論文2003-40CI-2-4

소프트웨어 최적 배포시기 결정 방법에 대한 고찰

(Study on The Optimal Software Release Time Methodology)

李 在 起 *, 朴 鍾 大 *, 南 相 植 *, 金 昌 奉 **

(Jaeki Lee, Jongdae Park, Sangsik Nam, and Changbong Kim)

요 약

소프트웨어 배포 문제는 프로젝트관리에 매우 중요하다. 왜냐하면 다양한 운용 환경 하에서 개발비용 및 에러의 발견, 수정 등에 밀접한 관계에 있기 때문이다. 본 논문은 대형 교환시스템 소프트웨어의 Release 시점을 예측할 수 있는 최적배포 문제로서 시스템의 안정도를 평가해 볼 수 있는 측면에서 소프트웨어 최적 배포 문제를 다루었다. 또, 신뢰도 평가 기준을 제시하여 제품의 적기 공급 및 개발자원의 효율적 이용 측면을 분석하고 신뢰성 평가 척도와 개발 비용 고려한 최적 배포 문제를 기술하였다. 그밖에 소프트웨어 신뢰도 성장 모델 중 지수형 모델을 근거로 한 초기 고장데이터를 활용하여 교환시스템의 소프트웨어 개발비용과 신뢰성 평가기준을 고려한 최적 배포시기를 결정하고 시험시 발생된 고장데이터에 대한 분석 및 관리 기법 등을 소개한다.

Abstract

An optimal software release, which is related to the development cost, error detection and correction under the various operation systems, is a critical factor for managing project. This paper described optimal software release issues to predict the release time of large switching system with the system stability point of view and evaluated a timely supply of target system, proper utilization of resources under the software reliability valuation basis. Finally, Using initial failure data, based on the exponential reliability growth model methodology, optimal release time, and analysis of failure data during the system testing and managing methodologies were presented.

Keywords : optimal release, MTBF : mean time between failure, software reliability, error, failure

1. 서 론

지식 정보화 사회가 발전함에 따라 사용자의 다양한 서비스 욕구와 함께 정보의 양이 폭발적으로 증가하게 되었다. 이와 같은 정보들을 실시간으로 처리하고 멀티 플랫폼 개발환경에서 개발된 프로그램들이 안정하게 동작하면서 고 신뢰성을 갖는 다양한 서비스를 제공할 수 있는 임베디드 시스템이 필요하게 되었다. 즉, 임베디드 시스템은 과거에 비해 복잡한 기능이 요구되고 있으며, 대형 소프트웨어를 개발해야 하는 추세에 있다. 다시 말해서 임베디드 시스템은 일반적인 시스템과는 달리 특정한 목적으로 작업하도록 설계되었으며 시스템 특성상 실시간 처리가 요구된다[12, 13].

이러한 요구사항을 만족할 수 있는 대형 임베디드

* 正會員, 韓國電子通信研究院
(ETRI)
** 正會員, 公州大學校
(Kongju University)

接受日字:2002年7月12日, 수정완료일:2003年1月29日

시스템으로서 초고속정보통신서비스 및 음성, 데이터, 화상 등 통합 멀티미디어 서비스 제공이 가능한 대형 ATM 교환 시스템이다. 본 시스템은 실시간 OS 및 DBMS를 적용한 시스템으로써 미들웨어(middleware)를 채택하고 개방형 구조(open architecture)의 객체지향(object oriented) 기법 도입 및 UML(Unified modeling language)을 이용한 모델링 기법을 채택하여 개발한 임베디드 시스템으로 그 구조는 <그림 1>과 같다. 위와 같은 대형 시스템을 개발하는 과정은 매우 복잡하며, 사용자가 원하는 시기에 안정성과 신뢰성을 갖는 시스템을 개발하여 적기에 인도하는 것은 매우 중요하다.

소프트웨어를 효율적으로 개발하려면 품질, 납기, 비용의 관점에서 소프트웨어의 각 공정에 대해 검토를 하는 개발관리 기술이 필요하다. 특히, 개발의 최종 단계인 시험단계는 시험 진행상황과 사용자에게 소프트웨어를 배포할 시기를 고려하여야 한다. 즉, 개발중인 소프트웨어에 대해 시험을 중단하고 사용자에게 소프트웨어를 인도할 시기를 정하는 방법은 개발관리 기술의 중요한 요소로 작용하며, 이것을 소프트웨어의 최적 배포문제(Optimal Software Release Problem) 라고 부른다[1].

과거의 소프트웨어 배포 문제는 납기를 중요시하여 개발관리자의 경험이나 판단에 의해서 좌우되는 경향이 많았다. 그러나 최근에는 최적 배포시기를 결정하는 평가기준은 소프트웨어 신뢰도 성장모델로부터 도출된 신뢰성 평가 척도와 총 기대되는 개발비용을 고려하여 결정된다. 이와 같은 방법은 소프트웨어를 사용자에게 인도하기까지 여러가지 측면을 고려하여야 하는 골치 아픈 문제로 시험시작후 얼마나 오랫동안 시험을 하여야 하는지를 결정하는 방법으로 다양한 운용환경에 의한 시험환경에서 고장의 발견 및 수정, 비용등을 고려하여야 하기 때문이다.

본 논문에서는 제 2장에서 배포시기 결정법에 대해 알아보고 3장에서는 신뢰도성장모델 중 시스템 개발초기에 적합한 모델인 지수형 성장모델에 근거하여 개발비용과 신뢰도를 고려한 최적 배포시기를 결정한다. 그 밖에 시스템내의 잔존 결함수와 평균고장간 시간(MTBF)등을 추정하여 최적배포시기를 상호 비교해 본다. 끝으로 결론 및 향후 연구방향에 대해 제시하고 맺는다.

Ⅱ. 소프트웨어 배포시기 결정법

1. 평가척도에 의한 배포시기 결정 방법[11]

소프트웨어 신뢰도 성장 모델에 근거한 최적 배포 문제는 모델로부터 도출한 정량적인 신뢰성 평가 척도를 설정된 목표치에 도달하는데 소요되는 총 시험 시간을 최적 배포시기로 삼는다. 즉 시험마다 검출된 결함 데이터를 누적시켜 축적된 결함수를 토대로 소프트웨어의 신뢰도를 계산해 내는 개수계측 모델로부터 고장 발생에 대한 평균치 함수 H(t)와 고장강도 함수(FDF : Failure Density Function)인 h(t)를 통한 NHPP (Non Homogeneous Poisson Process) 모델을 고려할 수 있다. 이 모델의 신뢰성 평가 척도는 아래와 같이 3가지로 표현된다.

■ 기대 잔존 에러수

$$n(t) = a - H(t) \tag{1}$$

a : 시험 전 소프트웨어 내 잔존 에러수

■ 소프트웨어 신뢰도[R(x|t)]

$$R(x \mid t) = e^{-H(t+x)-H(t)} \tag{2}$$

■ 평균 고장시간 간격(MTBF)

MTBF : Mean Time Between Failure

$$MTBF(t) = \frac{1}{h(t)} \tag{3}$$

위의 식 (1)과 식 (3)에 대해 t 마다 시험을 종료할 때 기대잔존 에러수 및 평균 고장시간 간격을 고려해 볼 때 그때마다 각각의 목표치를 n_0, M_0 라고하면 식 (2)는 시험시각 t 일때 시험을 종료하고 이때 운용시간 x에 대한 소프트웨어 신뢰도를 R(x|t) 라고 하면 시험의 진척상황에 따라 계속 소프트웨어 내의 잔존 에러수는 감소하는 경향을 띤다. 즉, 소프트웨어 신뢰도와 평균 고장시간 간격은 증가하게 되는 것을 의미한다.

이때 소프트웨어 최적 배포시기 T* 는 각각

$$\min[t \mid n(t) \le n_0],$$

$$\min[t \mid R(x \mid t) \geq R_0],$$
$$\min[t \mid MTBF(t) \geq M_0]$$

를 만족하는 시험시간을 의미한다. 위의 식들은 총 시험시간을 T로 했을 때

$$n(T) = n_0,$$
$$R(x \mid T) = R_0 (x \geq 0),$$
$$MTBF(T) = M_0$$

를 만족할 때 T = T_1을 최적 배포시기라 한다. 구체적으로 말하면 NHPP(Non Homogeneous Poission Process)에 의거한 지수형 신뢰도 성장모델(ESRGM : Exponential Software Reliability Growth Model)에 대한 최적 배포시기는 $H(t) \cong m(t) = a(1 - e^{-bt}), (a, b > 0)$로 표현되고 이때 위의 식들은 아래의 식

$$T_{11} = \frac{1}{b} \ln[\frac{a}{n_0}],$$
$$T_{12} = \frac{1}{b} \ln[\frac{-m(x)}{\ln R_0}],$$
$$T_{13} = \frac{1}{b} \ln[abM_0]$$

로 표현할 수 있다. 이때 최적 배포시기가 결정되는데 그 조건은 다음과 같다.

$$n(0) \geq n_0,$$
$$R(x \mid 0) \leq R_0,$$
$$MTBF(0) < M_0$$

2. 비용 평가기준에 의한 최적배포 방법

소프트웨어 신뢰도 성장 이론에 근거하여 관측된 고장 데이터에 대한 해석은 실제 문제로서 매우 흥미 있는 일이다. 즉, 소프트웨어 최적 배포시기를 정하는데 있어서 상반된 요인에 대한 고려를 할 수 있는데, 소프트웨어 내에 잠재하고 있는 에러수로 평가된 소프트웨어 신뢰도와 에러를 발견, 수정하는데 시험에 투입된 공수(工數)나 유지보수 비용과의 관계이다.

시험에 투입되는 시간이 길어지면 소프트웨어 내의 에러들이 많이 발견되어 운용단계에 이르면 신뢰도는 증가한다. 이와 반대로 소프트웨어의 운용이 지연되면 시험에 허비되는 시간이 많아지고 비용이 증대하게 된다. 즉, 운용단계에 발견되는 에러가 많을수록 유지보수 비용이 많아진다.

소프트웨어 최적 배포시기를 결정하기 위한 최적 배포 문제를 구하기 위해서 수식화 해보면 시험시간이 T일때 총 기대 소프트웨어 비용은 아래와 같이 정의된다.

$$C(T) = C_1 H(T) + C_2[H(T_{LC}) - H(T)] + C_3 T$$

T_{LC} : software life cycle, (일반적으로 $C_2 > C_1$ 임)
C_1 : 시험단계에서 발견된 에러 당 수정비용
C_2 : 운용단계에서 발견된 에러 당 수정비용
C_3 : 시험에 소요되는 단위시간 당 비용

평균치 함수 m(t)를 지수형 신뢰도 성장모델로 가정하면 위의 식은

$$C(T) = C_1 m(T) + C_2[m(T_{LC}) - m(T)] + C_3 T$$

가 된다. 이것을 시간 T에 대해 미분하면

$$h(T) = \frac{dm(T)}{dT} = abe^{-bT},$$
$$\frac{dC(T)}{dT} = -(C_2 - C_1)h(T) + C_3$$

가 된다. 이것을 다시 정리하면

$$h(T) = \frac{C_3}{C_2 - C_1}$$

이 된다. 이것은 단위 시험시간 당 발견되는 에러수를 의미하는 강도함수가 된다.

즉, T = 0일때

$$h(0) = ab,$$
$$h(0) \geq \frac{C_3}{C_2 - C_1}$$

이 되어 h(T)에 대한 유한값의 해를 구하면

$$T_0 = \frac{1}{b} \ln[ab \frac{C_2 - C_1}{C_3}]$$

가 된다. 결론적으로 시험시간 T에 대해 최적배포시기를 정리하면

$$0 < T < T_0 : \frac{dC(T)}{dT} < 0,$$
$$T > T_0 : \frac{dC(T)}{dT} > 0$$

가 된다.

즉, 최적배포시기 T* 는 $T^* = \min[T_0, T_{LC}]$ 로 정리된다. 위와 같이 소프트웨어 개발비용에 대한 연구[1-6]는 여러 차례 제기되었으며, 개발비용을 평가기준으로 삼아 소프트웨어의 최적배포시기를 연구한 예도 많이 있다[2-3, 7-8].

3. 신뢰도와 평가기준을 동시에 고려한 최적배포

소프트웨어 신뢰도와 평가 기준을 동시에 고려한 최적 배포 시기 결정법은 신뢰성 평가 척도와 소프트웨어 비용을 가지고 달성된 신뢰도에 대한 비용을 평가해 보는 방법으로서 투입된 개발비용의 유효성(cost-effectiveness)을 파악해 보는 매우 흥미 있는 일이다.

NHPP 모델에 대한 평균치 함수 H(t)와 시험을 진행시켜 달성된 신뢰도 목표치 R0를 달성할 때 식 (6)의 총 기대 소프트웨어 개발비용 C(T)를 최소로 하는 경우의 총 시험시간을 정하는 것이다.

즉, 운용시간 x(x > 0)에 대해 정리하면

$$\min imize C(T),$$
$$R(x \mid T) \geq R_0 (T \geq 0)$$

를 만족하는 시험시간 T를 구하면 된다. 지수형 신뢰도 성장모델의 평균치 함수를 m(t)라고 하면

$$\min imize\{C_1 m(T) + C_2[m(T_{LC}) - m(T)] + C_3 T\},$$
$$e^{-[m(T+x)-m(T)]} \geq R_0, (T \geq 0)$$

로 정리되어 각 경우에 대한 최적 배포시기를 구하면 아래 식 (4) ~ 식 (7)과 같다.

$0 < C_1 < C_2, C_3 > 0, x \geq 0, 0 < R_0 < 1$ 이고

$h(0) > \frac{C_3}{C_2 - C_1}, R(x \mid 0) < R_0$ 를 만족하면

$$T^* = \max[T_0, T_1] \quad (4)$$

$h(0) > \frac{C_3}{C_2 - C_1}, R(x \mid 0) \geq R_0$ 이면

$$T^* = T_0 \quad (5)$$

$h(0) \leq \frac{C_3}{C_2 - C_1}, R(x \mid 0) < R_0$

$$T^* = T_1 \quad (6)$$

$h(0) \leq \frac{C_3}{C_2 - C_1}, R(x \mid 0) \geq R_0$

$$T^* = 0 \quad (7)$$

가 된다.

Ⅲ. 시스템 데이터 적용 결과

기존 시스템 개발에 적용된 방법은 시스템의 안정도에 근거한 품질 평가 척도에 근거한 고전적인 방법이 적용되어 왔다. 그러나 이러한 안정도 측면만을 고려하면 시스템 개발비용이 높아져 가격경쟁력에서 뒤지게 된다. 즉, 시스템 개발에 있어서는 개발비용과 신뢰도를 동시에 고려하지 않을 수 없기 때문에 본 장에서는 신뢰도와 개발비용 평가기준을 동시에 고려한 데이터 적용 결과를 분석해 본다. 먼저 이를 위해서 스위치 링크 속도가 155Mbps인 40G 용량의 선행 개발 시스템인 ACE256 시스템과 스위치 링크 당 2.5Gbps, 전체 80G 처리용량을 가진 후발 시스템인 ACE2000 시스템의 개발과정에서 검출된 고장데이터를 분석하여 소프트웨어 배포시기 결정을 위한 신뢰도 측정을 위한 데이터로 활용하고 또 관측된 시스템의 구조에 대해 간단히 살펴본다.

시스템의 신뢰도 평가를 위한 기초자료로 사용될 검출된 고장데이터 및 시험 방법 등에 대해 알아본다. 우선 지수형 신뢰도 성장모델에 근거한 대형 ATM 교환기의 소프트웨어 개발 시 관측된 고장 발견데이터를 적용하면 <표 1> 및 <그림 2, 3>와 같다.

1. 시험대상 시스템 구조

전체 시스템 구조는 크게 10개의 모듈로 구성되어 있다. 즉, ATM 정합을 위한 AIM(ATM interface module), 다양한 프로토콜과 연결방법을 통합 관리할 수 있는 MGM(media gateway module), 인터넷서비스를 수용하기 위한 MIM(MPLS interface module), Gigabit stream을 수용하기 위한 GIM(Giga bit interface module for 2.5G), 저속 및 Frame relay, ADSL(Asymmetric digital subscriber line)을 수용하는 AMM(Access multiplex module), 그리고 시스템의 핵심 스위칭 역할을 하는 SFM(Switch fabric module)과 통신망관리를 위한 TMN과 교환시스템의 운용관리를 위한 MAS(Maintenance & Administration System), 시스템 클럭을 공급하는 NSM(Network Synchronization Module) 등이 있다.

그 외에 추가 모듈로 광(optic) 모듈을 수용하는 WDM(wavelength division module), 음성 데이터 교환과 같은 지터(jitter)나 지연(delay)에 민감한 서비스를 수용하기 위한 모듈 즉, AAL 1/2(ATM adaptation

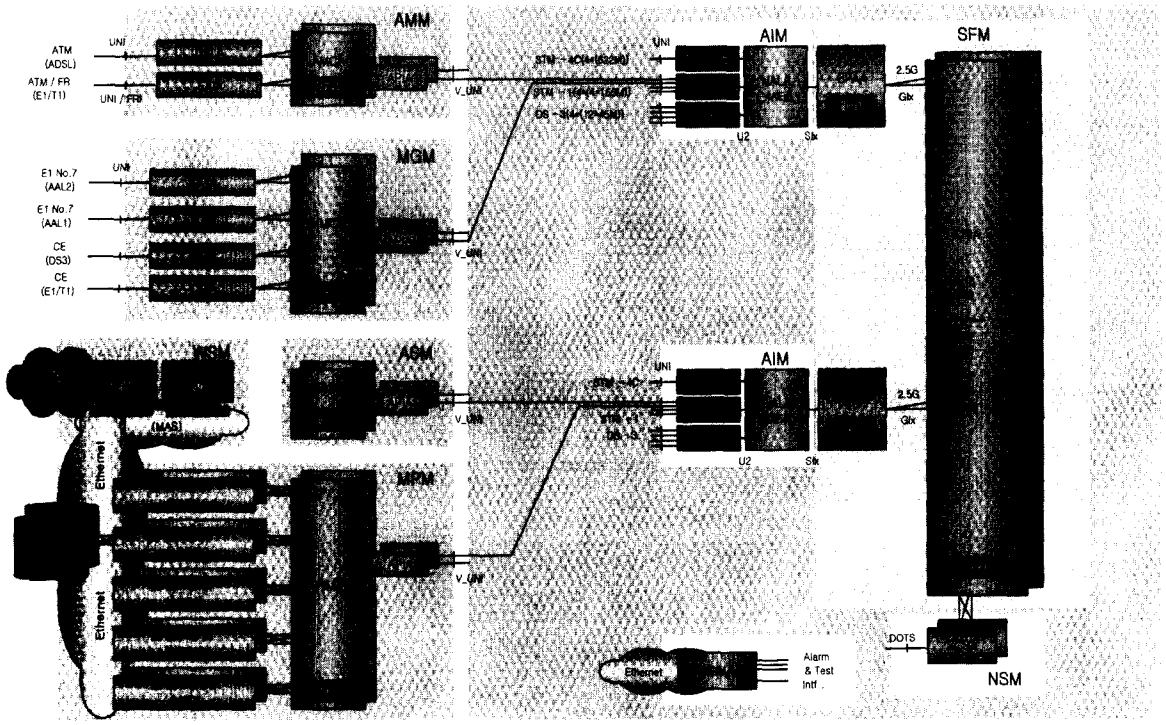


그림 1. 시스템 전체 구성도
Fig. 1. Overall of the system configuration.

layer ~ 1/2) Trunk 및 switching 기능 등을 처리해주는 ASM(AAL switching module) 들을 용도에 맞게 다양하게 구성 가능하도록 설계되어 있다.

시스템의 트래픽 처리용량은 20G ~ 160G까지 확장이 가능한 구조로 되어 있고 ATM 교환시스템에서 제공되고 있는 PVC(Permanent virtual path connection), SVC(switched virtual path connection)는 물론 PVC와 SVC의 혼합 형태인 Soft PVC, 사설망 정합을 위한 PNNI(Private network network interface) 기능을 제공하고 있는 대용량 ATM 교환시스템이다.

시스템의 본체와 운용자 터미널간은 미들웨어(Middleware)를 통해 ethernet 을 통해 통신하고 있다. 특히 미들웨어를 채택하고 워크스테이션에 운용 및 유지보수기능을 실장시킴으로써 시스템의 부하(load)를 분산시키고 다양한 OS환경을 하나의 통일된 환경으로 통합, 수용이 가능한 소프트웨어인 미들웨어를 채택한 것이 특징이다[10]. 즉, 본체에서는 교환시스템의 핵심기능인 호처리(call processing) 위주로 처리함으로써 ATM 교환기의 성능향상을 시도하고 기타 주기적인 job을 처리할 수 있는 운용, 유지보수기능을 시스템 외부에 두어 처리하도록 하여 기존의 시스템 개발개념인 하나의 시스템에서 모든 기능을 처리하도록 한 개념을 탈피하여 새로운 시스템 개발 능력을 보였다.

이 시스템의 전체 구성도는 <그림 1>과 같다.

2 시험 방법 및 고장 데이터 수집

고장데이터를 수집하기 위한 시험은 크게 기능담당자가 구현된 기능의 정상동작여부를 확인하는 기능시험(function test)과 소프트웨어 종합자가 각 기능들을 하나의 시험용 소프트웨어 패키지로 구성하여 수행하는 종합시험(Integration test) 그리고 품질보증을 위한 시스템 차원의 시스템시험(System test)으로 구분되어 수행된다. 소프트웨어 패키지는 통상 주 단위로 제작되어 시험되며, 전체 시스템의 소스 규모는 시스템 커널(System kernel)인 OS&DBMS가 300만 라인, 디바이스 제어용 Firmware가 100만 라인(ASIC은 제외), 응용프로그램이 150만 라인 등으로 소프트웨어 규모가 매우 큰 대형 교환 시스템이다.

각 시스템 별 소프트웨어 규모는 <표 1>과 같다.

시험시 검출된 세부 데이터는 <표 2>와 같으며, 매주 시스템 시험 시 검출된 초기 고장데이터를 토대로 평가한 축적 고장 데이터는 <그림 2, 3>에 나타내었다.

표 1. 시스템 별 소프트웨어 규모
Table 1. Scale of system software.

시스템 명	S/W 규모	블럭수	기능수
ACE256	240만 라인	133	250
ACE2000	550만 라인	149	147

표 2. 시스템 시험에서 검출된 고장 데이터 현황 - ACE256/2000 시스템
Table 2. Detection of the failure data in system test - ACE256/2000 System.

week	failure		week	failure		week	failure		week	failure	
1	3	7	7	2	7	13	7	9	19	8	4
2	1	11	8	8	6	14	4	4	20	7	2
3	0	19	9	7	5	15	6	2	21	2	-
4	2	8	10	7	9	16	13	0	22	1	-
5	4	11	11	3	6	17	8	1	23	1	-
6	5	5	12	1	5	18	19	3	24	3	-

* ACE256 시스템의 시험시작 24주 이후 고장데이터는 일부 제외
* 고장데이터(failure)란의 좌측은 ACE256, 우측은 ACE2000 시스템에 대한 데이터임

<그림 3>의 하나의 점은 5주 단위로 시스템 시험에서 수집된 고장 데이터의 실측치를 표시함)

위의 <표 1>의 시스템 시험에서 검출된 고장데이터를 가지고 지수형 신뢰도 성장모델에 근거하여 추정한 두 시스템의 초기 결함수(a1, a2)와 고장검출율(b1, b2)는 아래와 같이

$$\hat{a}_1 = 657.152, \hat{b}_1 = 0.0164414$$

$$\hat{a}_2 = 143.757, \hat{b}_2 = 0.0992314$$

로 추정되었다.

이에 대한 누적 고장데이터의 추정 결과는 <그림 2, 3>에 나타내었다.

고장 검출율은 ACE256 시스템에 비해 ACE2000 시스템에서 훨씬 높았다. 이는 유사한 환경에서 경험한 데이터의 축적과 개발 경험의 효과로 분석된다.

그러나 초기 시험 강도 및 시험, 개발자원의 투입 등이 고려되어야 하는 여러 변수가 작용하기 때문에 정확한 판단의 어려움이 뒤따른다.

이 데이터를 평균치함수 m(t)에 적용하면 각각

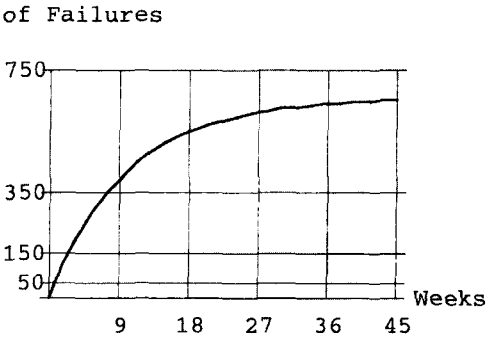


그림 2. 주별 초기 고장데이터 추정치 - ACE256
Fig. 2. Estimation of failure data per week - ACE256.

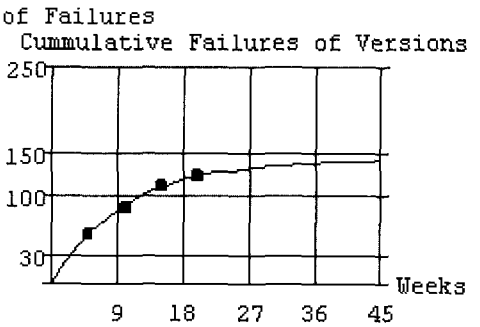


그림 3. 주별 초기 고장데이터 추정치 - ACE2000
Fig. 3. Estimation of failure data per week ACE2000

$$\hat{m}_1(t) = 657.152(1 - e^{-0.0164414t}),$$

$$\hat{m}_2(t) = 143.757(1 - e^{-0.0992314t})$$

가 되며, 이때 신뢰도 목표치를 $R_0 = 0.9$로 정하는 경우 이를 도달하기 위한 총 시험시간 $T = T^*$를 정할 수 있다. 즉, 운용시간 $x = 0.1$(주)에 대해 신뢰도 목표치 0.9에 도달하기까지의 최적 배포시기 T^* 를 구할 수 있다. 다시 말해서 $\min[T \mid R(0.1) \mid T] \geq 0.9]$를 만족하는 시간을 구하면 각각

$$T_1 = \frac{1}{0.01644} \ln[-\frac{\hat{m}(0.1)}{\ln(0.9)}] = 141.527$$

$$T_1' = \frac{1}{0.099} \ln[-\frac{\hat{m}(0.1)}{\ln(0.9)}] = 26.263$$ 가 된다.

$T_{LC} = 50$(주)인 경우 $C_1 = 1.0$, $C_2 = 5.0$, $C_3 = 50$(주) 일때 ACE256 시스템인 경우는

$$h(0) = ab = 657.152 * 0.01644 = 10.804$$
$$h(T) = \frac{c_3}{(c_2 - c_1)} = \frac{50}{5.0-1.0} = 12.5$$
$$T_0 = \frac{1}{0.01644} \ln[\frac{10.804*4}{50}] = 8.869$$

ACE2000 시스템인 경우는

$$h(0) = ab = 143.757 * 0.0992314 = 14.265$$
$$h(T) = \frac{c_3}{(c_2 - c_1)} = \frac{50}{5.0-1.0} = 12.5$$
$$T_0 = \frac{1}{0.099} \ln[\frac{14.265*4}{50}] = 1.334$$

가 된다. 즉, 2개 시스템에 대한 배포시기는 각각 T^* = min[T_0 , T_{LC}] = 9(주)와 2(주)가 된다. 다시 말해 ACE2000 시스템인 경우는 소프트웨어 배포 후 운용 시간은 T_{LC} - T = 50 - 2(1.334) = 48(주)가 되며, 여기서 총 기대되는 소프트웨어의 비용 최적치는 $C(T^*)$ =2576.818 가 된다. 이때 T^* = 1.334를 이용하는 경우 지수형 모델의 소프트웨어 신뢰도는

$$R(0.1 \mid T^*) = 0.75712,$$

(S-Shape Model인 경우는 0.7899)

로 되어 이는 초기 신뢰도 (R_0)보다 낮게 된다. 이때의 최적 배포시기는 신뢰도와 개발비용을 동시에 고려한 형태로 되어야 한다. 그러므로 이 경우는 아래의 식으로 표현되고

$$T^* = \max[T_0, T_{11,12,13}],$$
$$T_{11} = \frac{1}{b} \ln[\frac{a}{n_0}],$$
$$T_{12} = \frac{1}{b} \ln[-\frac{m(x)}{\ln R_0}],$$
$$T_{13} = \frac{1}{b} \ln[abM_0]$$

를 얻을 수 있다.

이 조건을 만족하는 두 시스템(ACE256/2000 시스템)의 최적배포시기(T_1)을 구하면 T_1 = 141.527 과 26.263 이 된다. 즉, 시험 시작 후 ACE256 시스템인 경우는 142주, ACE2000 시스템인 경우는 27주 정도 시험을 행한 후 소프트웨어를 배포하는 경우가 비용(Cost)과 신뢰도를 고려한 경우의 최적배포 시기임을 의미한다. 이 시기는 대략 시험시작일 기준 약 35개월과 7개월 후인 것으로 예상된다(<표 3> 참조).

가장 큰 이유는 시스템 시험시 검출되는 고장발견율과 시험환경 및 시험방법, 유사시스템에 대한 개발 경

표 3. 시스템 별 최적배포시기
Table 3. Optimal release time for systems.

시스템 별	최적 배포시기(week)
ACE256	142 (35 month)
ACE2000	27 (7 month)

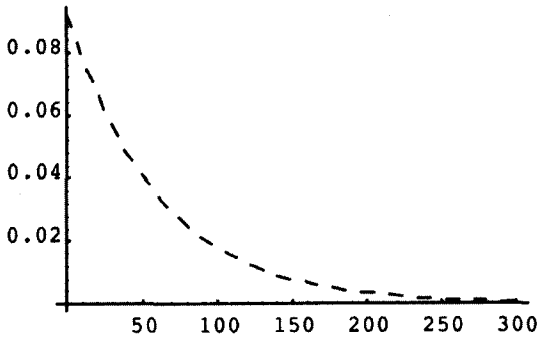


그림 4. 평균 고장 간 시간(MTBF) ACE256
Fig. 4. Mean time between failure - ACE256.

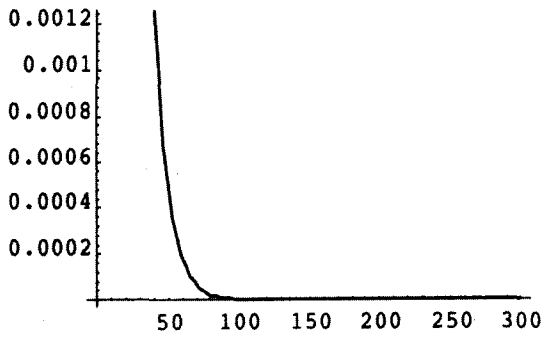


그림 5. 평균 고장 간 시간(MTBF) ACE2000
Fig. 5. Mean time between failure - ACE2000.

험등의 축적으로 인한 것으로 기인된다. 그러나 이 기간은 시험자원의 투입량에 따라 달라질 수 있으며, 시스템의 신뢰도에 따라 적절히 시험자원의 투입량을 조절할 수 있는 프로젝트 관리 기법이 필요하다. 즉, 시험능력(혹은 시험노력)을 고려한 평가 방법에 대한 것으로 이에 대한 연구결과가 기 발표된 바 있다.[9] 이때 시스템의 평균고장간 시간(MTBF : Mean Time Between Failure)은 주(week)당 시스템 별로 변화되는 추이 및 비교는 <그림 4, 5, 6>과 같다. (그림의 y 축은 고장수, x축은 시간(/week)을 의미함)

<그림 6>의 점선 부분은 ACE256 시스템의 MTBF 변화이고 실선은 ACE2000 시스템의 결과이다. 초기 고장데이터를 이용하여 시스템 전체의 신뢰도를 평가

하는데 다소 무리가 있지만 개발 경향을 파악(Trend Analysis)하는 관점에서 이에 대한 결과는 시스템 개발관리에 매우 중요하다.

이 그림에서 알 수 있듯이 선행 개발시스템인 ACE256 시스템보다 ACE2000 시스템의 MTBF가 급속히 줄어들어 경향을 보이는데 이는 유사한 시스템의 개발경험 축적은 물론 새로운 개발개념 및 툴의 도입, 기능통합 및 처리 등을 실시한 결과다. 그밖에 상용 칩(Chip)의 사용 및 MAS(Machine Administration System), 망관리 기능 등에 대한 일부 기능개발의 아웃소싱 등에 따른 시스템 개발기간의 단축 등에 따른 결과로 나타났다. <그림 6>에서 보다시피 두 시스템의 고장간 평균시간은 ACE256 시스템에 비해 ACE2000 시스템이 급격히 줄어들고 있는데 이 이유는 유사시스템의 개발경력과 소프트웨어 규모(소스코드) 대비 기능수의 축소 및 통합관리, 객체지향 및 UML(Unified Modeling Language) 설계 개념 도입, Middleware 채용으로 다양한 OS 환경의 소프트웨어 개발환경을 하나의 일관된 동작으로 통합관리하고 교환시스템의 핵심기능과 주변기능을 분리 수용하는 기능개발 정립을 통해 전체 시스템의 개발개념을 통일화하였다. 즉, 핵심 호처리 기능 등은 시스템에 내장시키고 비 실시간 처리 위주의 주기적인 job 수행 등 M&A 기능은 워크스테이션에 수용함으로써 시스템의 부하를 분담시키고 원활한 기능처리와 에러 발생을 줄여 기존 개발시스템과의 차별화된 새로운 개념의 시스템 개발 개념 도입한 결과로 표출되었다.

그밖에 시스템 내에 잔존하고 있는 고장수[n(t)]와 고장 강도를 나타내는 평균치 함수 m(t)를 비교해보면 아래 <그림 7, 8>로 표현할 수 있으며, 신뢰도를 고려하여 시스템 내의 잔존고장수[n(t)]와 평균치 함수만을 고려하면 ACE256시스템은 80주 전후, ACE2000시스템인 경우는 시험시작 후 약 60주 근처에서 배포하는 것이 최적인 것으로 밝혀진다. (<그림 7, 8> 참조)

즉, 초기 개발시스템(ACE256시스템)보다 최종개발시스템(ACE2000시스템)의 고장 검출율이 높은 원인은 유사 시스템 개발 경험의 축적 및 교환시스템 기능 구현시 프로그래밍 기법에 대한 숙련도 등으로 인한 고장 발견 및 수정의 Exposure 시기가 많이 앞당겨지고 있음을 <그림 7, 8>에서도 확연히 알 수 있다.

위와 같이 시스템 소프트웨어 최적배포시기의 tradeoff로써 여러 경우의 수를 고려하여야 함으로 프로젝트 관리자는 배포시기 판단에 많은 어려움이 따른다. 또, ACE2000 시스템의 최적 비용을 산출하면 $C(T^*)$ = 1494.55 가 된다. 이것은 소프트웨어의 총 기대 비용이 소프트웨어의 비용만을 평가기준으로 고려하는 경우에 비해서 (2576.818 - 1494.55) = 1082.268의 차이가 난다. 다시 말해서 소프트웨어 비용과 신뢰도 측정치의 양면을 고려한 모델이 유리함을 알 수 있다.

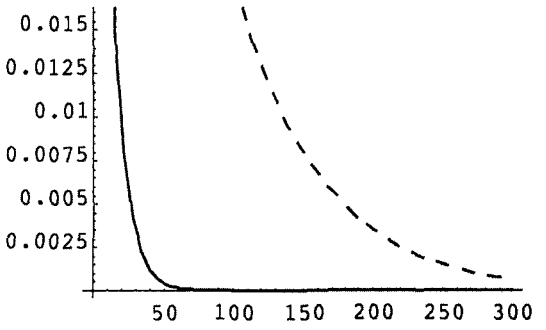


그림 6. 시스템 별 평균 고장 간 시간(MTBF)
Fig. 6. Compare of MTBF - ACE256/2000 system.

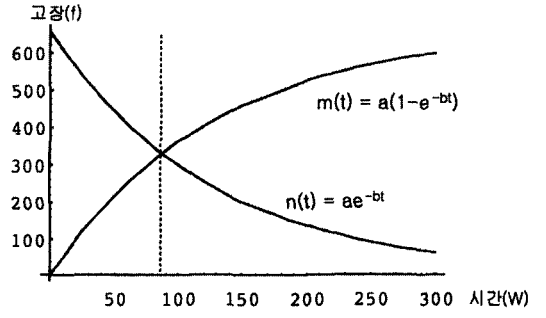


그림 7. 잔존고장과 평균치 함수와의 관계 - ACE256
Fig. 7. Latent fault vs m(t) function ACE256.

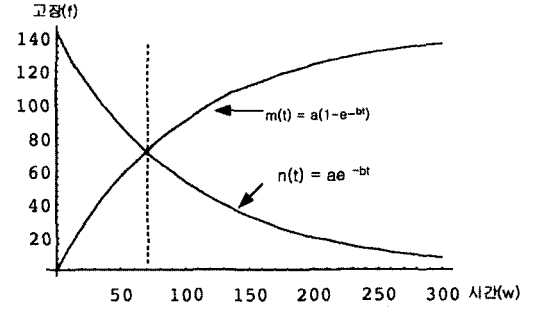


그림 8. 잔존고장과 평균치 함수와의 관계 ACE2000
Fig. 8. Latent fault vs m(t) function ACE2000.

3. 고장 데이터 관리 및 분석

고장데이터 관리는 시스템 형상관리 시스템과 연동하여 변경이력을 관리하는데 이에 대한 전용 툴을 별도로 두어 버전 별/ 프로젝트 별/ 개발자 별/ 세부 파일 별로 다양하게 관리할 수 있다.

교환소프트웨어 변경관리 기능은 개발자가 디버깅을 위해 혹은 추가 기능 요구 사항의 구현을 위해 필요한 여러 정보를 편리하게 작성할 수 있도록 도와주는 제반 기능과 이를 일괄적으로 등록하고 추적할 수 있는 기능들을 포함한다

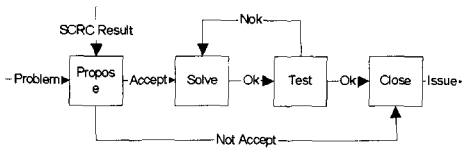
<그림 10>은 검출된 문제점에 대한 처리 방법으로 개발자와 관리자간에 역할분담으로 웹 및 도스 환경이 지원되는 관리시스템의 상태 천이 다이어그램이다.

문제점 처리에 대한 전체적인 흐름도는 <그림 10>과 같으며, 이를 처리하는 데 사용되고 있는 관리시스템은 시스템 전체 형상을 관리하는데 사용되고 있는 형상관리도구와 연동된 DDTS(defect data tracking system) 로 교환 시스템 개발 프로젝트 성격에 맞게 윈도우/웹 /도스 환경에서 동작될 수 있도록 자체 보완 개발된 시스템이다.

문제점 관리를 위한 전용 툴은 기존에 UNIX 환경에서만 지원이 가능한 ACE256 시스템 프로젝트 관리에 사용되어 왔던 소프트웨어변경이력관리시스템(<그림 9> 참조)인 CRMS(Software Change Request Management System)의 문제점을 좀더 보완하고 사용의 편리성을 추가한 툴로서 다양한 환경(DOS/UNIX/WEB/WINDOWS 환경 지원 가능)의 지원이 가능하고 우리 Project에 맞게 한글화 및 상태제어를 재조정하여 자체 보완 개발된 수정 버전의 한글판 DDTS가 사용되고 있다.[<그림 13> 참조]

시스템의 문제점 관리 및 처리를 위한 전용시스템의 상태처리 흐름도는 <그림 10>과 같으며, 실제 프로젝트 수행에 Duplication(D) 상태는 부득이한 경우를 제외하고 가급적 지양(止揚)하여 발생된 문제점에 대한 정확한 분석 및 조치, 사후관리를 철저히 함으로써 시스템의 품질 향상을 꾀하였다. 또한 온라인 점검을 통해 지연 되고 있는 문제점(P : postpone)에 대해서는 개발일정에 부합되도록 독려하여 개발기간 단축 및 비용을 줄이는데 힘썼다.

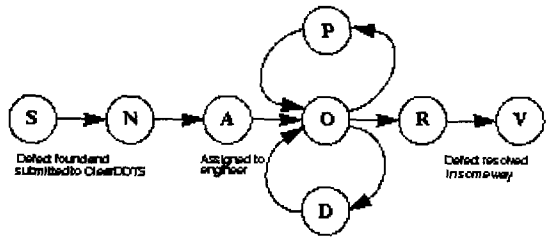
문제 해결 현황에 대한 아래 그림은 시스템 시험 시 검출된 문제점에 대한 처리 현황으로 전체 문제점 중 약 80%가 소스 프로그램 문제로 분석되었으며, 8% 정



CRMS Flow Diagram

SCRC : Software Change Review Committee

그림 9. CRMS의 상태 천이
Fig. 9. Control of state in CRMS.

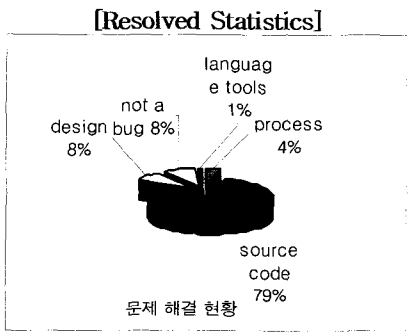


S : Submitted N : New A : Assign O : Open
D : Duplication P : Postpone R : Resolve V : verify

그림 10. 문제점 처리 상태 흐름도
Fig. 10. Problem state transition diagram.

도가 설계 결함으로 판명되었다. 초기 설계 결함의 발생 빈도가 높은 문제는 시스템의 신뢰도에 많은 영향을 미치기 때문에 초기 설계에 많은 노력을 기울여야 하는 문제점을 남겼다. 해결 방법상의 문제점들은 주로 소스 코드의 에러로 인한 문제 발생이며, 버그가 아닌 8 %의 문제점은 교환기능 중 소프트웨어 버전관리기능을 추가하기 위한 변경사항으로 이점은 기능추가로써 별도로 분리하고 있다(그림 [Resolved Statistics] 데이터 참조).

그외 제시한 통계데이터는 ATM Project에서 관리하고 있는 문제점 중 발견 방법, 해결방법 및 문제점의 심각성에 대해 분석한 결과이다.



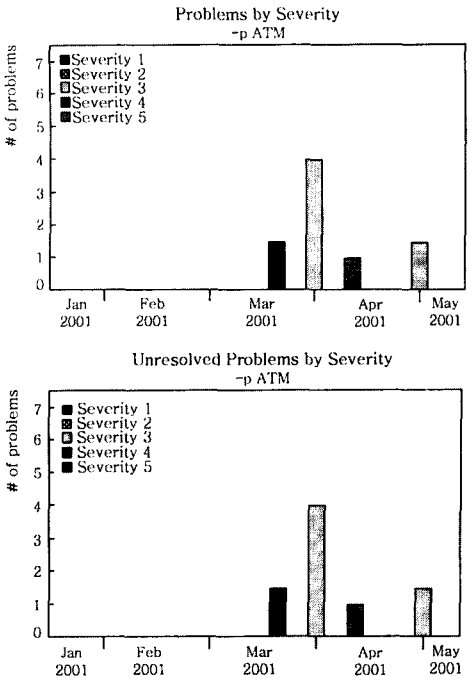


그림 11. 심각성에 의한 월별 문제점 제기 및 미해결 상황

Fig. 11. Problems by Serverity.

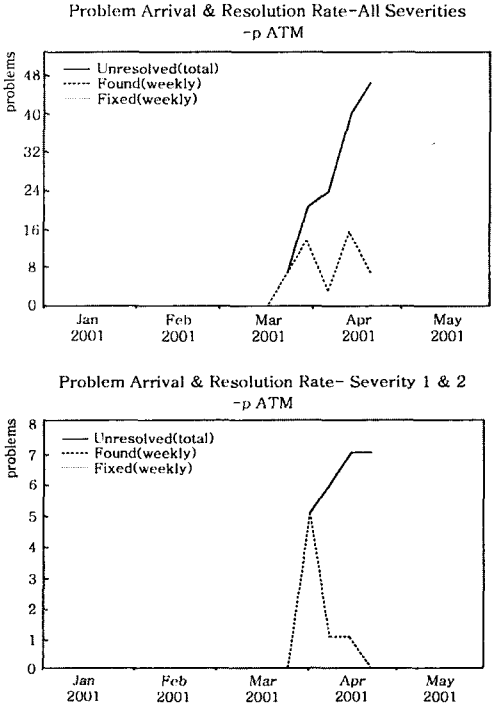


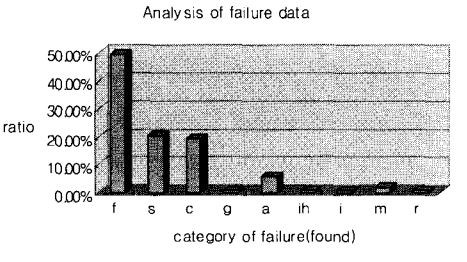
그림 12. 주별 문제점 발견 추이와 전체 미해결 현황

Fig. 12. Problem Arrival & Resolution Rate per week.

이 결과를 살펴보면 문제점 발견은 대체로 기능시험 과정에서 절반이 걸러지며 시스템시험과 사용자 추가 요구사항에 의한 기능 수정이 40 % 정도를 차지함으로써 low level의 검증보다는 상위 레벨에서의 기능검증이 활발히 이루어지고 있음을 알 수 있었다. 이점은 개발 초기의 세부 설계단계에 대한 고려(review)가 절실히 요구됨을 의미한다.

다시 말해서 초기 단계에 수행하여야 할 품질개선활동이나 툴에 의한 기능 검증이 좀더 요구된다. 이러한 점이 추후 시스템 개발 시 고려하여야 할 개발환경상의 문제로 높은 신뢰성을 갖는 교환시스템 개발환경구축의 풀어야 할 숙제이다.

ATM Project 문제점 분석 통계 데이터 예[Found Statistics]



Index : functional test(f) system test(s) customer use(c) group code review(g) author code review(a) in-house normal use(ih) interactive test(i) manual review(m) random unplanned test(r)

아래에 제시된 Severity 별 분포 내역은 검출된 문제점의 심각성에 따라 분류된 것이다. 즉, 운용상에 지장이 없는 사소한 문제점부터 Critical 한 문제점까지 5단계(severity 1 ~ 5, 높을수록 심각성이 많은 문제점을 의미함)로 구분함으로써 처리하고 있는 문제점의 비중을 파악할 수 있도록 하여 우선순위에 따른 개발일정의 완급을 조절하도록 조치하였다. 이렇게 함으로써 프로젝트 참여자에게 개발에 필요한 정보를 제공함으로써 개발의 용이함과 편리성을 제공하였다.

대체로 문제점들은 3주 이내에 해결할 수 있는 사소한 문제들이 95% 정도 차지하고 있으며 조치시간(correct time)이 긴 심각성을 띤 문제점은 전체의 5% 정도인 것으로 분석되었다.

발견된 문제점에 대한 심각성 및 고장 발견, 해결에 대한 추이는 <그림 11, 12>와 같다.

[Severity Statistics]

severity 1 Problems = 13.89%
severity 2 Problems = 2.78%
severity 3 Problems = 77.78%
severity 4 Problems = 2.78%
severity 5 Problems = 2.78%

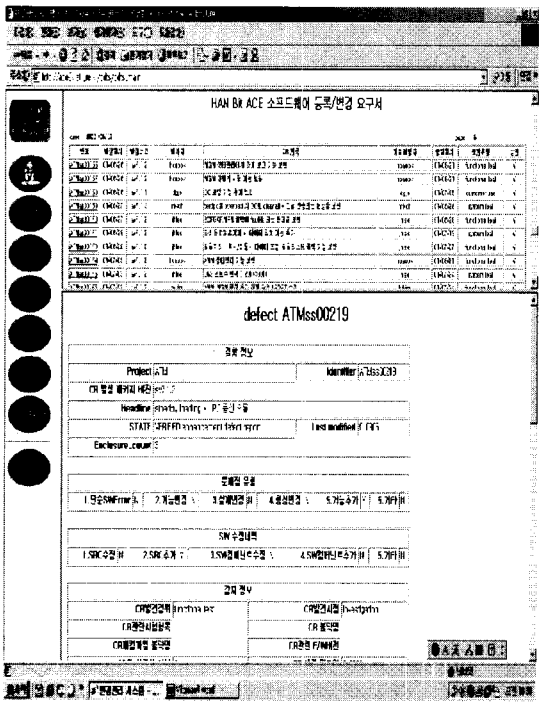


그림 13. ACE2000 시스템 변경관리 시스템 구성도
Fig. 13. Configuration of DDTS for ACE2000 System.

Ⅳ. 결론 및 향후 연구방향

본 논문에서는 널리 사용되는 소프트웨어 신뢰도 성장 모형의 대표적 모델 중 NHPP 모델에 근거한 소프트웨어 신뢰도 성장 모델을 이용하고 소프트웨어 개발관리의 응용으로서 소프트웨어 최적 배포문제를 언급하였다. 최적 배포문제는 소프트웨어의 시험 공정관리에 의한 진보관리 문제와 시험능력의 최적 배분에 의한 배포시기 결정 방법이 있다.

본 논문에서는 소프트웨어 평가 기준에 의한 소프트웨어 신뢰도 성장모델로부터 도출된 신뢰도 평가척도를 가지고 사용자가 원하는 소프트웨어 신뢰도 목표치를 만족하는 납기를 예측해 보았다. 또 검출된 고장데이터를 다각도로 분석하여 좀더 정확한 배포시기 결정을 위한 신뢰성 데이터를 적용하도록 했다.

향후 연구 방향은 구체적이고 정밀한 에러 발견 사상 등을 포함한 소프트웨어 최적 배포시기를 결정하는 방법이다. 실제 소프트웨어 개발에 투입된 비용을 분석하여 예측된 추정치와 상호 비교 연구 등이 필요하다.

참 고 문 헌

[1] S. Yamada, "ソフトウェア信賴性評價技術 : ソフトウェア信賴度成長モデル入門", HBJ Publishing, Integrated libraries No.42, pp.195 206, 1989.4.

[2] E. H.Forman and N. D. Singpuwalla, "Optimal time interval for testing hypotheses on computer software errors", IEEE Trans. Reliability, Vol. R 28, No. 3, pp.250 253, Aug. 1979.

[3] H. S. Koch and P. Kubat, "Optimal release time of computer software", IEEE Trans. Software Eng., Vol SE 9, No. 3, pp.323 327, May 1983.

[4] S. Yamada and S. Osaki, "Optimal software release policies with simultaneous cost and reliability criteria", European J. Operational Research, Vol. 31, No. 1, pp.46 51, 1987.

[5] R. W. Wolverton, "The cost of developing large scale software", IEEE Trans. Computers. Vol. C 23, No. 6, pp.615 636, Jun. 1974.

[6] K. Okumoto and A. L. Goel, "Optimal release time for software system based on reliability and cost criteria", J. System and Software, Vol. 1, No. 4, pp.315 318, 1980.

[7] 大寺浩志, 山田, 成久洋旨, "ソフトウェアの 運用段階におけるエラー發見事象を 考慮した最適リリース", 電子情報通信學會論文誌, Vol. J71 D, No.7, pp. 1338 1340, Japan, 1988.7.

[8] S. Yamada and S. Osaki, "Cost reliability optimal release polices for software system", IEEE Trans. Reliability, Vol. R 34, No. 5, pp.422 424, Dec. 1985.

[9] 이재기, 신상권, 홍성백, 윤병남, "시험노력을 고려한 개발단계의 소프트웨어 신뢰성 평가", 대한전자공학회논문지, 제36권 S편 제3호, pp. 18

26., Mar. 1999

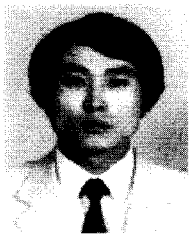
[10] 이재기, 신상권, 남상식, "미들웨어와 UML을 활용한 교환소프트웨어의 개발과 관리", ETRI 전자통신동향분석지 제16권 제5호, pp. 1 10., Oct. 2001

[11] 山田 戌, 大寺浩志, "ソフトウェア信頼性 ~ 理論と實踐的應用 ~ ", SEツリ ズ, ソフト.りサ チ.センタ, pp. 317 323, 1990. 2.

[12] Embedded Systems(http://www.Embedded.com), 2002.11.

[13] Edwared A. Lee, "What's Ahead for Embedded Software ?", IEEE Computer, pp. 18 26, 2000. 10.

저 자 소 개

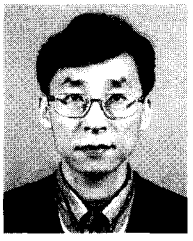


李 在 起(正會員)

1985년 2월 : 서울산업대학교 전자공학과(공학사). 1989년 5월 : 청주대학교 대학원 전자공학과(공학석사). 2002년 : 공주대학교 전기전자정보통신공학과 박사과정. 1983년 3월 : 현재 한국전자통신연구원 책임연구원. <주관심분야 : 소프트웨어 신뢰도, 시험 및 검증, 소프트웨어 품질 향상>

南 相 植(終身會員)

1981년 2월 : 단국대학교 전자공학과 졸업(공학사). 1983년 2월 : 단국대학교 대학원 전자공학과 졸업(공학석사). 1999년 2월 : 단국대학교 대학원 전자공학과 졸업(공학박사). 1985년 3월~현재 : 한국전자통신연구원 책임연구원(팀장). <주관심분야 : ATM Technology, Signal Integrity>



朴 鐘 大(正會員)

1987년 2월 : 영남대학교 전자공학과 학사. 1989년 2월 : 영남대학교 전자공학과 석사. 1994년 8월 : 영남대학교 전자공학과 박사. 1997년 1월~현재 : 한국전자통신연구원 선임연구원. 1995년 8월~1996년 9월 : Toyohashi Univ. of Technology Post Doctor. <주관심분야 : VLSI설계, MWP/MMWP, Optical Interconnection>

金 昌 奉(終身會員)

1983년 : 고려대학교 전자공학과(공학사). 1988년 : Florida Tech 대학 전자공학과(공학석사). 1992년 : Texas A&M 대학 전자공학과(공학박사). 1993년~현재 : 공주대학교 정보통신공학과 교수. 2000년~현재 : 미국전기전자공학회(IEEE) Senior Member. <주관심 분야 : 광통신망, 광전송>