 0.955 \times \frac{0.00112}{120} = 8.91\times 10^{-6}$failures/second

$\lambda(40,000) = 0.00112\, e^{-[(8.91\times 10^{-6})\times 40,000]} = 0.000$ure/CPU second

테스팅 기반 기능 구현 주기 내에서의 시스템 소프트웨어에 대한 신뢰성 예견 예제를 위와 같이 나타낼 수 있다. 위 Prediction 예제를 바탕으로 Estimation을 하면 단위 실행에 따른 예견을 전체 Estimation 하기 위해 코드의 길이가 20,000 KSLOC (5 Unit)에서 100,000 KSLOC가 됐으며 각각의 Estimation 매개변수 값은 표 [5]와 같다.

표 [5]. 소프트웨어 신뢰성 Estimation 매개변수 수치 실례

매개변수 명	매개변수 수치
K	0.0667
ET	29
EC(t)=ED(t)	23
λ_0	1.0
θ	0.9
τ	380

이 때 소프트웨어의 MTTF, $\lambda_{Exponential}$, $\lambda_{Logarithmic}$의 값은 다음과 같이 구할 수 있다.

$$\text{MTTF}(t) = \frac{1}{K[\frac{ETF - EC(t)}{I(t)}]} = \frac{1}{0.0667(29-23)} = 2.498751 \text{ days}$$

$$\lambda_{Exponential} = \lambda_0 \exp^{[\frac{-\lambda_0\tau}{ETV}]} = 1.0 \exp^{[\frac{-1.0\times 380}{29}]} = 0.000002 \text{ 고장/day}$$

$$\lambda_{Logarithmic} = \frac{\lambda_0}{\lambda_0\theta\tau+1} = \frac{1.0}{(1.0)\times(0.9)\times(380)+1} = 0.002916 \text{ 고장/day}$$

$R(t)_{Exponential} = 1 - \lambda_{Exponential} = 1 - 0.000002 = 0.999998$

$R(t)_{Logarithmic} = 1 - \lambda_{Logarithmic} = 1 - 0.002916 = 0.997084$

위 결과에서 알 수 있듯이 고장 강도와 실행 누적 시간 매개변수를 적용한 Logarithmic 모델이 Exponential 모델에 비해 실행 시간 기준의 내장형 시스템 소프트웨어에서는 좀더 정확한 신뢰성 데이터를 보장한다는 것을 알 수 있었다.

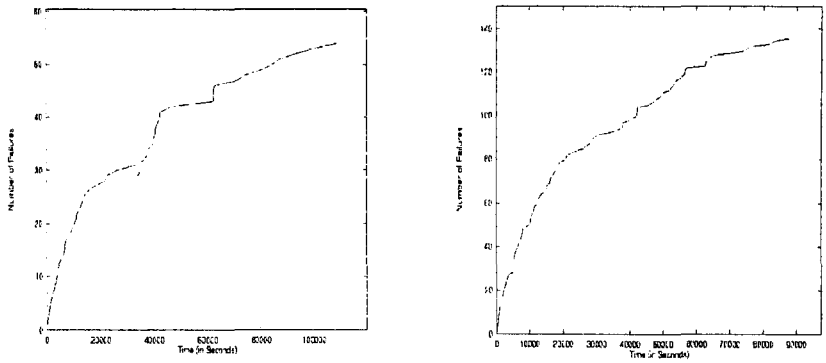


그림 [12]. 상용 시스템 소프트웨어와 기간 소프트웨어의 결함 수 비교

4. 결론

본 논문에서는 비-동질 안정 프로세스(Non-Homogeneous Poisson Process) 모델을 기반으로 사용자 인터페이스(User Interface) 테스팅을 위한 테스트 지그(Test Zig)를 구현했으며 CASRE(Computer Aided Software Reliability Engineering) Tool을 재구성하여 실제 실시간 기반 내장형 시스템 소프트웨어의 신뢰성 수치를 도출해 냈다. 이 신뢰성 수치 연산을 위해 함수식과 매개변수를 수립했으며 해당 소프트웨어의 예견 수치(Prediction value)와 평균 고장 시간(MTTF : Mean Time to Failure), Exponential, Logarithmic 형태의 고장률(Failure rate)과 신뢰성(Reliability)을 평가 수치(Estimation value)를 도출 할 수 있었다.

이 수치에서 알 수 있듯이 예견 신뢰성(Prediction Reliability)은 테스팅 신뢰성(Testing Reliability)이라고 할 수 있으며 평가 신뢰성(Estimation Reliability)은 운영 신뢰성(Operation Reliability)이라고 할 수 있다. Testing Reliability의 값은 ABCD 면적으로 나타낼 수 있고 Operation Reliability는 ABC'D 면적으로 나타낼 수 있었다. 이때 λ,은 소프트웨어가 양산(release)될 때의 고장 강도(Failure Intensity)를 나타낸다.

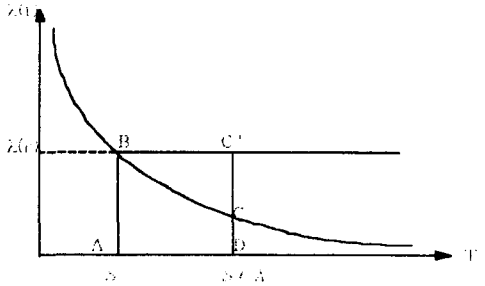


그림 [13]. Testing & Operation Reliability 곡선

실시간 기반 내장형 시스템 소프트웨어의 신뢰성은 평가 수치(Estimation value)에서 알 수 있듯이 소프트웨어는 하드웨어에 비해 평균 고장 간격 시간이 매우 짧고 신뢰성 수치 또한 운영체제 종속적이며 보다 높은 실시간 기반 내장형 시스템 소프트웨어의 신뢰성을 구현하기 위해서는 안정된 소프트웨어 프로세스와 운영체제를 구축하고 보다 높은 테스팅 커버리지를 확보해야 한다는 것을 알 수 있었다.

참고문헌

[1] "Software Safety and Reliability" written by Debra S. Hermann IEEE Computer Society ISBN : 0-7695-0299-7

[2] "Standard for Software Reliability Engineering" written by Dr. Norman F. Schneidewind of

Naval Postgraduate School Monterey, CA 93943.

[3] "Thesis : Accurate Software Reliability Estimation" submitted by Jason Allen Denton of Computer Science. Dept. Colorado State University 1999.

[4] "Introduction of Computer Software Reliability Estimation Tool(CASRE)" DACS of DoD. written by Raymond Wong SENG 609.11 Dept. Computer Science, Dept. Electrical and Computer Engineering University of Calgary Alberta, CANADA T2N 1N4 April 5th, 1999

[5] "Metrics for Evaluation of Software Reliability Growth Models" written by Francis C.L. Chan, Paul P. Dasiewicz and Rudolph E. Seviora Dept. Electrical and Computer Eng. University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

[6] "A Study of Operational and Testing reliability in Software reliability analysis" written by B. Yang, M. Xie edited by reliability eng. and system safety 70 (2000) 323-329 Dept. Industrial and system eng. National University of Singapore, Kent Ridge, Singapore 119 260. Received 9 August 1999: accepted 10 June 2000

[7] "Software Safety and Reliability" written by Debra S. Hermann IEEE Computer Society ISBN : 0-7695-0299-7

[8] "Software Test Coverage and Reliability" written by Yashwant K. Malaya, Naixin Li James M. Bieman, Rick Karcich, Bob Skibbe. Colorado State & Storage Tek 2270 South 88th Street Louisville, CO 80028-2286

[9] "HandBook of Software Reliability Engineering" Written by Michael, Lyu. Edited by McGraw Hill 1995. ISBN 0-07-039400-8, IEEE Computer Society Press.