고장 분석과 교환 소프트웨어의 최적 배포

An Analysis of Failure Data and Optimal Release Time of Switching Software

이재기(J.K. Lee) 시스템종합팀 선임연구원
신상권(S.K. Shin) 시스템종합팀 연구원
이수종(S.J. Lee) 시스템종합팀 책임연구원
남상식(S.S. Nam) 시스템종합팀 책임연구원, 팀장

본 논문은 ACE2000 시스템 소프트웨어의 Release 시점을 예측할 수 있는 최적 배포문제로, 시스템의 안정도를 평가해 볼 수 있는 측면에서 소프트웨어 최적 배포문제에 대해 살펴보고 평가기준을 제시하여 제품의 적기 공급 및 개발자원의 효율적 이용 측면을 분석한다. 즉, 신뢰성 평가척도와 개발비용을 고려한 최적 배포문제를 기술하였다. 또 여러 가지 소프트웨어 신뢰도 성장모델 중 지수형 모델을 근거로 한 소프트웨어 개발비용과 신뢰성 평가기준을 고려한 배포시기를 결정하여 보았다.

I. 서 론

소프트웨어를 효율적으로 개발하려면 품질, 납기, 비용의 관점에서 소프트웨어의 각 공정에 대해 검토를 하는 개발관리 기술이 필요하다. 특히, 개발의 최종단계인 시험단계는 시험 진행상황과 사용자에게 소프트웨어를 배포할 시기를 고려하여야 한다. 다시 말해서 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 배포문제는 배포시기 즉, 납기를 중요시하여 개발관리자의 경험이나 판단에 의해서 좌우되는 경향이 많았다. 그러나 최적 배포시기를 결정하는 평가기준은 소프트웨어 신뢰도 성장모델로부터 도출된 신뢰성 평가척도와 총 기대되는 개발비용을 고려하여 결정된다.

본 논문에서는 제II장에서 ACE2000 시스템 개발중에 발생된 제반 문제점들을 분석하고, III장에서 이를 정리하여 신뢰도 성장모델 중 개발초기에 적합한 모델인 지수형 성장모델에 근거하여 개발비용과 신뢰도를 고려한 최적 배포시기를 결정하고, IV장에서는 결론 및 향후 연구방향에 대해 제시하고 맺는다.

II. 소프트웨어 문제점 분석

1. 소프트웨어 관리 및 문제점 관리

교환기 개발에 활용되고 있는 각종 소프트웨어 컴포넌트들에 대한 체계적이고 효율적인 관리와 개발 전 과정(Life Cycle)에서 발생되고 있는 문제점들에 대한 전체 프로젝트 관리 측면의 도구로 Rational사의 ClearCase(Clear Comprehensive Software Configuration Management System)와 Clear DDTS(Clear Distributed Defect Tracking System)가 이용된다[2],[3]. 이 툴을 이용하여 관리되고 있는 버전별 소프트웨어 규모 및 문제점에 대한 처

<표 1> 버전별 소프트웨어 관리 현황

버전 명	규모(KLOC)	비고
SV7.1.1(A)	54	1차 기본기능
SV7.1.1(B)	47.2	1차 핵심기능
SV7.1.2(A)	58	1차 추가기능
SV7.1.2(B)	71.4	2차 추가기능

리 현황은 <표 1>과 같다.

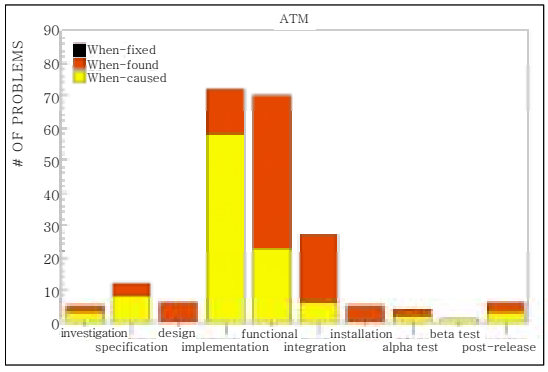
버전별 특징을 간략히 소개하면 SV7.1.1 1차 핵심기능의 소프트웨어는 AIM(ATM Interface Module)을 통한 PVC(Permanent Virtual Connection) 위주의 기본 호 처리 및 운용보전기능이 포함된 소프트웨어인 반면에 SV7.1.2 추가기능 패키지는 AM(Access Module), MG(Media Gateway Module)가 포함된 소프트웨어이다. 이때 추가된 기능으로는 F/R 및 저속 ATM의 통합관리, Soft-PVC, B-ICI 및 B-ISUP간 중계호 연동기능, MG/AM 가입자 상태관리 및 유지보수기능, 시스템 형상 확장에 따른 형상관리기능의 보완 등이 추가 또는 수정되었다.

SV7.1.1 버전의 두 개 버전은 2001년 3월 말 Library 통합 및 기능 재정립에 의한 관련 소스 변경으로 2001년 4월경에는 3월에 비해 소프트웨어 소스 라인 수가 일시적으로 줄어들었다. 이후 신규 개발기능의 추가에 따른 소스 라인 수가 지속적으로 증가하였다.

SV7.1.2 패키지는 F/R 연동 및 저속(2Mbps), PSTN 연동 및 AAL2 트렁킹(스위칭 포함), ADSL 정합기능들이 포함된 버전으로 SV7.1.1 버전이 AIM(622/155/45Mbps 정합) 위주의 가입자 정합기능을 갖춘 대형 ATM 코어용 교환기 소프트웨어인 반면에 SV7.1.2 버전은 다양한 가입자 정합기능을 제공하는 에지용 교환 소프트웨어이다.

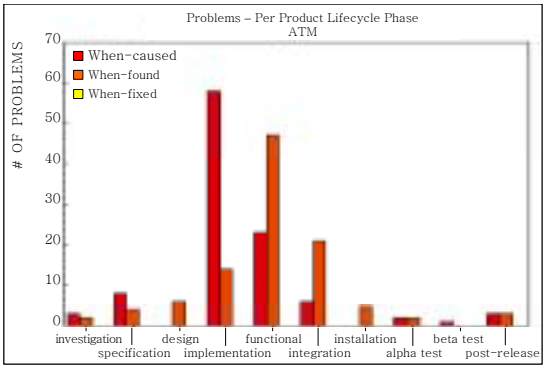
문제점 관리 및 추적 시스템으로 활용되고 있는 ClearDDTS는 Project 특성에 맞게 환경을 변경하고 개발 참여자의 이용률을 극대화하고 문제점에 대한 파악을 용이하게 하기 위해서 한글지원 기능을 도입하였다(ClearDDTS는 한글지원이 불가능하며 영문만 지원함). 이 결함 추적 시스템을 활용하여 버전별, 프로젝트별 이력관리를 수행하는데 이 활동은 소프트웨어 개발에 많은 정보를 제공해주고 있다. 즉, 소프트웨어 결함 추적을 위한 변경관리는 "소프트웨어 개발 과정에서 생겨난 각종 문제점들을 도출하여 이를 제안, 검토, 해결, 시험 및 종결에 이르는 상태를 추적해 나감으로써 개발 관리를 효율적으로 수행하기 위한 활동"으로 정의함으로써 소프트웨어 개발 과정에서 발생되는 각종 문제점들을 종합 정리하고 해결 상태를 계속 추적해 나감으로써 소프트웨어 등록/변경 요구서 관리 업무를 일원화하고 개발관리 업무의 효율을 높인다.

ACE2000 시스템 개발에 활용되고 있는 변경관리 시스템은 웹 인터페이스, X-window, TTY 인터페이스, Command line 인터페이스 모드를 지원해 주는 한편 GUI 환경을 제공해 주어 사용이 편리하다. 그 외에 툴에서 제공되고 있는 웹 인터페이스의 사용방법을 설명해 줌으로써 이용이 편리하다. 또한 한글지원이 가능한 보완 툴로 관리되고 있는 ATM Project의 제반 문제점 처리 현황은 (그림 1) ~ (그림 4)와 같다.

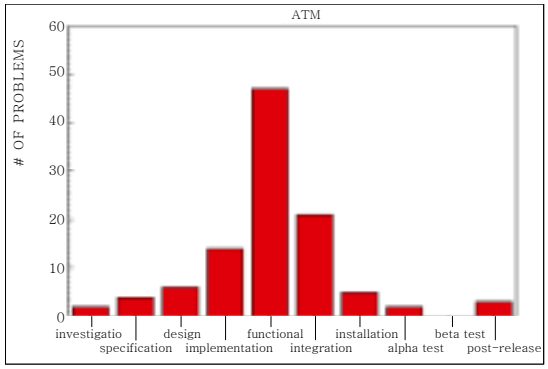


(그림 1) 각 기능별 문제점(원인 vs 수정) 현황

(그림 1)의 예를 보면 시스템 개발 초기단계인 소프트웨어 구현, 기능시험 및 종합시험에 많은 문제점이 발생되고 있음을 알 수 있다. 특히, 고장발생의 원인분석 결과 구현단계에 많은 오류가 발견되고 있는 점을 볼 때 개발일정에 문제가 있는 것으로 판단된다. 시스템 설계단계에서 이에 대한 충분한 고려가 되지



(그림 2) 각 기능별 문제점(원인 vs 발견) 현황



(그림 3) ATM Project 전체 문제점 발견 현황

못한 것으로 추정된다. 문제점이 수정(해결)되는 현황은 기능시험에서 가장 많이 처리되고 있는 점으로 보아 구현된 소프트웨어에 대한 기능 확인이 불충분한 상태임을 그림을 통해 알 수 있다.

또 (그림 2)에서 알 수 있는 것은 고장원인이 주로 기능구현 상에 존재하며 발견되는 시점은 기능시험 시 주로 발견되어 개발자 차원의 기능검증이 불충분하다는 것을 시사한다.

2. 형상변경 관리

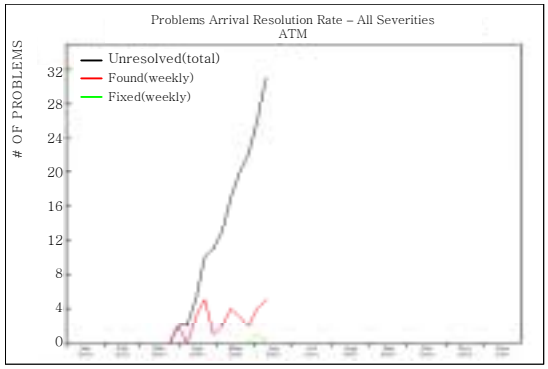
형상변경 관리활동은 형상인식활동, 형상변경 제어활동, 형상상태 기록, 보고활동 및 형상감시활동으로 나눌 수 있다.

형상변경 관리활동의 특징은 이미 개발 완료된 제품들에 대한 변경을 엄격히 통제함으로써 기존 제품들의 체계적인 관리와 품질향상을 가져오는 데 있다.

형상변경단계에서의 형상인식활동이란 Revision 규칙에 따라 새로 추가되거나 변경, 삭제되는 형상들을 정리하는 일이며, 형상변경 제어활동은 변경요구(Change Request: CR)에 대하여 그 흐름을 제어함으로써 변경요구에 대한 진행상황 및 변경요구의 내역을 관리하는 데 있다.

ATM 교환시스템의 각 기능 범주별 ATM Project의 전체 문제점 발견 현황은 (그림 3)과 같으며, 문제점 발견(Found) 및 해결(Resolve), 수정(Fixed)에 대한 전체 진행현황은 (그림 4)와 같다.

그 밖의 형상상태의 기록, 보고활동은 형상형성단



(그림 4) ATM 전체 문제점 발견/해결/수정 현황

계의 활동과 동일하며 형상감시활동은 수시로 제품 데이터 베이스에 들어 있는 내용물이 각종 형상정보와 일치하는가를 확인하는 활동이다.

교환소프트웨어 변경관리기능은 개발자가 디버깅을 위해 혹은 추가기능 요구사항의 구현을 위해 필요한 여러 정보를 편리하게 작성할 수 있도록 도와주는 제반기능과 이를 일괄적으로 등록하고 추적할 수 있는 기능들을 포함한다. 여기에 포함되는 기능들은 다음과 같다.

첫째, Unix 용 ClearDDTS를 이용한 기능으로

- Command-line interface, X-windows interface, Web interface 기능
- 사용자 인증기능
- 변경요구의 상태 정보 처리기능
- 변경요구에 대한 단위기간 별 각종 통계 제공기능

- 변경요구 사항의 DB화를 지원하는 정합기능

등이 있으며, 둘째 웹 기반 변경요구의 등록 및 조회 처리기능으로서

- 웹을 통한 변경요구의 등록
- 웹을 통한 변경정보의 내용 및 통계 조회
- 사용자의 인증 및 액세스 제한기능

등을 제공하여 시스템 개발단계의 모든 변경이력을 관리하는 데 이용된다.

아래에 제시한 통계데이터는 ATM Project에서 관리하고 있는 문제점 중 발견방법, 해결방법 및 문제점의 심각성에 대해 분석한 결과이다. 이 결과를 살펴보면 문제점 발견은 대체로 기능시험 과정에서 절반이 걸러지며 시스템시험과 사용자 추가 요구사항에 의한 기능 수정이 40% 정도를 차지함으로써 하위 레벨의 검증보다는 상위 레벨에서의 기능검증이 활발히 이루어지고 있음을 알 수 있었다. 이점은 개발 초기의 세부 설계단계에 대한 고려(Review)가 절실히 요구됨을 의미한다.

다시 말해서 초기단계에 수행하여야 할 품질개선 활동이나 툴에 의한 기능검증이 좀더 요구된다. 이러한 점이 추후 시스템 개발 시 고려하여야 할 개발환경상의 문제로 높은 신뢰성을 갖는 교환시스템 개발환경구축의 풀어야 할 숙제이다.

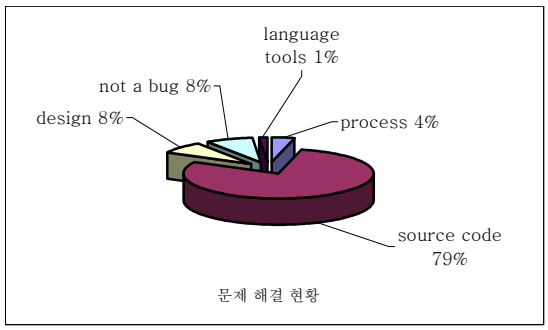
ATM Project 문제점 분석 통계 데이터 예

[Found Statistics]

functional test	49.07%
system test	20.37%
customer use	19.44%
group code review	0.93%
author code review	5.56%
in-house normal use	0.93%
interactive test	0.93%
manual review	1.85%
random unplanned test	0.93%

해결 방법상의 문제점들은 주로 소스 코드의 에러로 인한 문제 발생이며, 버그가 아닌 8%의 문제점은 교환기능 중 소프트웨어 버전관리기능을 추가하기 위한 변경사항으로 이 점은 기능추가로 별도로 분리하고 있다.

[Resolved Statistics]



문제 해결 현황

아래에 제시된 Severity 별 분포 내역은 검출된 문제점의 심각성에 따라 분류된 것이다. 즉, 운용상에 지장이 없는 사소한 문제점부터 Critical한 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 조치시간이 긴 심각성을 띤 문제점은 5% 정도인 것으로 분석되었다.

[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%

가. 소프트웨어 형상관리 도구

ATM 소프트웨어 개발은 다수의 소프트웨어 컴포넌트라는 개념을 도입하여 병렬개발의 장점을 십

분 이용하여 개발함으로써 개발기간을 줄이고 소프트웨어 소스코드의 품질을 높이고 있으나 새로운 기능의 추가나 변경에 따른 변경관리 도구와 버전의 관리와 개발의 전과정을 분석하고 예측하는 소프트웨어 개발공정관리 도구가 미비하여 소프트웨어 종합과 검증, 그리고 개발관리에 한계를 보이고 있다. 이를 극복하기 위해 소프트웨어 빌더, 작업공간관리, 버전제어를 위한 소프트웨어 형상관리 도구가 필요하다. 이러한 종합적인 도구를 이용할 경우, 소스와 변경관리, 그리고 개발과정의 공정관리에 대한 기술혁신을 통하여 최적의 개발공정을 찾을 수 있고, 또한 사용자의 다양한 요구를 수용할 수 있는 효과를 가져올 수 있다.

위의 요구사항을 수용하는 대형 교환 시스템인 ACE2000 소프트웨어의 형상관리 시스템은 버전관리를 안정화되고 다양한 방법으로 실현할 수 있는 기본적인 도구를 필요로 한다. 이러한 점을 고려하여 본 형상관리 시스템은 병렬개발을 지원하는 버전관리 도구인 ClearCASE를 기본으로 이용한다.

버전관리 도구로 널리 이용되는 ClearCASE는

- 지금까지 많이 사용되던 버전관리도구인 SCCS, RCS, CVS 등과 호환이 되며
- 병렬개발을 지원하며
- 결함관리 시스템인 ClearDDTS(Defects Data Tracking System)와의 연동기능을 제공한다.

개발 프로젝트의 독립된 개발과 동시적으로 시험 및 버그픽싱(bug fixing) 작업을 가능하게 하여 형상관리와 개발작업의 효율을 신장시킬 수 있다. 위의 기능 이외에 형상관리 도구에서 제공되는 기능은 형상관리와 변경관리 통합기능, 도큐먼트 관리기능, 기존 패키지 제작도구와 호환기능 모듈 제공, 자동 백업기능, 소프트웨어 품질 측정을 위한 원시 데이터 수집기능 등을 제공한다. 이와 같이 형상관리 도구들은 독립적인 도구로서의 가치뿐만 아니라 한 도구의 출력이 다른 도구의 입력으로 사용됨으로써 전체도구의 자동화를 통한 지원업무를 수행한다.

나. 품질표준 설정 및 평가관리

설계 표준에는 신뢰성 설계, 운용 보전성 설계, 하드웨어 및 소프트웨어 설계 표준들이 있으며 이러한 설계 표준들은 시스템 설계, 테스트 및 유지보수 상에서 발생하는 모든 활동들을 추적, 분석하는 데 용이하며 시스템 변경이 쉽고 시스템 신뢰성 및 유지보수성을 향상시키는 데 그 목적이 있다.

1) 신뢰도 배분 및 분석

ATM 교환기 개발에 있어서 제1의 품질목표는 신뢰성이며 이러한 신뢰성 목표 달성을 위하여 신뢰도 배분 및 분석기법을 적용하여 시스템을 설계하는 데 유용한 정보를 제공하고 신뢰도 설계의 방향을 제시함으로써 근본적이고 직접적인 품질 향상을 도모한다.

2) 부품관리

규격화된 부품을 사용하지 않을 경우 야기되는 시스템의 성능 및 신뢰도 저하, 개발완료 후 상당량의 유지보수 비용 등을 감안해 볼 때 부품관리의 필요성은 절실하다. 이에 ATM 교환기 시스템 개발에서는 시스템의 요구된 품질 및 신뢰도를 확보하고 보증하기 위하여 부품의 규격서 및 인증부품 목록 등을 작성하여 부품관리 업무에 주력한다. 즉, 통합부품관리시스템(Component Operation Management System: COMS)의 도입과 네트워크를 통한 부품발주, 재고상황 등을 한눈에 파악하여 부품수급과 재고자산을 효율적으로 관리하여 핵심부품의 적기 공급 및 개발일정의 지연 요소를 배제하여 인력 및 경비의 절감뿐만 아니라 시스템의 개발기간을 단축하도록 한다.

3) 계량 계측기기 관리

개발 제품의 특성치를 정확하게 측정하기 위해서 ATM 교환기 개발에 사용되는 모든 계량 계측기기의 정확도와 정밀도를 유지하도록 관리하는 것은 품질보증 측면에서 중요하다.

4) 시험 및 평가

운용중에 발생 가능한 에러들을 조기에 발견하여 상당한 양의 운용 유지비를 줄일 수 있을 뿐 아니라 보다 신뢰성 있는 제품 생성을 위해서 철저한 시험활동이 요구되어 진다. ATM 교환기 시스템에서는 기본시험, 기능시험, 시스템시험으로 구분하여 실시되어졌으며, 사전에 시험에 대한 치밀한 계획과 준비 작업으로 시험에 임하고 또한 시험실시 후의 평가 자료들이 피드 백(Feed-Back) 되어 수정 보완 시의 유용한 자료로 사용한다.

III. 소프트웨어 배포시기 결정법

1. 평가척도에 의한 배포시기 결정 방법

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

- 기대 잔존 에러 수

$$n(t) = a - H(t) \quad (1)$$

a: 시험 전 소프트웨어 내 잔존 에러 수

- 소프트웨어 신뢰도[R(x|t)]

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

- 평균 고장시간 간격(MTBF)

MTBF: Mean Time Between Failure

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

(1)과 (3)에 대해 t 마다 시험을 종료할 때 기대 잔존 에러 수 및 평균 고장시간 간격을 고려해 볼 때 그 때마다 각각의 목표치를 n_0, M_0라고 하면 (2)는 시험시각 t 일 때 시험을 종료하고 이때 운용시간 x에 대한 소프트웨어 신뢰도를 R(x|t)라고 하면 시험의 진척상황에 따라 계속 소프트웨어 내의 잔존 에러 수는 감소하는 경향을 띤다. 즉, 소프트웨어 신뢰도와 평균 고장시간 간격은 증가하게 되는 것을 의미한다. 이때 소프트웨어 최적 배포시기 T* 는 각각

$$\min[t \mid n(t) \le n_0],$$
$$\min[t \mid R(x \mid t) \ge R_0],$$
$$\min[t \mid MTBF(t) \ge M_0]$$

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

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

를 만족할 때 T = T1을 최적 배포시기라 한다. 구체적으로 말하면 NHPP에 의거한 지수형 신뢰도 성장모델(Exponential Software Reliability Growth Model: ESRGM)에 대한 최적 배포시기는

$$H(t) \cong m(t) = a(1 - e^{-bt}), (a, b > 0)$$

로 표현되고 이때 위의 식들은 아래의 식

$$T_1 = \frac{1}{b} \ln[\frac{a}{n_0}],$$
$$T_1 = \frac{1}{b} \ln[\frac{-m(x)}{\ln R_0}],$$
$$T_1 = \frac{1}{b} \ln[abM_0]$$

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

$$n(0) \ge n_0,$$
$$R(x \mid 0) \le R_0,$$
$$MTBF(0) < M_0$$

2. 개발비용 평가기준에 의한 최적 배포방법

소프트웨어 신뢰도 성장 이론에 근거하여 관측된 고장데이터에 대한 해석은 실제 문제로서 매우 흥미 있는 일이다. 즉, 소프트웨어 최적 배포시기를 정하는 데 있어서 상반된 요인에 대한 고려를 할 수 있는데, 소프트웨어 내에 잠재하고 있는 에러 수로 평가된 소프트웨어 신뢰도와 에러를 발견, 수정하는 데 시험에 투입된 공수(工數)나 유지보수 비용과의 관계이다. 다시 말해서 신뢰도 목표치를 정하고 달성된 신뢰도와 관련된 비용의 Trade-off를 그림으로 나타내는 방법이다.

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.

일반적으로 운용단계에서 발견되는 에러 당 수정에 필요한 비용은 시험단계보다 높은데 시험단계로부터 운용단계로 소프트웨어를 이행하는 과정의 최적 배포시기를 구하는 Trade-off 문제가 존재하게 된다. 그러면 소프트웨어 최적 배포시기를 결정하기 위한 최적 배포문제를 구하기 위해서 수식화 해보면 시험시간이 T일 때 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 에러 수를 의미하는 강도함수가 된다.

즉, T = 0일 때

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

이 되어 h(T)에 대한 유한 값의 해를 구하면

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

가 된다. 결론적으로 시험시간 T에 대해 최적 배포시기를 정리하면

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

가 된다. 즉, 최적 배포시기 T*는

$$T^* = \min[T_0, T_{LC}]$$

로 정리된다. 위와 같이 소프트웨어 개발비용에 대한 연구는 여러 차례 제기되었으며, 개발비용을 평가기준으로 삼아 소프트웨어의 최적 배포시기를 연구한 예도 많이 있다[4]-[10].

3. 신뢰도와 평가기준을 동시에 고려한 최적 배포

소프트웨어 신뢰도와 평가기준을 동시에 고려한 최적 배포시기 결정법은 신뢰성 평가척도와 소프트웨어 비용을 가지고 달성된 신뢰도에 대한 비용을 평가해 보는 방법으로서 투입된 개발비용의 유효성(cost-effectiveness)을 파악해 보는 매우 흥미 있는 일이다. 즉, 소프트웨어 개발비용과 신뢰도라는 2개의 평가기준을 고려하여 최적 배포시기를 구하

는 것이다.

NHPP 모델에 대한 평균치 함수 H(t)와 시험을 진행시켜 달성된 신뢰도 목표치 R_0를 달성할 때 (6)의 총 기대 소프트웨어 개발비용 C(T)를 최소로 하는 경우의 총 시험시간을 정하는 것이다. 즉, 운용시간 x(x > 0)에 대해 정리하면

$$minimize C(T),$$
$$R(x|T) \geq R_0 (T \geq 0)$$

를 만족하는 시험시간 T를 구하면 된다.

지수형 신뢰도 성장모델의 평균치 함수를 m(t)라고 하면

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

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

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

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

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

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

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

$$h(0) \leq \frac{C_3}{C_2 - C_1}, R(x|0) < R_0$$
$$T^* = T_1 \quad (6)$$

$$h(0) \leq \frac{C_3}{C_2 - C_1}, R(x|0) \geq R_0$$
$$T^* = 0 \quad (7)$$

가 된다.

4. 신뢰도와 개발비용 평가기준을 동시에 고려한 실제 데이터 적용 결과

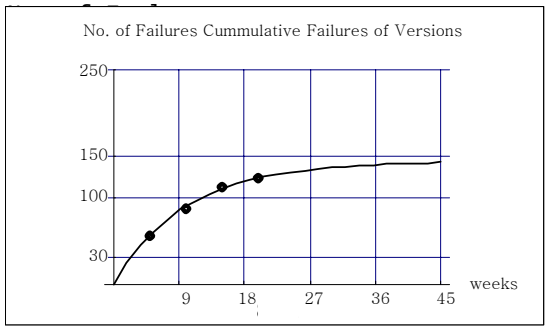
지수형 신뢰도 성장모델에 근거한 58만 라인 규모의 대형 ATM 교환기의 소프트웨어 개발 시 관측된 고장 발견데이터를 적용하면 (그림 5)와 같다.

교환 소프트웨어의 구현 언어는 C, C++ 및 Java로 구현되었으며, 교환기의 핵심기능인 호 제어기능 및 GSMP(General Switch Management Protocol) 처리기능 등 프로토콜처리 관련 소프트웨어는 교환기 본체에 그리고 시스템의 형상정보, 유지보수기능 및 시스템 상태에 대한 주기적 감시기능은 워크스테이션에 상주하여 기능을 크게 이원화하고 시스템의 성능을 향상시킨 분산제어 구조로 구성하였다.

교환기 본체와 워크스테이션간의 통신은 미들웨어(Middleware)를 통해 처리하고 이더넷을 통해 UDP(Uni-Direction Protocol) 통신을 수행한다.

◆ 시험 방법 및 고장데이터 수집

고장데이터를 수집하기 위한 시험은 크게 기능담당자가 구현된 기능의 정상동작여부를 확인하는 기능시험과 소프트웨어 종합자가 각 기능들을 하나의 시험용 소프트웨어 패키지로 구성하여 수행하는 종합시험(Integration test) 그리고 품질보증을 위한 시스템 차원의 시스템시험으로 구분되어 수행된다. (그림 5)는 주별 시스템 시험 시 검출된 초기 고장데이터를 토대로 평가한 축적 고장데이터이다(그림에서 점은 5주 단위로 수집된 실측치를 의미함).



(그림 5) 주별 초기 고장데이터 추정치

위의 검출된 고장데이터를 가지고 지수형 신뢰도 성장모델에 근거하여 추정한 초기 결함 수(a)와 고장검출률(b)는 각각

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

로 추정되었다. 이 데이터를 평균치 함수 m(t)에 적용하면

$$\hat{m}(t) = 143.757(1 - e^{-0.00992314t})$$

이 되며, 이때 신뢰도 목표치를 R_0 = 0.9로 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

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

가 된다. T_{LC} = 50(주)인 경우

C_1 = 1.0, C_2 = 5.0, C_3 = 50(/주) 일 때

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

가 된다. 즉, 배포시기는

T* = min[T_0 , T_{LC}] = 1.334(주)가 된다.

다시 말해 소프트웨어 배포 후 운용시간은

T_{LC} − T = 50 − 1.334 = 48.67(주)가 되며, 여기서 총 기대되는 소프트웨어의 비용 최적치는

C(T*) = 2576.818가 된다. 이때 T* = 1.334를 이용하는 경우 소프트웨어 신뢰도는

$R(0.1 \mid T^*) = 0.0125$ 로 되어 이는 초기 신뢰도(R_0 = 0.024) 보다 낮게 되므로 이때의 최적 배포시기는 신뢰도와 개발비용을 동시에 고려한 형태로 되어야 한다. 이 경우에는 아래의 식

$$T^* = \max[T_0, T_1],$$
$$T_1 = \frac{1}{b} \ln[\frac{a}{n_0}],$$
$$T_1 = \frac{1}{b} \ln[-\frac{m(x)}{\ln R_0}],$$
$$T_1 = \frac{1}{b} \ln[abM_0]$$

를 얻을 수 있다. 이 조건을 만족하는 T_1을 구하면 T_1 = 26.263이 된다.

즉 시험시작 후 27주 정도 시험 후 배포하는 경우가 비용과 신뢰도를 고려한 경우의 최적 배포시기임을 의미한다. 이 시기는 대략 시험시작일(2001.3.) 기준 약 7개월 후인 2001년 10월 중순경이 예상된다. 그러나 이 기간은 시험자원의 투입량에 따라 달라질 수 있으며, 시스템의 신뢰도에 따라 적절히 시험자원의 투입량을 조절할 수 있는 프로젝트 관리 기법이 필요하다.

또 최적 비용을 산출하면 C(T*) = 1494.55가 된다. 이것은 소프트웨어의 총 기대비용이 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.

Ⅳ. 결론 및 향후 연구방향

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

본 논문에서는 소프트웨어 평가기준에 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 연구방향은 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, 1989, pp. 195 – 206.

[2] Rational, "ClearCASE Reference Manual: Release 4.0," 2001, pp. E1 – E19.

[3] Rational, "ClearDDTS Administrator's Guide: V4.1," 2001, pp. 1 – 22.

[4] 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, Aug. 1979, pp. 250 – 253.

[5] H.S. Koch and P. Kubat, "Optimal Release Time of Computer Software," *IEEE Trans. Software Eng.*, Vol. SE-9, No. 3, May 1983, pp. 323 – 327.

[6] S. Yamada and S. Osaki, "Optimal Software Release Policies with Simultaneous Cost and Reliability Criteria," *European J. Operational Research*, Vol. 31, No. 1, 1987, pp. 46 – 51.

[7] R.W. Wolverton, "The Cost of Developing Large-scale Software," *IEEE Trans. Computers*, Vol. C-23, No. 6, Jun. 1974, pp. 615 – 636.

[8] 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, 1980, pp. 315 – 318.

[9] 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, Japan, 1988. 7., pp. 1338 – 1340.

[10] S. Yamada and S. Osaki, "Cost Reliability Optimal Release Polices for Software System," *IEEE Trans. Reliability*, Vol. R-34, No. 5, Dec. 1985, pp. 422 – 424.