소프트웨어 품질관리와 변경제어 기법을 통한 소프트웨어 형상관리

Software Quality Management and Software Configuration Management Base on Change Request Control Method

이재기(J.K. Lee) 시스템종합팀 선임연구원
신상권(S.K. Shin) 시스템종합팀 기술기능원
남상식(S.S. Nam) 시스템종합팀 책임연구원, 팀장
박권철(K.C. Park) 교환기술연구부 책임연구원, 부장

소프트웨어는 개발과정에서 빈번한 변경이 발생한다. 특히, 대형 시스템 개발에 있어서 소프트웨어의 변경요구는 다양하고 복잡하게 된다. 이러한 변경요구에 따른 소프트웨어의 품질관리 또한 중요한 이슈가 되며 변경요구는 개발조직과 연계하여 자연스럽게 제어되어야 한다. 효과적으로 형상관리를 완벽하게 처리하면서 최종 시스템 소프트웨어로 수용함으로써 적시에 배포할 수 있는 기법은 프로젝트 관리 측면에서도 매우 중요하다. 본 고에서는 개발중인 시스템의 소프트웨어의 변경요구를 개발조직의 관리 특성에 맞추어 시스템의 형상관리에 완벽을 가하고 요구사항에 대한 수용기간을 단축시키며 체계적으로 관리, 제어하는 기법과 이에 따른 품질관리 방안 등을 소개하고 이를 지원하는 S/W 종합 도구 및 운용사례 등을 밝힌다.

I. 서 론

대형 소프트웨어 시스템을 개발하는 데 있어서 최종 시스템이 완성되기까지 막대한 양의 소프트웨어가 생산되며, 생산된 소프트웨어에 대한 변경요구가 끊임없이 요구된다. 즉, 지속적으로 변화하는 소프트웨어들에 대한 형상관리와 시스템 신뢰도 측면의 품질관리가 요구된다.

최근의 통신 소프트웨어의 경우는 사용자의 요구사항이 매우 다양해지고 급변하는 추세로 인해 형상에 관한 완벽한 관리가 필요하게 되었다. 이러한 요구사항에 따라 효율적인 관리를 통해 개발기간의 단축 또는 사용자가 원하는 최적 배포가 요구된다. 위와 같은 요구사항을 만족하기 위해서는 좋은 개발환경과 개발 방법론을 갖추었다 하더라도 각 단계별로 체계적으로 관리되지 못하면 프로젝트를 성공적으로 수행하지 못한다.

본 논문은 제II장에서 소프트웨어 품질 관리시 고려되어야 할 사항과 에러의 검출, 분석 및 발생 방지책, 종합적 품질관리에 대해 논하고 III장에서는 소프트웨어 변경제어 기법을 이용한 소프트웨어 관리 방법을, IV장에서는 본 논문의 핵심인 형상관리 방법과 소프트웨어 종합환경에 대해 소개하고 결론을 맺는다.

II. 소프트웨어 품질관리

과거의 소프트웨어 품질은 메모리를 적게 차지하

고 실행시간이 짧은 소프트웨어를 요구했지만 현재는 개발과 유지보수 양면을 고려한 프로그램 작성이 요구된다. 즉, 프로그램의 품질 평가가 종래와는 다르게 에러가 적은 신뢰성, 사후 유지보수가 편리한 유지보수성, 기능의 확장이 용이한 확장성을 고려한 소프트웨어 품질로 평가된다.

개발중인 소프트웨어 품질을 잘 관리하기 위해서는 개발 도중에 발생되는 에러를 발견 즉시 수정하여야만 고품질의 소프트웨어를 획득할 수 있다. 이와 같이 개발 단계별로 발생되는 에러에 대한 검출 방법 및 분석, 방지책에 대한 내용을 살펴보면 다음과 같다.

1. 에러의 발생 분석

에러가 발생하는 경우에 분석을 위해 고려하여야 할 사항은 아래와 같다.

- 발생시점(calendar time)
- 발생부분(location)
- 발생빈도(ratio)
- 발생원인(cause)

이 있으며, 에러의 발생비율에 대한 데이터는 TRW의 조사분석 결과 설계와 코딩이 각각 64%, 36%를 차지하는 것으로 조사되었으며, MITRE-report는 50%씩 분포하는 것으로 이미 연구 발표된 바 있다[1]. 또한 에러의 발생원인에 대한 분석은 AM. Pietrasenta가 여러 프로젝트에 관해 논문을 발표한 바 있으며, 이 연구 논문에서 설계, 코딩, 수정 착오에 대해 발생원인을 잘 설명하고 있다.

- 설계(design)
  ① 설계언어의 부족
  ② 표현의 애매(미비)
  ③ 각 설계 단계의 변환오류 및 정보 전달 에러
  ④ 설계 결함
  ⑤ 문서화 미비
  ⑥ 방법론 결여(각 모듈의 인터페이스 기준)
  ⑦ 교육(training)의 불충분
- 코딩(coding)
  ① 테스트의 의존도
  ② 테스트 대상의 부적절
  ③ 변경요인의 과다 발생
  ④ 코드 점검 미비(code complete)
  ⑤ 표준화 부족(coding standard)
  ⑥ 교육
- 수정착오(correction error)
  ① 설계착오 수정의 어려움
  ② 수정에 대한 검토 미흡
  ③ 수정에 대한 테스트 부족
  ④ 파급효과(side-effect)

2. 에러의 방지

소프트웨어 작성과정의 불명확에 의해 발생되는 원인은 주로 필요한 기준의 설정 불충분, 도구의 미비, 변경의 과다, control의 결여 등이며 에러를 줄이기 위한 방법은 작성 과정을 명확히 하여 보다 좋은 도구를 이용하고 변경을 적게 하여 제어를 잘하도록 하는 것이 가장 좋은 방법이다.

3. 에러의 검출 방법

에러의 검출 방법으로는 두 가지 방법이 있는데

① 품질 좋은 프로그램을 가능한 한 싼 비용으로 개발하는 방법은 에러를 적은 비용을 들여 검출하는 방법을 모색하는 것이 첫번째요
② 효과적인 에러 검출 원칙은 에러 발생 직후에 에러를 발견하는 방법이다.

그밖에 1985년 미국에서 발생한 TQM(Total Quality Management)을 도입하여 프로젝트에 참여하고 있는 조직원 전체가 참여하는 조직적인 품질관리가 되어야 한다. TQM은 품질을 관리하고 보증하는 일 뿐만 아니라 품질을 계획하고 조직하고 감독, 통제하는 기능을 포함한다[2, 3]. 특히, 고객 중심의 품질관리와 지속적인 품질개선이 이루어져야 하며, 이에 대한 방법을 약술하면 아래와 같다.

- 고객중심의 관리
- 조직적 참여와 팀웍
- 지속적인 품질개선

등이다. 기타 품질향상 기법으로는 현재 많은 기업에서 적용하고 있는 관리 기법으로 1987년 모토롤라(Motolora)에서 출발한 Six-sigma가 있는데, 이는 통계적 척도와 주어진 환경에서 인지하고 일하는 방법으로 “Working harder”가 아닌 “Working smarter”를 의미하는 품질에 대한 경영전략으로 각광 받고 있으며, Asia brown boveri(93), Allied signal(94), GE(94)에 성공적으로 적용되는 등 많은 기업들을 중심으로 성공적으로 채택되었다[4]. 또 노키아(Nokia), 소니(Sony), 록히드, 폴라로이드(Polaloid) 등에서도 채택하는 등 아시아 및 유럽 전역에서 이 기법들을 도입하고 있다. 이 기법은 통계적 기법을 도입한 데이터에 의한 관리와 교육, 훈련을 통한 품질 혁신이다.

III. 소프트웨어 변경제어 기법

개발과정에 있는 시스템에서의 소프트웨어 변경요구는 정의된 요구사항에 대한 변경과 시스템으로의 통합 단계에서 수행되는 시험의 결과로 인해 발생된다.

이미 정의된 시스템 요구사항에 변동이 발생하면 차기 버전에 적용되고 또 개발주기의 시스템 시험단계에서 문제점들에 대한 소프트웨어 변경요구로 나타낸다.

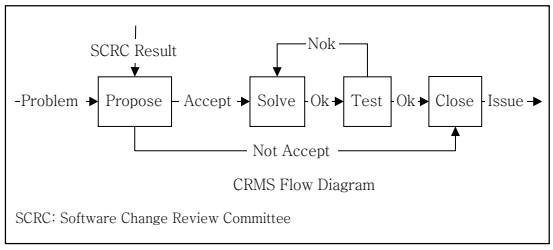
시험단계에서 발생하는 소프트웨어 변경요구는 수요가 많고 복잡, 다양하며 요구사항을 수용하기 위한 효과적인 제어체계가 준비되지 않으면 시스템의 개발기간이 길어질 수 밖에 없다. 이러한 결과는 소프트웨어 형상관리에도 영향을 미쳐 시스템의 품질보증은 물론 신뢰도도 보장할 수 없게 된다. 즉, 다수의 복잡 다양한 소프트웨어 변경요구를 효율적으로 제어하기 위한 체제가 구축되어야 한다. 이러한 체제를 구축하기 위한 요구조건을 살펴보면 아래와 같다.

- 간단 명료한 상태제어 구조
- 개발 및 관리조직과의 원활한 연계
- 변경요구의 신속한 수용
- 정확한 소프트웨어 형상관리

등이다. 이러한 요건을 충족시키기 위한 변경제어 기법을 기술하면 III장 1절과 같다[5].

1. 상태제어 구조

소프트웨어 변경요구가 발생하여 시스템의 패키지로 수용되는 절차는 (그림 1)과 같다.

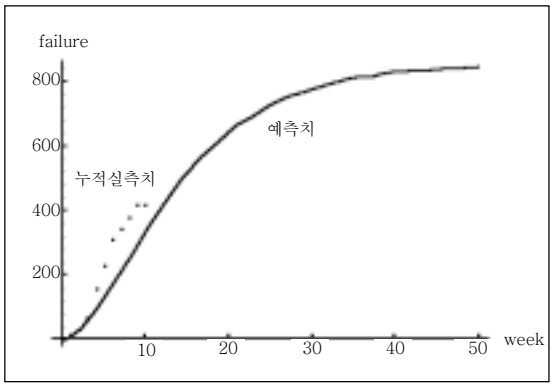


(그림 1) CRMS의 상태 천이도

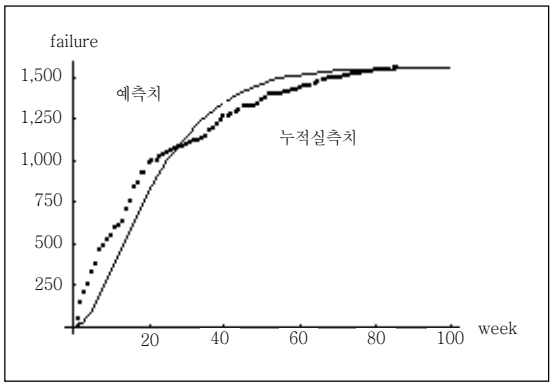
하나의 소프트웨어 변경요구가 수용되어 종결될 때까지의 단계가 많은 경우는 소프트웨어 형상변경에 신중함을 기하고 기존 시스템 소프트웨어에 대한 품질보증 및 신뢰도를 확보할 수 있게 한다. 이와 같은 방법은 소프트웨어 배포 이후에 발생하는 문제점으로 인한 소프트웨어 변경요구를 관리하기 용이한 반면에 단계가 많아지고 복잡해지는 관계로 변경요구의 수용에 다소 지연이 생기며, 개발에 필요한 문서 외에 각 개발단계별로 파생되는 문서가 많아져 또 다른 문서관리가 필요하게 되는 단점이 존재한다.

개발단계의 소프트웨어 변경관리는 개발자와 관리자 모두 쉽게 인식할 수 있는 체제로 시스템의 신뢰도와 소프트웨어 품질보증의 확보 차원에서 순수 개발관리 이외의 부가적인 노력을 절감할 수 있는 체계가 필요하다.

(그림 1)의 제어 구조에 따라 소프트웨어에 대한



(그림 2) 누적 변경요구 처리 현황(ATM)



(그림 3) 누적 변경요구 처리 현황(TDX)

변경요구가 받아지고 종결될 때까지의 상태관리 및 상태변경은 개발 및 관리조직의 연계로 이루어지며, SCRC(소프트웨어 변경 검토 위원회)에서는 P 상태의 제기된 문제점들을 review하고 시험 결과에 대한 검토를 거쳐 확정하여 그 결과를 차기 패키지에 반영한다.

S 상태에서는 개발 담당자가 P 상태에서 제기된 문제를 분석, 해결하면 자체 시험을 거쳐 소프트웨어 Source 등록을 요구하며, 이때 CR 상태를 시험중(T) 상태로 변경, 시험을 요구하게 된다.

시험상태의 CR들은 시험자에 의해 종합 검증을 거친 후 SCRC의 결과에 따라 종결(I 상태) 또는 해결중(S 상태)으로 천이된다. 이에 대한 상세 내용은 III장 2절에 기술한다.

CRMS(Change Request Management System: 변경이력관리 시스템)으로 관리되고 있는 TDX 및 ATM 시스템의 소프트웨어 변경요구에 대한 처리 현황을 살펴보면 (그림 2) 및 (그림 3)과 같다. 그림에서 프로젝트가 진행되는 동안 신뢰도 목표치를 정한 후 예측한 데이터와 실제 변경을 가한 상태의 데이터로 개발 초기에는 예측한 데이터보다 많은 변경이 이루어지고 있음을 알 수 있다. 이와 같은 이유는 개발 초기의 시스템 개발에 유연한 대처를 위해 많은 부분이 수정되었다고 말할 수 있다. 한편으로는 초기 설계시 많은 변경으로 인해 개발기간이나 유지보수에 많은 문제점을 안고 있는 것으로도 해석이 가능하며, 이러한 점(시스템 설계 기술)은 차기 프로젝트 수행시 개선해야 할 과제이다.

(그림 2)에서 x축은 기간을 나타내며, 그 단위는 주(week)이고 y축의 단위는 변경요구 횟수를 의미한다.

2. 개발과 관리조직과의 연계 활동

소프트웨어 변경요구 수용을 위한 개발 및 관리조직의 형태는 문제해결을 위한 주기적으로 변경요구상황을 점검, 검토하는 개발관리조직과 문제해결을 주도하는 구성원인 개발조직으로 형성된다. 개발관리조직은 회의체로서 변경수용의 전반적인 상황점검과 내용 검토, 진도 및 결과 판정, 전체 개발일정 등을 조정한다. 조직 내의 각 구성원의 역할은 다음과 같다.

- 관리자(Manager):

변경요구 검토회의 주관 및 해결자 선정, 시험결과 검토, 회의자료 정리 및 통보, 문제 해결 확인

- 개발자(Developer):

변경요구 해결 및 소프트웨어 등록, 문제점 분석, 변경요구 제안 및 변경문서 작성 등

- 종합자 및 시험자(Integrator & Tester):

소프트웨어 형상 확인 및 패키지 제작, 시험 수행 및 문제점 도출, 시험 결과 정리 및 통보

등으로 각 조직 내에서 허용된 범위의 상태 변환 권한이 주어져 책임과 의무로 연계 관리, 운용되고 있다.

3. 변경기능 수용을 위한 지원도구

소프트웨어 변경 수용을 위한 도구는 CRMS로 UNIX 환경 하에서 C-언어 및 AWK로 구현되었으며, 그 기능은 크게 모든 변경이력을 저장, 변경, 검색 및 출력기능으로 분리 운용되고 있다.

변경기능은 단순한 내용변경과 상태변경기능으로 구분되며, 상태변경은 각각의 주어진 고유 권한에 의해서 접근변경이 허락되고 그 이력이 Log file(또는 History file)로 저장되어 추적 관리가 가능하다.

검색기능은 특정 변경요구에 대한 검색, 전체 상황에 대한 종합상황, 상태별 검색기능, 팀별 종합상황 등 개발관리에 필요한 제반 정보들을 제공하고 있다.

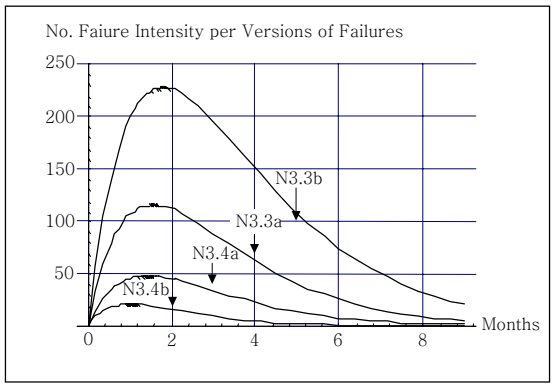
출력기능은 특정상태에 대한 출력과 검색기능과 연계된 전체 종합상황표 출력 등 다양하게 제공된다.

종합상황표에 대해 간단히 언급하면 우선 CR 번호와 발생일시, 버전정보, CR 제목, 해결 담당자, 문제유형, 상태(해결중이거나 처리 완료된 상태를 의미) 등을 현재 진행되고 있는 변경요구에 대해 전체적으로 파악할 수 있는 정보를 제공해 준다[6].

4. 등록/변경요구 관리도구 운용 결과

소프트웨어 등록/변경요구 처리를 위한 관리시스템인 CRMS tool은 TDX-10 ISDN 시스템의 소프트웨어 변경요구 관리부터 ATM 교환시스템인 HANB-ISDN(High-Advanced National Project Broadband ISDN)의 ACE64/256 시스템의 변경이력 관리까지 사용되고 있다. 각 시스템별로 구분되어 소프트웨어 버전별로 관리되고 있으며, 실제 도구에서 운용되고 있는 데이터는 TDX-10 ISDN 시스템이 1,900개 정도이고((그림 4) 참조), ATM 교환시스템은 ACE64 시스템의 소프트웨어 버전인 SV3.1부터 SV3.5까지 버전별로 250~300개 정도로 관리되고 있다.

대형 시스템인 ACE256 시스템은 180개 정도 관리되어 전체 도구가 관리하고 있는 총 등록/변경요구서는 3,500여 개 정도이다. 그밖에 ATM 시스템에



(그림 4) TDX-10 ISDN 시스템 버전별 등록/변경 요구 관리 현황

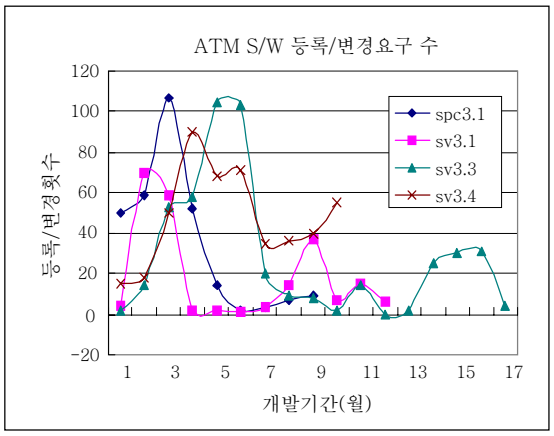
대한 Problem Note로 FM(Fault Management) tool에서 별도 관리되고 있는 문제점 개수는 2,000개 정도이다. 각 시스템별 CR 및 문제점 관리 현황을 살펴보면 <표 1>과 같다.

<표 1> 시스템별 CR 및 문제점 관리 현황

시스템명	관리 CR 개수	비 고
TDX-10 ISDN	1,980	시스템 문제점은 FM으로 별도관리
ACE64(SV3.5)	312	FM 관리 문제점 2,000개 (별도관리)
ACE256(SV5.1)	186	시스템 문제 별도관리

(그림 5)는 HANbit ACE64 ATM 교환시스템의 개발기간에 따른 소프트웨어 등록/변경요구 관리의 버전별(SPC3.1~SV3.4) 현황으로 PVC(Permanent Virtual Connection) 기능 위주의 1차 시제품부터 SVC(Switched Virtual Connection) 교환시스템인 SV3.4까지의 현황을 일목요연하게 나타내었다((그림 5)에서 SV3.2는 SV3.1에 포함됨: 제어계의 변경 및 하드웨어 재설계로 인해 개발기간이 단축되어 데이터가 적음).

시스템의 개발기간은 대략 버전별로 9개월에서 최장 17개월 정도 소요되었으며, 본격적인 시스템 개발은 '95. 12월부터 시작된 PVC 기능 위주의 SPC3.1(Semi-Permanent Connection version 3.1)을 시작으로 SVC 기능이 추가된 SV3.1 버전('96. 3월



(그림 5) ATM 시스템 버전별 등록/변경 관리 현황

개발 시작)부터 '97. 12월까지 SV3.4 버전에 국가망 1, 2차 기능(ITU-T/ATM-Fourm 권고 기능 채택)이 수용되었고 이후 SV3.5 버전에 PNNI(Private Network-Network Interface) 연동, SVP(Switched Virtual Path) 기능 등 추가 요구사항이 적용되었다.

또한 '99년 초부터 2000년 4월까지 단위 스위치의 용량이 10Gbps로 확장된 ACE256 교환기에 SV5.1 버전으로 개발되어 기 개발된 각종 기능들이 본 시스템에 이식되었다. 즉, ACE256 교환기에는 PNNI 연동기능이 강화되었고 CPU의 교체(Super SPARC에서 MPC750으로 변경) 및 컴파일러 변경, DBMS의 보완 뿐만 아니라 OS의 변경, 장치제어계 도입에 따른 mSROS(micro-Shell Real-time OS)의 추가, N-ISDN/PSTN 연동, STM-4c 수용, AAL2(ATM adaptation Layer 2) Trunking 기능, ADSL(Asymmetrical Digital Subscriber Line) 가입자 수용, VP 자원관리 기능 등이 이루어졌다.

IV. 소프트웨어 형상관리

1. 소프트웨어 형상체계 및 관리 규모

소프트웨어 생산의 기본이 되는 단위는 컴파일 및 보관의 업무를 처리하는 단위로 파일이 되며, 소프트웨어 블록은 한 개 이상 다수의 파일로 구성되어 있다. 이 실행모듈은 실행단위로 분류되고 각 프로세서에 올라가는 로딩 블록의 단위이기도 하다.

가. 공통파일(시스템, 사용자)

공통파일은 시스템과 사용자 공통파일로 구분되며, 이것들에 대한 현황은 주로 프로세서간 메시지 정의파일, Spec module 파일인 library, 프로세서 id 정의파일 등과 블록간 공통으로 사용되는 include file, 실행모듈별 Makefile 등 많은 파일이 존재하며, 이것들은 시스템 공통파일과 프로세서 공통파일, 블록을 구성하는 파일로 분류할 수 있다.

나. 중간파일, 목적파일

릴레이션의 dg 파일을 컴파일한 목적파일인 F_*, PLD.ucf, Xref.dil, Hdr.dil, dfsf, edml data 형태 규격파일과 각 블록에서 컴파일시 생성되는 *.o, *.ms 파일 등이 여기에 속한다.

다. 최종파일(PLD, MMSDATA, A.out, 기타 미공개 파일)

특정 Target 시스템에 맞추어 작성된 데이터 목적파일 및 입·출력 메시지에 대한 프로세서 실장 데이터, 각 블록별로 컴파일된 실행모듈이 존재하고 이밖에 공개하기 어려운 소스프로그램 파일들이 존재한다.

패키지 종합환경에서 체계적으로 관리되고 있는 TDX 및 ATM 소프트웨어의 규모를 살펴보면 <표 2>~<표 4>와 같다.

<표 2> 소프트웨어 블록수 및 규모(TDX 시스템)

기능범주	블록수	규모(KLOC)	기능수
CP	45	390.7	270
DH	14	171.9	33
Ad	44	315.3	290
OM	29	240.7	241
Kernel	8	217.5	-
Total	140	1336.1	834

<표 3> 팀별 소프트웨어 관리 현황(ATM)

버전별 / 규모별	SV3.5.3 (ACE64)	SV5.1.2 (ACE256)
S/W 규모	977,469	832,157
W/S 소스 규모	1,003,915	1,005,286
F/W 소스 규모	422,791	477,326
합 계	2,404,175	2,314,769

<표 4> 소프트웨어 소스 관리 규모(ATM)

형상 / 팀별	기능수		블록수	
	ACE64	ACE256	ACE64	ACE256
호 제어	165	91	37	42
운용	25	34	31	33
보전	64	56	40	46
TMN	70	69	12	12
총계	324	250	120	133

2. 소프트웨어 종합 절차

소프트웨어 종합 절차는 각 프로세서에 탑재될 실행모듈과 데이터를 생성하기 위한 환경을 구축하고 생성된 모듈들을 하나의 패키지로 제작, 시험하며, 계속적인 소프트웨어 컴포넌트(component)들을 관리, 유지하는 업무를 종합이라 부른다. 이에 대한 절차는 대략 패키지 제작 계획서 작성 및 공지, 등록/변경요구서 접수 및 검토, 공통파일 등록 및 제작/배포, 소스프로그램과 데이터파일 등록 및 제작, 시스템 load Tape(SLT) 제작(ATM에서는 해당 파일을 W/S으로 전송한 후 시스템에 로딩) 및 배포, 시험 순으로 진행된다[7].

최근에는 워크스테이션(W/S)의 성능 향상에 따른 소프트웨어 종합환경시스템과 시스템 제어용 운용터미널로 사용되고 있는 W/S간의 로딩 방법으로 SLT (System Load Tape) 제작이 필요 없는 실행파일 전송방법(File Transfer Protocol: FTP) 및 Target 시스템으로의 다운로딩이 가능하여 시스템의 시험환경 개선과 개발기간의 단축을 꾀하고 있다.

가. 패키지 등록 및 제작 절차

1) 패키지 제작 계획
2) 변경요구 접수 및 검토
3) 공통파일 제작 및 배포
4) 소스프로그램 등록 및 컴파일(compile)
5) Target system으로의 파일 전송 및 시험
6) SLT 제작 및 배포 등

나. 패키지 제작 계획

개발일정에 따라 패키지에 추가되어야 할 기능을 결정하고 기 발생된 문제점을 분석하여 이에 대한 보완시기 및 패키지 반영시기를 결정하는 일로 버전 구축 작업과 동시에 이루어지며, 제작 계획서에는 반드시 목적, 등록 일정, 등록 기능범위, 참고사항과 버전명이 포함되어야 한다.

다. 등록/변경요구 접수 및 검토

등록/변경요구서에 따른 보완내역을 검토하고 반영 대상이 되는 소프트웨어를 등록 파일에 명시하여야 한다. 변경/등록요구서는 각 상태별(제안/해결/시험/종결)로 관리되어야 하며, 프로젝트에 참여하고 있는 모든 사람에게 처리되고 있는 상황을 온라인으로 제공되도록 툴을 이용하여 시스템 개발업무가 원활하도록 지원한다.

라. 공통파일의 제작 및 배포

개발자(또는 사용자)가 등록하는 공통파일 소스는 제작도구를 통해 다른 소스프로그램 내에서 include될 수 있는 형태로 변경된다. 이 파일들을 공통파일이라 부르며, 여기에 속하는 파일들을 열거하면 데이터, 프로세스, 메시지, 입·출력 메시지 및 Shared library 파일, include file 등이 있다.

마. 소스프로그램 등록 및 컴파일

개발자가 등록한 소스프로그램은 실행모듈과 PLD(Program Loaded Data) 생성 과정으로 나뉘어 실행되며, 그 절차는 아래와 같다.

- 실행모듈 생성

생성된 공통파일을 이용하여 등록할 소스프로그램을 컴파일하고 기능 검증을 거친 후 등록도구(reg)를 통해 소프트웨어 종합관리 디렉토리로 등록되고 관리된다. 이 파일들은 C/CHILL-Compiler를 통해 각종 공통파일들과 함께 링크(link)되어 실행모듈로 생성된다.

• PLD 생성

개발자가 기 등록한 데이터 구조파일에 맞게 작성한 시험데이터는 툴에 의해 형상정보, 구성정보 및 릴레이션(relation), 프로세서 단위로 컴파일되어 제작된다.

바. 개발자와 종합자의 업무 언급

종합자는 사용자 공통파일 등록 시간에 개발자가 등록요구한 공통파일들이 등록되었는지 확인하고 개발자가 등록한 파일들을 등록변경요구 내역을 재검증한다. 사용자 공통파일 검증시 에러가 발생한 부분 및 미 등록분에 대한 내용을 파악, 패키지에 반영여부를 결정한다.

개발자는 변경요구서에 언급된 내역 중 사용자 공통파일을 패키지 제작일정(종합일정)에 맞추어 우선 등록하고 사전에 충분한 검증을 통한 소스파일은 나중에 준비된 시기에 등록한다. 이 이유는 사용자 공통파일은 컴파일을 통해 생성되는 최종 공통파일을 만들기 위한 소스파일로써 여러 블록에서 공동으로 사용되는 파일들로 그 종류와 제작 절차가 까다롭고 타 블록에도 영향을 미치므로 이에 대한 변경 및 보완은 신중을 기하여야 한다. 특히 사전에 연동이 필요한 블록 관계에게 공지하여 충분한 검토를 거친 후 변경, 보완하는 것이 필요하다. 대부분의 컴파일 시 발생되는 에러들이 공통파일의 변경으로 인한 인터페이스 에러이다.

사. 시험자와 종합자와의 관계

패키지 종합이 완료되고 시험용 소프트웨어 패키지가 제작되면 시험자는 해당 파일을 시스템에 로딩하여 기능의 검증 절차를 밟는다.

시험 과정에서 도출된 문제점 중 패키지 제작상의 문제는 종합자에 통보하여 보완 여부를 항시 확인하여야 하는 관계로 시험자와 종합자는 항시 검증될 패키지에 대한 정보의 공유가 이루어져야 하는 협조체계에 있어야 한다. 또한 패키지에 적용된 기능이나 제외된 기능에 대한 차기 패키지 적용여부, 시험범위 선정 등 소프트웨어 패키지에 대한 전반적인 의견교환이 있어야 한다.

아. 배포문서 작성 및 패키지 종합 정리

소프트웨어 배포는 소스프로그램, 실행모듈, 지원도구가 삼위일체가 되어야 하는 관계로 시험결과, 등록/변경요구서, 패키지 종합결과 등을 포함하여야 하며 배포문서에는 배포 목적과 일시, 배포 내역 및 배포처 등을 포함한다.

3. 소프트웨어 종합환경 소개

소프트웨어 종합환경은 종합할 소프트웨어들에 대한 관리 디렉토리(directory)와 이에 맞게 개발 및 보완된 각종 지원도구들로 형성되어 있으며, 이것을 살펴보면 다음과 같다[8].

가. 디렉토리 구조

디렉토리 구조는 크게 최상위 노드인 OFC 노드가 존재하고 그 하위에 AT 노드가 존재함으로써 버전별 소프트웨어 종합환경이 구축되고 이것은 다시 하드웨어(hw), 펌웨어(fw), 문서(doc), 소프트웨어(sw)로 나누어져 체계적으로 관리된다. 이 구조에 대한 설명을 살펴보면 아래와 같다(세부 디렉토리 구조는 뒷편의 Appendix 참조).

나. 디렉토리 설명

소프트웨어 종합환경인 IDEAS(Integrated Software Development Environment for ATM Switching System-TDX-10에서는 t10sc 혹은 isrc[ISDN Switching System Resource Control]로 부름)에서

는 개발자의 작업영역을 작업노드(Working NODE)라 하고, 작업노드는 IDEAS의 기동시 자동 생성하며, 사용자는 제공된 디렉토리(작업노드)에서 개발을 진행한다. 작업노드의 생성 구조는 IDEAS의 generic별로 다를 수 있으나, ATM에서 쓰이는 일반적인 구조는 다음과 같다.

여기서 디렉토리의 top은 generic별 이름이 주어지고 이하 하드웨어, 펌웨어(firmware), 문서 및 소프트웨어의 집중 관리를 위한 subdirectory가 있으며, 이중 소프트웨어는 sw 이하에 구성되고 sw는 SWCDB(Software Component Data Base)인 swc를 포함하여 각각에 대한 subdirectory를 갖는 구조로 되어 있다. 이들 디렉토리에 대해 간략하게 소개를 하면 다음과 같다.

① swc: software component data base
② ap: application source directory
③ exec: 최종 블록별로 생성된 실행모듈이 위치
④ xref: VPATH상 현재 및 상위노드에 의해 생성된 cross-ference 정보를 포함
⑤ dg: dg 관련 directory(*.dg 및 F_* 관련 파일)
⑥ dgpool: PLD 관련 directory

exec 이하에는 PLD와 MMC의 DATA를 생성하기 위한 pld/data/dml subdirectory를 갖으며, 각각의 블록(blkn)에는 블록의 소스, SWCDB 정합용 CIF(Component Interface File)와 Makefile 그리고 Makefile을 생성하기 위한 conf 파일이 존재한다. ACE256 시스템의 장치제어계에 대한 OS(mSROS) 및 관련 펌웨어 소스를 관리하기 위한 환경(Tornado-I, II)은 별도로 구축되어 운용중이며 이에 대한 상세 구조는 생략한다.

다. 종합 지원도구 또는 개발 지원도구

소프트웨어 종합 지원도구들은 운용중 도구 자체의 문제나 새로운 기능의 추가나 보완에 따라 수시로 변경되어야 한다. 또한 필요에 따라 신규 개발이 필요하고 보완 개발된 도구들은 철저한 검증을 통해 높은 신뢰도를 유지하여 이용률을 높여야 한다. 개발자가 직접 개발한 도구들은 사용하기에 편리하도록 개량, 개선하여 종합환경을 지원하는 메인 프레임인 대형 컴퓨터의 성능을 높임으로써 시스템의 개발 능률을 향상시키는 요소로 작용할 수 있어야 한다[9].

V. 결론 및 향후 연구방향

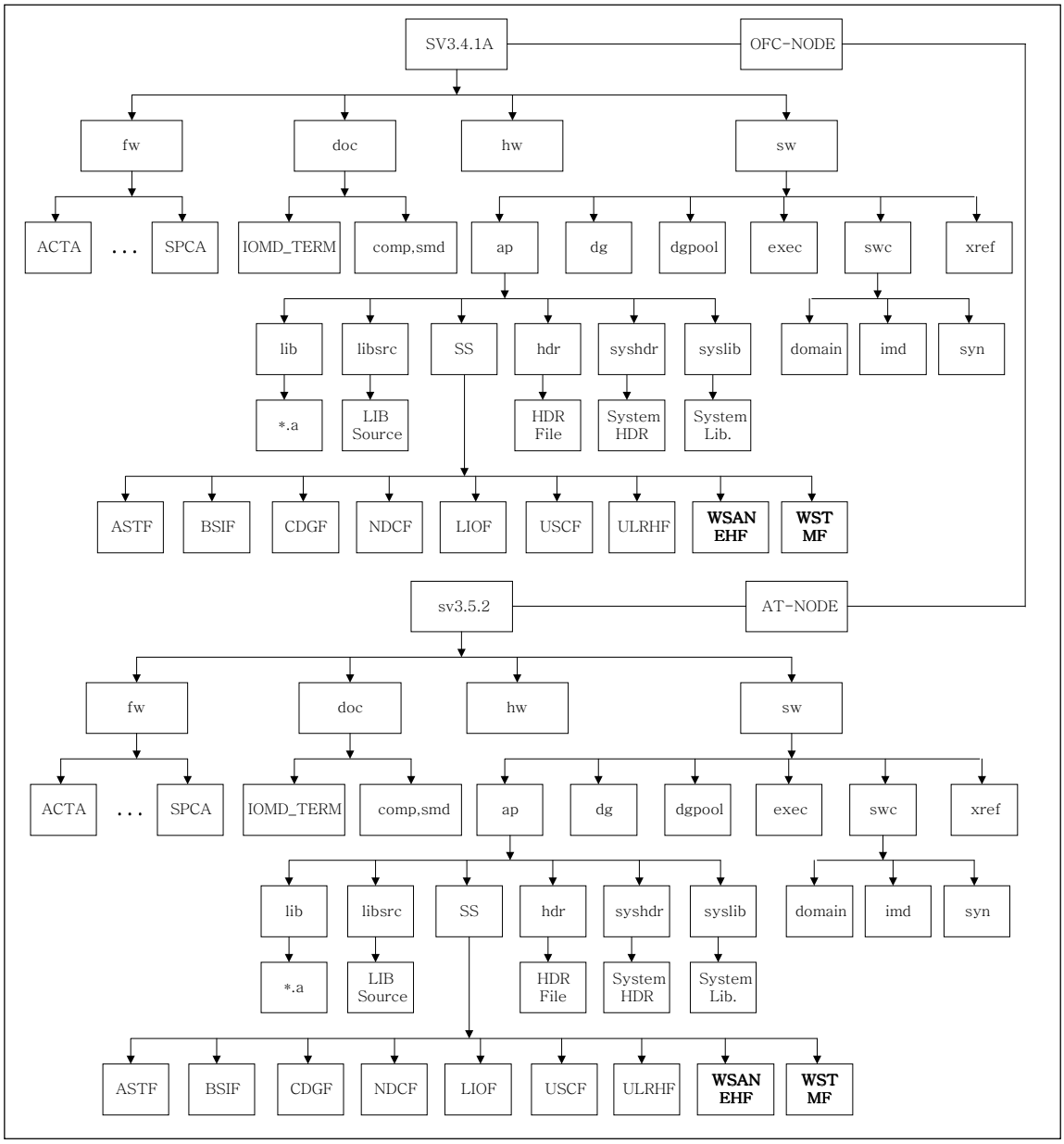
시스템이 대형화되고 사용자의 요구사항이 다양해지며, 빠르게 변화하는 추세에 부응하기 위한 소프트웨어 품질보증과 변경관리를 통한 형상관리는 사용자의 요구에 따른 적기 배포 측면에서 시스템의 종합, 시험기술과 함께 매우 중요하다.

소프트웨어 형상관리를 위한 종합방법과 지원환경은 대규모 소프트웨어 개발에 중요한 요소로 작용하며, 개발비용의 감소와 생산성 향상에 큰 영향을 미치기 때문에 소프트웨어 종합방법은 사업의 특성, 조직의 성격, 개발자의 능력과 개발환경에 크게 좌우된다. 절대적이고 최선인 방법은 존재하기 어려우며 이를 보완해 나가는 방법을 취해야 할 것이다. 이런 관점에서 좀더 연구되어야 할 사항은 개발계획 수립, 효율적인 시험환경 구축, 모듈의 복잡성과 결합도, 응집도, 새로운 개발환경 구축, 소프트웨어 재사용 방법과 신규 개발 방법론의 도입, 프로그래밍 언어의 선택, 개발능력에 따른 CASE Study 도입 등이다.

소프트웨어 개발과정에서 빈번히 발생하고 있는 변경요구 관리를 간단하고 일관된 절차로 요구의 신속한 대응을 목적으로 하는 변경제어 기법을 제안하고 도구의 운용사례를 소개하였다. 위의 기법들은 실제 개발 업무에 도입하여 소프트웨어 변경관리의 효율성과 신속성, 형상관리에 완벽성을 기할 수 있었다[10].

본 고는 소프트웨어 개발시 수행되는 제반 활동인 품질관리와 변경제어 기법을 도입한 형상관리 방법에 대해 소개함으로써 소프트웨어 개발에 대한 전반적인 이해와 대형 프로젝트를 수행하고자 하는 사람에게 도움을 주고자 한다.

부록: 노드별 디렉토리 구조도



참 고 문 헌

[1] 정왕호, 이기식, "소프트웨어 생산기술," 정익사, 1981. 11., pp. 198 – 205.

[2] 강금식, 품질경영(TQC, TQM), 박영사, 1997.

[3] John L. Hradesky, *Total Quality Management Handbook*, McGraw-hill Inc., 1995.

[4] QUART, "우리나라 기업의 6 시그마 적용을 위한 방안," 대한산업공학회 추계학술대회, 1998.

[5] K.S Kim *et al.*, "Tools and Techniques for Change SW Systems and Building SW Packages," ICCT'94, 1994. 6.

[6] 이재기, "ATM 교환기 기술전수 교재(I)-패키지 종합환경 중 변경이력관리 시스템(CRMS) 소개," 2000. 6.

[7] 김기재 외, “TDX-10 ISDN 소프트웨어 종합기술,” 한국정보과학회 소프트웨어공학연구지, 1993. 10.
[8] 신상권, “ATM 교환기 기술전수 교재(I)-패키지 종합환경 중 IDEAS 환경 소개,” 2000. 6.
[9] ETRI, “ATM 소프트웨어 설계 지침서,” 1996. 11.
[10] 이재기, “고장 데이터 분석을 통한 교환 소프트웨어 특성 연구,” 한국통신학회 논문지 제 23권 8호, 1998. 10., pp. 1915 – 1925.