소프트웨어 형상관리와 작업정보 리포팅을 통한 소프트웨어 제작 성능 향상

김 정 일[†] · 이 은 석[††]

요 약

대규모의 과제를 수행함에 있어 소프트웨어 형상 관리는 필수이다. 소프트웨어 형상관리 (Configuration Management)의 범주는 버전관리 외에도 각 개발자의 작업영역 관리, 소프트웨어 제작 관리, 개발 프로세스 제어부분까지 폭 넓게 포함하고 있다.

본 논문은 이러한 형상관리 부분에서 소프트웨어 제작부분의 최적화를 위해 다른 부분들이 어떻게 상호 유기적으로 조정되어야 하는지에 방향성을 두고 있으며 특히 형상관리 되어지는 정보 중에 어떠한 내용을 리포팅 할 때 소프트웨어 제작부분의 효율이 높아지는지에 대해 분석하고 그것을 기반으로 새로운 리포팅 시스템을 설계, 구현하고 평가하였다. 평가시에는 기존의 형상관리도구들의 관련 기능들에 대한 분석을 바탕으로 한 상대적 유효성에 대해 평가하였으며 각 리포팅 대상자들에게 관련 정보를 제공했을 때 생겨나는 변화에 대하여 설문 조사하여 그 유효성의 정성적 평가를 추가하였다.

키워드 : 소프트웨어 형상관리, 리포팅 시스템, 소프트웨어 제작, 형상관리도구

Performance Improvement of Software Build through Software Configuration Management and Work Information Reporting

Kim Jeongil[†] · Lee Eunseok[††]

ABSTRACT

A software configuration management(SCM) is essential for processing large scale project.

The scope of SCM involves each developer's work space management, software building management, and development process control as well as version control. In this paper we focus on what parts should be controlled systematically for the optimized software build that is an important part of the SCM. We also analyze that to increase the efficiency of software build, what kind of configuration management information should be reported. Based on the analysis, we have actually designed and implemented a new reporting system and evaluated it. The evaluation includes comparative evaluation in efficiency based on the analysis about the related functions provided by existing tools and some additional qualitative evaluation through the questionnaires from stakeholders.

Key Words : Software Configuration Management, Reporting System, Software Build, CM Tools

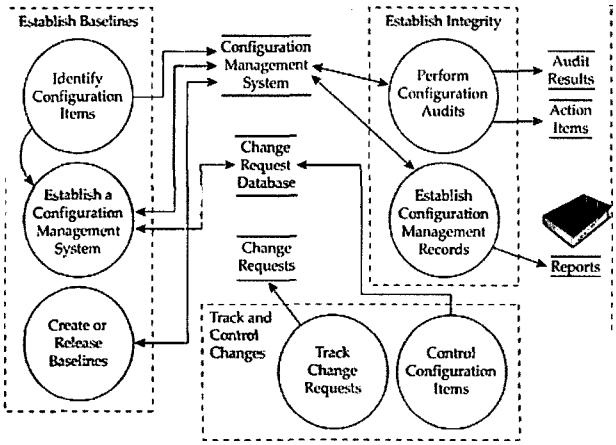
1. 서 론

형상관리(Configuration Management)의 정의는 소프트웨어 개발전반에 걸쳐 형상항목을 식별하고 정의해서 이런 형상항목이 변경되어 가는 것에 대해 기록하고 리포팅하고 완전성과 무결성을 보장해주는 일련의 개발관리 활동에 관한 지원 프로세스라 할 수 있다[1]. 소프트웨어 시스템은 유지보수, 사후 기능 강화, 다른 환경 또는 플랫폼에서 운영 될 수 있도록 서로 다른 다수의 모듈로 구성된다. 또 각 모듈들은 여러 버전으로 관리되어야 하며, 주어진 특정 환경에서는 한 모듈의 변경은 다른 여러 모듈에 영향을 미치게 된다. 따라서 형상관리를 통해 모듈의 다른 버전을 관리하여야 하며 올바른 모듈이 모여 형상이 이루어짐을 보증할 수 있어야 한다.

(그림 1)은 CMMI에서 정의한 형상관리의 전반적인 과정을 보여 주고 있다. 최종적인 결과물로 검사결과물과 실행항목과 리포트들이 나오는 것을 알 수 있다. 이중 본 논문에서 대상으로 하는 리포팅의 경우, 다음과 같은 주요한 역할을 가지고 있다[2, 3].

1) 각 역할별로 제공되는 리포팅 정보는 업무의 범위 및 업무의 현황을 명확하게 인식하여 작업하도록 한다.
2) 리포팅의 내용을 통해 소프트웨어 개발 과정의 여러

† 정 회 원 : 삼성전자 정보통신 총괄
†† 종신회원 : 성균관대학교 정보통신공학부 교수
논문접수 : 2005년 1월 12일, 심사완료 : 2005년 11월 1일



(그림 1) 형상관리 흐름

역할을 담당하는 사람들간의 의사소통에 도움을 준다.

형상관리를 위한 상용도구는 Lucent Technology의 Sablime [4], IBM Rational의 ClearCase[5], Telelogic의 Synergy[6] 등이 잘 알려져 있다. 이러한 형상관리 도구들도 과제 구성원들을 위한 리포팅 기능을 제공하고 있지만, 부분적이거나 단편적이어서 리포팅 본래의 역할을 충분히 못하는 실정이다. 구체적으로, 도구의 주요 사용자인 개발자와 소프트웨어를 종합해서 최종 결과물을 만드는 패키지 빌더(Integrator), 형상관리를 전담하는 형상관리자, 과제 관리자 등은, 각자의 관점에서 필요로 하는 정보는 어느 정도 얻을 수 있으나, 보다 전체를 파악하고 그것을 기반으로 개별 작업에 효율적으로 재반영 하기 위해서는 사용자의 많은 노력을 요구하게 된다.

기존의 형상관리 도구를 통해 실제 프로젝트 수행을 할 때 여러 관련자들이 필요로 하는 정보들이 제공되지 않거나 형상관리 도구의 자체문제로 인해 생겨나는 항목들을 정리하면 다음과 같다.

(1) 전체의 형상상태 정보가 제공되지 않는다.

정보를 얻기 위해서는 개발자나 관리자가 여러 과정을 추가적으로 처리해야 하기때문에 불편함이 따른다. 형상관리 상태를 기반으로 프로젝트의 진행사항을 나타내어 주는 정보가 부족하다. 이로 인해 개발자들의 모든 작업의 결과는 형상관리 시스템에 보관되나 실제로 프로젝트의 특정 지표로 반영되지는 않고 있다.

(2) 형상관리도구 사용 할 때 문제가 발생하면 시간 및 작업 손실이 크다.

예를 들면 ClearCase는 내부적인 프로세스간의 RPC(Remote Process Call)가 너무 많아 시간적인 면의 부하가 발생한다. 이러한 RPC가 개발자들의 작업영역간에도 공유하는 부분이 있기에 특정 개발자의 작업영역에 문제가 생기면 다른 개발자도 영향을 받게 되어 작업이 지연된다. 또한, 형상관리 도구는 자체 데이터베이스를 가지고 있다. 이 데이터베이스의 인터페이스 구현기술에 따라 생겨나는 문제점을 언급할 수 있다. 예를 들면 Sablime은 릴레이션을 바탕으로 하여 형상관리가 이루어진다. 이러한 데이터베이스의 릴레이션간의 불일치 문제가 발생하면 개발자들이 작업을 할 수 없게 된다. 이러한 관점에서 보면 관리자 환경이나 개발자 환경에 문제가 생겼을 때에 즉각적으로 대처하지 못하면 모든 개발자의 작업이 진행되지 못하는 큰 문제가 생겨난다. 즉 형상관리 중에 생겨나는 형상관리상의 문제점에 대한 리포팅이 빨라야 하고 정확할 필요가 있다.

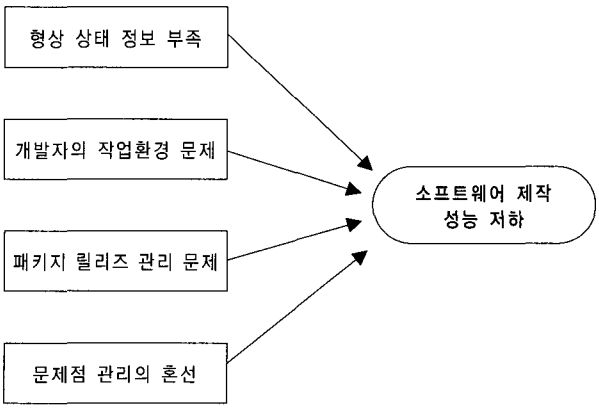
(3) 제품군별/사업자별로 다양한 버전의 패키지 릴리즈 관리에 어려움이 있다.

일례로 통신시스템을 중국, 일본, 인도네시아 등에 수출하게 되면 핵심기능은 비슷하겠지만 각 나라의 사업자별로 요구사항이 틀린 부분이 있다. 또 한 사업자에 대해서도 기능추가 및 개선의 요구가 생겨나면 다양한 버전세트(패키지)가 존재하게 되어지는데 이러한 상황에서 개발자 및 패키지 관리하는 사람간의 원활한 협조관계가 이루어지지 않으면 잘못된 작업으로 인해 혼선이 발생되어진다[7].

(4) 개발관련 문제점 해결을 위해 많은 시간이 소요된다.

개발 진행 중에 생겨나는 문제와 현장에 적용 후 각 사업자의 현장에서 생겨나는 문제점은 항상 존재한다. 등급에 따라 우선 적용하기도 하고 추후에 일괄적으로 적용하기도 한다. 이러한 문제점을 해결하기 위해서는 다양한 릴리즈와의 연관성을 고려한 작업환경이 필요하다. 문제점과 소스내용 변경과의 관계가 분명하지 않은 것도 문제가 된다. 예를 들면 A사업자에 a기능을 추가하기 위해 어떠한 소스를 변경하였는지 역 추적이 명확하지 않으면 이전 버전과의 차이를 매번 점검해야 하는 불편함이 있다. 또한 이런 문제점의 정보가 개발자간, 팀간에 공유가 되지 않아서 불필요하게 중복하여 처리하는 경우도 있다. 즉 문제점(MR : Modification Request)과 해당 작업 소스간의 연관성이 분명하지 않고 문제점 정보에 대해서 다양한 면에서 분석이 부족한 상태이다.

상기의 문제점을 간략하게 정리하면 (그림 2)와 같다.



(그림 2) 리포팅 기능과 관련한 형상관리의 문제점

본 논문에서 제기하는 근본적인 문제는 위와 같은 형상관리 중에 생겨나는 문제점들이 패키지제작에 영향을 주어 제작에 불필요하게 많은 시간이 소요된다는 것이다. 소프트웨어 제작의 성능을 올린다는 것은 짧은 시간 안에 문제점이 없는 양질의 소프트웨어를 제작하는 것을 목표로 한다

따라서 본 논문에서는 다음과 같은 특성을 갖는 리포팅 기능을 제안한다.

(1) 형상관리 시스템을 통해 관리되는 형상상태 정보를 설정하여 제공한다.
(2) 형상관리상의 문제에 대해 모니터링 가능하고 문제점에 대해 조속히 해결할 수 있는 정보를 형상관리자에게 제공한다.
(3) 패키지 제작에 대한 정보가 공유되어야 하며 빌더의 작업환경에 대한 정보를 제공한다.
(4) 문제점관리 시스템을 통해 관리되는 변경관련 정보를 개발팀의 요구사항에 맞게 제공한다.

제안하는 리포팅 시스템은 대표적인 형상관리 시스템인 ClearCase 와 연계해서 실제 설계, 구현 되었고 현업에서 정성적, 정량적으로 그 유효성을 확인하고 있다. 이하 2장에서는 관련도구를, 3장에서는 제안시스템의 구성과 주요 기능을 기술한다. 4장에서는 실제 구현내용과 평가내용에 대해 요약 정리한다. 5장에서는 결론 및 향후의 과제에 대해 기술한다.

2. 관련 연구

전장에서 언급한 문제점들을 해결하기 위해서는 형상관리 자체의 기술적인 측면에서 접근할 수도 있지만 본 논문에서는 관련 정보 즉 리포팅 기능 관점에서 접근하였다. 이외에 형상관리를 기반으로 한 데이터마이닝 기술을 이용한 접근방법[8, 9]도 있으나, 본 논문에서는 기존의 형상관리 도구의 리포팅 기능을 추가, 확장하는 것을 기본 방침으로 하여 관련 연구에서 제외하였다.

2.1 관련 도구 분석

CVS(Concurrent Versions System), Sablime, ClearCase 도구들의 기능에서 리포팅 관련 기능만을 별도로 정리하면 다음과 같다.

(1) CVS

CVS[10]는 GNU의 도구로서 RCS(Revision Control System)를 기반으로 한 버전관리 도구이다. 도구 내에 별도로 리포팅 관련된 명령어는 존재하지 않으나, 약간의 연관성을 갖는 명령어를 <표 2>에 정리한다. 이외 CVS 기반의 리포팅 도구로는 cvsmonitor(http://ali.as/devel/cvsmonitor)라는 것이 있는데 cvs 레포지토리내에 등록되어있는 내용을 분석해서 월별로 각 사용자별 진척 사항을 보여준다.

(2) Sablime

Lucent Technology사의 형상관리 도구로서 데이터베이스 구조가 여러 릴레이션으로 구성되어 있고 이를 통해 형상정보가 관리된다. 리포팅 관련하여 <표 2>와 같은 명령어가 제공된다. 한편, 리포트 도구를 사용하는 경우는 <표 1>과 같이 5가지로 요약되는 기능을 제공한다.

〈표 1〉 sablime 도구의 report 명령어 세부내용

항 목	주요 내용
mr reports	문제점 정보를 여러 폼으로 제공
external_mr	external project의 mr 리포트
group	각 role을 가진 group과 group원들의 정보제공
source	작업중인 파일의 정보를 제공
mr vs sfile	문제점과 관련하여 작업중인 파일의 리스트를 제공

(3) ClearCase

Unix/Linux 전용으로 사용하는 경우에는 별도의 리포트 도구가 존재하지 않는다. 단지 개별적인 cleartool command (clearcase 내부 shell 명령) 중에서 필요에 따라 선별하여 사용하여야 한다. 명령어로는 find, lsco, history 등이 있다. find 의 경우는 개발자가 작업한 내역을 검색하도록 제공해주는 도구로 개발자가 모든 항목을 갖추어 검색하기가 어려운 단점이 있다. window 버전의 경우는 report builder(2.2 참조)라는 도구가 있으나 지정된 형식에 한정하여 지원하고 있고 그 외의 필요로 하는 정보에 대해서는 지원이 어렵다.

위에 언급한 3가지의 형상관리 도구의 리포트 관련 사항을 정리하면 <표 2>와 같다.

상기의 내용을 종합해서 실제 과제에 적용할 때 고려되어야 할 사항은 다음과 같다.

〈표 2〉 형상관리 도구의 리포팅 관련 명령어 비교

도구	명령어	제공정보
CVS	status	현재 작업branch정보 및 labeling정보 제공, 현재의 파일의 작업상태 제공
	history	각 버전에 대한 작업이력을 보여줌. labeling한 tag정보 및 commit정보 제공
	annotate	각 버전의 line 단위의 작업이력에 대한 내역 제공
	log	해당파일의 version history에 대한 정보 제공
Sablime	query	단일 relation에 대한 정보를 제공
	report	복합 relation에 대한 정보를 제공. 5개 항목으로 구별하여 정보를 제공
	ssql	query와 report에 해당되는 것은 개발자가 직접 수행하는 query
	ptsaudit	과제내의 개발자의 Role 및 개발자 작업내역에 대한 audit수행 도구
ClearCase	report builder	2.2항목 참조
	find	개발자가 원하는 정보를 Query
	lsco	현재 개발자가 작업중인 정보를 제공
	fmt_ccase	특정명령수행에 따른 결과물 Format지정

CVS는 개인 및 소수에게 적합하다. 기능적으로 리포트 기능 관련 도구가 있지만 실제로 사용이 미비하다. Sablime은 내부 작업 프로세스가 명확하게 갖추어져 있고 MR기반의 작업이 가능토록 되어 있는 도구이다. 하지만 이 또한 개발자에게 한정된 리포팅 도구를 제공하고 있으며 조직에 적합한 형식의 결과물을 얻기는 어렵다.

ClearCase도 다양한 환경에서의 형상관리 기법은 제공하지만 필요로 하는 정보를 세분화하여 제공하는 리포트는 없다. ClearCase에 대한 세부적인 내용은 2.2 항목 후반부에서 언급 되어진다.

2.2 기존 대응 가능한 방법론

전장에서 제기한 네 가지의 문제점과 세 종류의 형상관리 도구의 리포팅 부분에 대한 기능 분석을 통해 필요로 하는 리포팅 정보를 정리해 보면 <표 3>과 같이 요약된다.

〈표 3〉 현재의 리포팅의 문제점과 필요한 리포팅 정보

문제점	주요 필요 정보	주요 관리자
형상상태 정보 부족	형상상태 정보	개발자
개발자의 작업환경 문제	개발자 작업환경 문제점 정보	형상 관리자
릴리즈 관리상의 문제	릴리즈 관련정보	패키지 빌더
문제점 관리의 혼선	문제점 정보	과제 관리자

<표 3>에서 분류한 4개의 항목에 대해 CVS, Sablime에서 제공하는 정보는 다음과 같다. CVS의 형상상태정보는 부분적이고 특정 파일에 한정해서 살펴 볼 때는 유익하지만 전체적인 프로젝트 관점에서 정보를 제공하는 것은 부족하다. 개발자 작업환경의 문제는 발생시에 개발자 작업환경에만 영향을 주기 때문에 관리자의 입장에서는 파악되는 것이 없다. 릴리즈 정보 측면에서는 레이블에 의해 붙여진 태깅 정보에 의해 관리가 되고 있다. 그러나 과제에서 CVS를 통해 복잡한 릴리즈 관리는 할 수 없다. 개발자에게 문제점과 연계해서 제공하는 프로세스가 없다. 수정내용에 관한 커멘트를 입력하는 정도 또는 버전 등록할 때 특정 스크립트가 수행되도록 해서 별도의 문제점 관리시스템과의 인터페이스를 가져갈 수는 있다.

Sablime의 형상상태 정보는 세분화되어 있어 다양한 리포트정보를 얻을 수 있다. 그러나 소스가 관리 되는 것을 직관적으로 알 수 없기 때문에 MR(Modification Request) 기준의 정보만이 의미가 있다. 개발자 작업환경의 문제점도 개발자별로 개인 사용 로그로 기록이 남기 때문에 이 파일의 분석을 통해 접근이 가능하다. 릴리즈 정보는 MR기준으로 릴리즈를 하기 때문에 릴리즈 된 여러 MR을 묶어서 관리하고 MR 리포트를 통해서 이러한 기능을 지원한다. 문제점 정보는 MR을 기준으로 모든 소스생성 및 변경, 패키징 작업이 수행되도록 프로세스화 되어 있기 때문에 MR 리포트를 통해서 관리되고 있다.

하기의 내용은 본 논문에서 제시하고 구현하고자 하는 리포팅 시스템의 적용대상 환경이 되는 ClearCase 관련된 주요 정보에 대한 현황이다.

(1) 형상상태 정보

현재 IBM에서 window용 클라이언트 버전에 report builder라는 도구를 통하여 <표 4>와 같은 정보를 얻을 수 있다.

〈표 4〉 ClearCase report builder의 기능

구 분	내 용
Element (형상관리 대상이 되는 파일 및 디렉토리)	특정 개발자에 의해 특정 날짜 이후로 작업중인 Element
	특정 개발자에 의해 생성된 Element
	Element type이 변경된 Element
	파일 이름에 의한 Element 검색
	특정 날짜 이후에 새롭게 등록된 Element
	날짜별로 등록된 버전
Attribute	특정 attribute 조건을 만족하는 Element(버전 등록시에 Element에 속성을 부여하는 것이 가능)
Branches	하위 브랜치가 생성된 후 상위 브랜치에서 변경된 Element
	특정 브랜치를 가진 Element
	다중 브랜치 구조를 가진 Element
	특정 브랜치에 최종 머지된 버전
	특정 브랜치의 최종 버전
	버전이 없는 브랜치
Labels	두개의 레이블간의 변경이 생긴 Element
	특정 레이블을 가진 Element
	레이블이 없고 특정 날짜보다 오래된 Element
	두개의 레이블간의 변경된 버전
	레이블을 가진 버전
Triggers	트리거를 가진 Element

이 도구를 사용할 때는 해당되는 값을 일일이 입력해 주어야 한다. 가령 위의 조건들을 조합해도 전체적으로 정리된 통합정보를 얻을 수는 없다. 또한 대다수의 개발과제가 Unix/Linux 환경에서 진행되기 때문에 개발자가 window 버전의 클라이언트를 설치해야 하는 불편함이 있다. 개발자의 작업환경을 samba도구를 이용해서 Unix/Linux 파일시스템을 window기반에서 작업할 수 있도록 하는 방법도 고려되나 개발자가 window, Unix/Linux 에서 이중 작업을 해야 하는 불편함이 있다. 또한 Unix 상에서는 특정 명령을 이용해서 형상상태 정보를 얻더라도 원시 데이터수준이기 때문에 추가적인 작업이 필요하다.

(2) 개발자 작업환경의 문제점 정보

개발자 작업환경에서의 형상관리도구를 이용하면서 발생하는 문제점은 관련 서버 프로세스의 로그로 시스템에 남겨진다. 현재는 위의 각 프로세스 로그에 관련 메시지만 저장될 뿐 이를 통해 어떤 조치가 자동적으로 이루어지는 것은 아니다. 또한 현재 형상관리 시스템의 사용 상태를 파악하는 도구로는 rgy_check가 있다. ClearCase 환경에서 작업을 할 때 레지스터리 서버라는 것이 있어서 개발자의 작업정보

에 대해 일치성을 점검하는 역할을 한다. rgy_check 도구를 통해 등록된 레지스터리의 정보와 틀려진 개발자의 작업환경에 대해서는 파악하여 알려 준다.

(3) 릴리즈 관련 정보

다양한 버전세트를 관리하기 위해 형상관리 도구에서 제공하는 일반적인 방법이 브랜치(Branch)와 머지(Merge)에 의한 작업 방법이다[11, 12]. 이런 브랜치와 머지에 대해 과제별로 적절한 전략을 가지고 운영하고 있지만 패키지를 구성하여 릴리즈 관리를 해야 하는 면에서는 레이블링하여 구별하는 기능 외에는 없고, 전술한 report builder도구에서는 레이블간의 비교 정보가 제공되고 있는 정도이다. 현재 개발자가 과제에 접근할 때 작업영역을 선택할 수 있도록 하고 있다. 즉 레이블링 기준으로 작업환경을 여러 개로 관리하고 개발자가 선택하여 작업할 수 있도록 하고 있다.

(4) 문제점 관련 정보

형상관리 범주에는 변경관리(Change Management)가 포함되어 있다. 이것은 소스의 변화와는 별개로 어떤 문제에 의해 어떤 소스가 변경되었는지를 파악할 때, 문제점에 대한 처리까지 소스와의 연관성을 두어 고려하고 있다. 형상관리 도구에서도 버전이 올라갈 때 코멘트를 통해 변경내용을 기술하도록 하고 있다. 그러나 과제를 관리하는 측면에서는 변경관리에 대한 프로세스가 반영된 상용 도구들이 있다. 현재 DDTS(IBM 제품)를 이용해서 변경관리를 하고 있는데 형상관리도구와 통합을 통해 해당 문제점에 대해 변경 소스를 파악할 수 있다. 또한 자체적으로 질의를 통해 리포팅 정보가 제공된다. 하지만 과제팀에 특화된 정보를 제공하기 위해서는 자체 리포팅만으론 부족함이 있다.

상기와 같은 기존의 도구들의 적용 한계점으로 과제팀에서 요구하는 항목에 적합하게 리포팅 되는 새로운 형태의 리포팅 시스템을 필요로 한다. 예를 들면 특정과제의 하루 단위의 문제점 처리 현황에 대한 종합적인 정보를 개발팀이 원하는 양식으로 제공하는 등 소프트웨어 제작에 직접 관련되는 리포팅 기능의 제공이 요구되고 있다.

3. 제안 시스템

3.1 설계 목표

형상관리라는 것은 업무 프로세스상에서 여러 관련자들이 원활하게 관련된 정보를 공유하면서 현재의 상태에 대해서 명확하게 파악가능 할 때 효율화를 높일 수 있고 궁극적으로 소프트웨어 제작의 효율을 높일 수 있다. 이를 위해 역할별로 필요로 하는 정보를 효율적으로 얻을 수 있도록 리포팅 시스템을 구축한다.

3.2 시스템 구성 및 주요기능

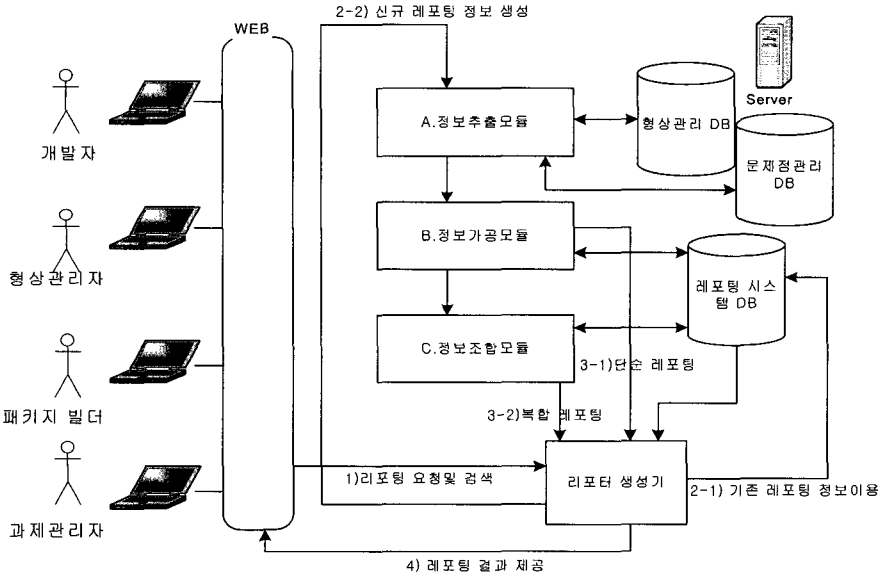
제안 시스템의 구성은 (그림 3)과 같다.

정보추출 모듈(A)은 각 역할별로 필요로 하는 정보 검색 및 요청을 하였을 때 리포팅 시스템 DB내에 저장되어 있지 않는 정보나 신규로 생성을 요청하는 정보의 경우 동작하는 부분이다. 형상관리 DB와 문제점 관리 DB에서 필요로 하는 원시데이터를 가져오는 부분이다.

정보가공 모듈(B)은 정보추출 모듈에 의해서 가져온 원시데이터에서 필요로 하는 정보를 얻기 위해 재 조정하는 모듈이다. 이 모듈에 의해서 실제로 원하는 결과값을 얻을 수 있는 경우도 있다.

정보조합 모듈(C)은 정보가공 모듈에서 얻어진 결과값과 기존의 처리된 결과값으로 리포팅 시스템의 데이터베이스에 저장되어 있는 내용을 조합해서 세부정보를 얻고자 하는 경우에 사용되는 모듈이다.

이와 같이 처리된 정보가 각 역할별로 요청하는 정보에



(그림 3) 시스템의 전체 구성도

의해서 실시간 별로 처리되는 루틴(2_2)이 있고 저장해 두었다가 필요할 때 처리하는 과정(2_1)으로 나뉘어져 있다.

리포트 생성기를 통해 최종적으로 제공되는 정보는 4개의 역할(개발자, 형상관리자, 패키지 빌더, 과제관리자)별로 각각 <표 5, 6, 7, 8>에 정리하였다.

주요기능은 2 항목에서 살펴 본 기존 형상관리 도구에서 제공되는 정보의 부족한 부분을 보안하기 위해 구체적인 필요로 하는 정보가 제공되도록 구현했다. <표 5, 6, 7, 8>의 정보는 특정 범주에 있는 사람만을 대상으로 한정해서 제공하지는 않는다. 그렇지만 우선적으로 필요로 하는 대상이 존재한다. 또한 <표 5, 6, 7, 8>의 항목은 각 관련자에게 필요한 부분에 한정하여 정보를 제공하는 것이다. 추후에 추가되거나 조합이 되어서 정보의 변경을 가져 올 수도 있다. 실 시간적으로 정보를 가져와야 하는 경우도 있겠지만 이런 경우는 시간이 많이 소모되기에 주간단위로 일단위로 제공하는 것도 고려하여 설계하였다.

아래는 정보가공 및 분류를 통해서 역할별로 제공되는 정보의 내용을 기술하였다.

앞으로 기술된 표의 항목부분에 기능들을 3가지로 분류할 수 있도록 하였다. (표항목 우측 부분에 표시)

* 기존에도 구할 수 있는 정보의 포맷변경
** 기존 정보에 다른 정보를 조합하여 변경이 생긴 경우
*** 기존의 기능과는 별개로 새로이 생겨난 기능

3.2.1 형상상태 정보

형상상태 정보를 통해 개발자들이 현재 자신의 작업상태를 정확하게 파악할 수 있도록 한다. 이로 인해 불필요한 작업이 생겨나지 않도록 하고 개발자간의 원활한 의사소통이 이루어 질 수 있도록 한다[13].

<표 5>의 내용은 기존의 형상관리도구에서 기본적으로 제공하는 정보뿐 아니라 별도의 가공을 통해 정리된 정보가 기술되어 있다. (그림 3)에서 언급한 정보가공과 정보조합모듈 부분이 필요한 경우가 많다. 아래 정보는 현업에서 개발자들이 요구하고 필요로 하는 정보를 취합해서 정리된 것이다.

또한 아래 정보를 얻기 위해 개발자의 주간 작업 패턴에 대한 분석도 하였고 이를 통해 의미 있는 정보에 대해 개발팀의 의견을 구해 정리하였다.

3.2.2 개발자 작업환경의 문제점 정보

개발자 작업환경에 문제가 생겼을 때 개발자가 곧바로 처리하지 못하는 부분은 형상관리 담당자가 문제해결을 위해 관여하게 된다. 이런 상황을 포함하여 형상관리 담당자의 업무처리에서 필요로 하는 정보를 정리하였다[14]. 또한 개발자의 작업환경에 문제가 생겼을 때 이에 대한 조치가 빠르게 이루어지도록 하며 미리 문제에 대해 예측하고 조치할 수 있도록 한다. 형상 관리자가 위의 개발자 작업환경의 문제점뿐 아니라 운영상의 현황파악을 위한 리포팅 정보도 필요로 한다. <표 6>의 2-5 항목에는 형상관리자의 구체적인

〈표 5〉 개발자에게 제공되는 정보

항 목	주요 정보
1-1. 현재 자신의 작업 정보 제공* (checkout 하여 수정중인 파일 정보제공)	일자, 작업view, 디렉토리명, 파일명
1-2. 관련 블록 및 관련 개발자의 작업 정보** (타 개발자의 checkout 하여 작업중인 제공)	디렉토리명, 일자, 개발자이름, 작업view, 파일명, Base 버전
1-3. 작업 파일의 history및 변경사항 정보 제공** (작업파일의 변경내용에 대한 정보)	파일명, 버전, 작업자, 레이블 정보
1-4. 주간 작업량 정보 제공*** (개발자가 주간별, 또는 릴리즈별 작업정보 제공)	주간, 일자, 디렉토리명, 브랜치, 파일명, 버전, 릴리즈, Base,
1-5. DO(Derived Object) 정보제공* (기존에 생성된 object정보)	일시, 작업자, CR정보
1-6. CR(Configuration Record) 정보 제공** (결과물과 관련되어진 소스 파일의 dependency 정보)	일시, 작업자, 관련파일, 작업 스크립트
1-7. 할당되어진 문제점별 변경되어진 소스파일 정보 *** (문제점 별로 변경되어진 소스 정보 제공)	MR, MR상태, 소스리스트
1-8. 작업 영역 관련 정보** (작업 view 정보)	View이름, 종류, 현황, 문제점

〈표 6〉 형상관리자 제공정보

항 목	주요 정보
2-1. 형상관리 시스템 운영관련 정보*** (과제 운영현황 파악 및 주간 작업량 파악)	과제명, 개발자, 디렉토리, 파일, 소스변경량
2-2. 개발자 작업상의 문제점 정보** (개발자 및 문제점 정보를 제공함)	개발자, 문제점, 처리내역
2-3. 개발시스템별 사용현황 정보*** (vob의 공유 및 클라이언트 시스템 설치 현황정보)	시스템명, 과제명, 개발자명, 클라이언트 버전, 패치 버전
2-4. 로그정보를 기반으로 한 문제점 처리***	메시지별 에러유형 파악
2-5. 형상관리 운영 정보*** (형상관리상의 변경 및 주요사항 관리정보)	형상관리상의 변경내용, 변화추세

업무영역의 리포팅이 포함되어 있다. 이 내용은 기존에 구축되어 있는 항목이기에 세부사항은 제외하였다.

3.2.3 릴리즈 관련 정보

각 개발자들이 작업완료 후에 최종적으로 소프트웨어를 제작하는 패키지 빌더에게 우선적 의미가 있다[15]. 빌더가 작업한 내역들이 관리되고 각 레이블버전별로 진행되고 있는 상황에 대해서 개발자들에게 공유되도록 한다. 빌더와 개발자간의 의사소통이 이루어지는 정보가 제공된다. 동일한 소스를 가지고 다양한 패키지를 제작하는 빌더는 개발자에게 다양한 작업환경을 제공하게 된다. 이러한 부분에 명확한 의사소통이 되도록 관련정보를 파악하여 개발자의 잘못된 작업을 최소화 시켰다. 또한 패키지 간의 변경내용을 명확하게 파악하도록 하였고 제작 패키지 내에서 생긴 문제점 처리하는 부분에 대해서도 빌더가 모니터링 할 수 있도록 하였다.

〈표 7〉 패키지 빌더 제공정보

항목	주요 정보
3-1. 브랜치별 작업현황 정보 및 머지(Merge) 정보 파악** (개발자의 특정 시점 이후 작업형태 및 작업결과 정보)	작업일시, 개발자, 브랜치명, 디렉토리, 파일명, 버전
3-2. 패키지 제작 정보를 제공*** (기존 제작되어진 패키지 릴리즈 정보를 개발자와 공유)	일시, 레이블명, comment
3-3. 신규 패키지와 기존 패키지와의 변경내용 정보제공*** (변경내용에 대한 정보)	릴리즈, Base, 개발자, 일시,디렉토리,파일명,버전
3-4. CR(Configuration Record)정보 제공** (결과물과 관련되어진 source 파일의 dependency 정보)	일시, 작업자, 관련파일, 작업 스크립트
3-5. 패키지 제작 후 문제점 해결 공유*** (패키지 제작 후 검증단계에서 발생한 문제점 관리)	문제점번호, 상태, 제목, 내용, 관련 작업자
3-6. 패키지의 변경소스 분석내용 ** (패키지 제작 후 블록별 변경소스량의 정보)	블록명, 추가 라인수, 삭제 라인수, 변경라인수

3.2.4 문제점 정보

과제 관리자에게는 소스의 변경내용보다는 문제점 처리가 어떻게 되고 있는가의 정보를 제공하는 것이 유익하다. 개발자들에게 할당된 MR들의 처리현황을 수시로 점검하여 과제 관리를 할 수 있도록 한다.

이를 위해 상용도구가 아닌 자체적으로 제작한 문제점 관리 도구도 존재하며 과제팀에 특화된 주간, 월간 리포트를 제공하여 과제의 문제점 해결 진척상황을 파악할 수 있도록 하였다. 현재 문제점 관련해서는 메트릭으로 관리하여 별도의 분석자료로 사용하고자 진행 중이다.

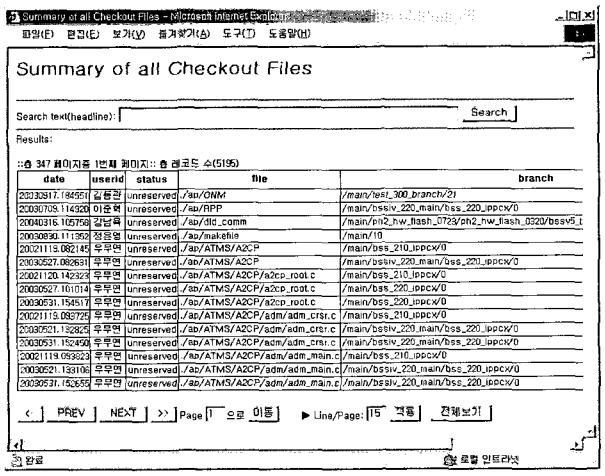
〈표 8〉 과제 관리자 제공정보

항 목	주요 정보
4-1. 문제점 처리 현황정보** (MR별 처리상황 정보)	과제명, 문제점상태, 세부내역
4-2. 작업자별 작업 현황 정보** (주간에 개발자별 진행 상황)	과제명, 개발자, 디렉토리, 파일, 소스 변경량
4-3. 패키지 릴리즈 현황 정보*** (패키지내에 반영되어진 MR정보)	릴리즈명, 제작일시, 포함된 문제점 들
4-4. 일간, 주간, 월간 리포트***	과제팀에 적합하게 customizing된 문제점 요약 정보

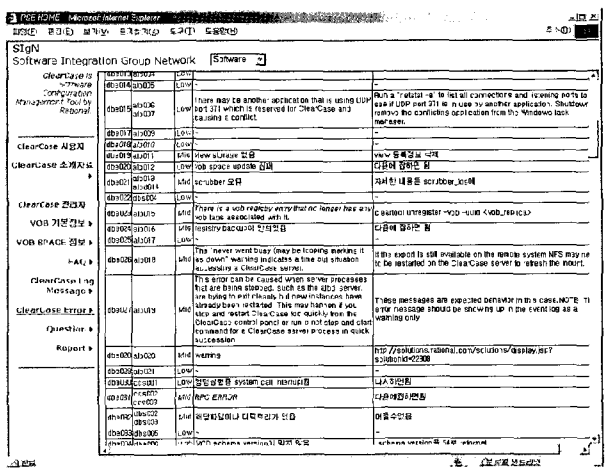
4. 구현 및 평가

시스템의 구현 환경은 CGI 기반의 Web환경으로 perl 과 ksh을 이용하였고 결과물은 xml과 html로 web 브라우저를 통해 관련자에게 제공된다. 주기적인 시간에 관련자에게 이메일로 통보되는 경우도 있다. 구현 결과물의 예는 아래 그림과 같다.

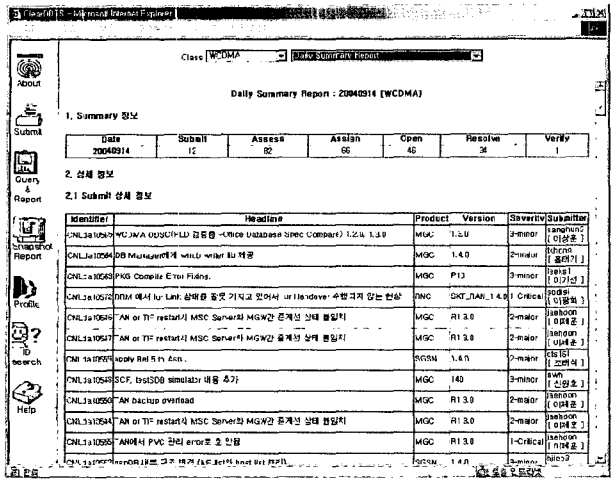
(그림 4)는 <표 5>의 1-2항목의 구현 예이다. 관련 블록의 작업중인 파일들에 대한 정보를 개발자에게 제공한다. (그림 5)는 <표 6>의 2-4항목의 구현 예로써 형상관리 시스템의 로그를 분석하여 형상관리자에게 제공한다. (그림 6)은 <표 8>의 4-4항목의 문제점 리포트 중 과제팀의 관리자에게 제공되는 일일단위의 리포트 정보이다.



(그림 4) 개발자의 checkout 정보



(그림 5) 형상관리 시스템 log 분석정보



(그림 6) 일간 문제점 요약정보

구현한 기능이 기존의 형상관리도구에서의 리포팅 기능과 어떤 차별화가 있는지를 살펴보면 <표 9>와 같다.

비고란의 항목들의 분리기준은 다음과 같다. 구현시스템과의 비교기준은 ClearCase가 되어진다.

-기존 : 기존에도 구할 수 있는 정보의 포맷변경.
-보완 : 기존 정보에 다른 정보를 조합하여 변경이 생긴 경우
-추가 : 기존의 기능과는 별개로 새로이 생겨난 기능

〈표 9〉 기존형상관리 도구와 구현 기능간의 비교

대상자	기능	구현 시스템	CVS	Sablime	Clear Case	비고
개발자	개인작업정보	O	△	△	O	기존
	타인작업정보	O	X	△	△	보완
	history,변경정보	O	△	O	O	보완
	주간작업량정보	O	X	X	X	추가
	Object 정보	O	X	X	O	기존
	Dependency정보	△	X	X	△	보완
	문제점관련파일정보	O	X	X	△	추가
	작업환경정보	O	X	X	O	보완
형상관리자	형상관리 운영시스템정보	O	X	△	△	추가
	개발자작업문제점	O	X	O	O	보완
	시스템별 현황정보	O	X	△	△	추가
	log 기반 정보	O	X	△	△	추가
	형상관리 운영정보	O	X	△	△	추가
패키지빌더	Branch/Merge정보	O	X	X	O	보완
	패키지제작정보	O	X	O	△	추가
	패키지간 비교정보	O	X	X	△	추가
	패키지 관련 CR 정보	△	X	X	△	보완
	패키지제작문제점정보	O	X	O	△	추가
	블록별 변경소스량 정보	O	X	X	X	추가
과제관리자	문제점처리정보	O	X	O	△	보완
	개발자별 문제점정보	O	X	O	O	보완
	패키지별 문제점정보	O	X	O	△	추가
	일간,주간 문제점정보	O	X	X	X	추가

〈표 10〉 구현 기능항목별 주요 장점

기능항목	구분	세부 항목	주요 효과
1.개발자	기존	1,5	정형화된 format으로 정보제공
	보완	2,3,6,8	복합적이고 세부적인 정보를 얻게 됨
	추가	4,7	개발자가 전체적인 종합정보를 얻게 됨
2.형상관리자	보완	2	개발자의 문제점을 사전에 처리할 수 있게 함
	추가	1,3,4	현 운영 상황에 대한 모니터링 효과
			각 개발호스트 간의 작업 내역 모니터링 효과
			문제점 분석을 통해 향후 문제점 발생시에 효과적으로 대처
3.패키지빌더	보완	1,4	빌더가 개발자의 작업영역별 정보 확인 가능
	추가	2,3,5,6	개발자와 빌더, 시험자간의 의사소통 정보 제공
			문제점 발생시에 기존 안정화된 버전과의 비교를 통해 변경내용 제공
4.과제관리자	보완	1,2	복합적이고 세부적인 정보를 얻게 됨
	추가	3,4	과제팀내에 특화된 정보를 얻게 됨.

<표 10>은 주요 기능항목에 대해 구현한 항목들을 통해 제공되는 정보를 통해 얻게 되는 장점들을 정리한 내용이다. 이들 항목들은 <표 5, 6, 7, 8>에서 나온 것이다. 보완 또는 추가된 기능으로 인하여 네 종류의 사용자 관점에서 아래와 같은 효과를 거둘 수 있었다.

제안 시스템에 대한 평가는 소프트웨어 제작 효율에 대한 유효성을 정량적으로 평가하기 위하여 제작 업무에 직접 관련되는 빌더 관점에서 평가작업을 하였다. 즉 패키지관리의 용이함을 입증하기 위해 패키지 제작정도의 변화와 패키지 제작 후 문제점 발생 건수의 변화를 살펴보았다. 또한 개발자 및 관리자를 포함한 설문조사를 통하여 리포팅 시스템으로 통하여 어떤 변화가 있었는지도 살펴보았다.

1) 특정기간 내 패키지 제작 정도의 변화

개발자의 작업 단계에서 불필요한 실수가 감소함에 따라 패키지 제작회수가 줄어 들게 되었다. 이와 관련하여 형상관리 메트릭[16]이용하여 값을 구할 수 있다.

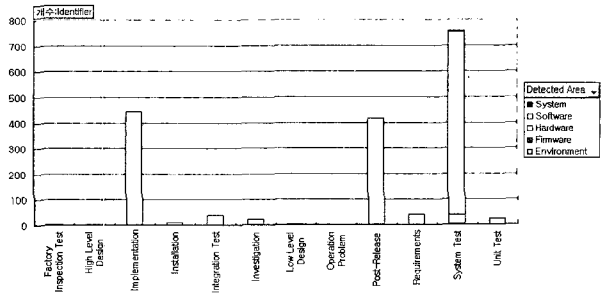
위의 형상관리 메트릭은 각 테스트 단계(시스템 시험/사용자 인수 시험등)중에 문제점이 어느 정도 파악 되는가? 정상적인 릴리즈를 위해 패키지 제작 횟수, 릴리즈 주기는 어떠한가? 전체 릴리즈는 어느 정도 발생하는가? 성공적인 릴리즈는 어느 정도인가를 수치화 한 메트릭이다.

이중에 성공적인 패키지 제작회수의 비율을 구하는 것이 가능하다. 이전 체계하에서는 관리되는 과제 중에 패키지 제작 후 부팅 불량률이 50% 정도로 재 작업 후에 패키지 제작을 하는 횟수가 많았으나 형상관리와 리포팅 시스템을 통해 0%에 가깝게 부팅 에러가 줄어들게 되어 패키지 제작 회수가 획기적으로 줄어든 사례가 있다.

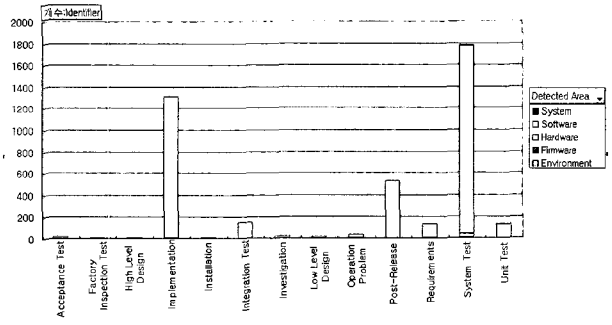
2) 패키지 제작 후 문제점 발생 건수의 변화

검증단계에서 발생되는 문제점에 대해 변경관리 시스템에

등록되어 문제점이 처리될 때 소스와의 연계를 통해 문제점을 처리함에 따라 동일한 문제점이 중복하여 발생하는 경우가 줄어든다. 또한 CR(Configuration Record : Object 생성 관련정보 추출가능) 정보 및 릴리즈간의 소스 가운데 변경된 정보를 제공함으로 인하여 문제점 처리의 시간을 줄일 수 있게 된다. (그림 7)과 (그림 8)은 리포팅시스템 적용 전후의 문제점 발생의 변화를 나타내고 있다. 동일 그림을 통해 자체검증, 신뢰성시험, 인수시험의 단계에서의 문제점 등록이 줄어들었음을 알 수 있다. 특히 이 정보가운데 현장문제(post release)의 비율이 기존 24% 에서 12%로 감소 된 것으로 파악되었다.



(그림 7) 적용전의 문제점 등록현황



(그림 8) 적용후의 문제점 등록현황

이 비율이 줄어든 주된 이유는 다음의 사실에 기인한다.

리포트 시스템을 통해 개발자들이 자신의 작업환경과 타인의 작업환경에 대한 이해도가 높아짐으로 인해 잘못된 소스등록이 줄어든다. 블록별 소스변경량 정보가 제공되어 변경량이 많은 부분은 집중적으로 코드리뷰를 진행하고 자체시험을 할 때도 중점적인 테스트를 하도록 하였다. 이로 인해 릴리즈 후에 생겨나는 패치가 줄어들게 되었다.

문제점이 등록되는 것에는 많은 요인들이 있기에 오랜 시간을 두고 상호관련성에 대한 추가적인 검증이 필요하다.

3) 설문에 의한 파악 건

<표 11>은 형상관리 기반의 리포팅 시스템을 적용하기 전후의 개발자 및 빌더와 관리자의 설문내용을 기반으로 작성되었다. 3개의 과제에 대해서 각 과제별로 9명 (개발자 5명, 빌더 2명, 관리자 2명)씩 설문한 결과이다. 관련 업무부분에 소요되는 시간이 줄어들고 정보량은 더욱 만족스러워진 것을 알 수 있다. 시간효율은 대분류의 항목 관련된 정보를 얻기 위해 기존에 소요되던 시간과 비교해서 현재가 어떠한지를 비교 설문한 것이고 정보효율은 기존의 없거나 있던 정보에 대해 리포팅 시스템을 통해 얻게 된 현재의 정보의 내용에 대해 만족정도가 어떠한지를 평가한 것이다. 설문자로 각 항목에 대해 1~5점을 부여하도록 하여 평균 3.5 이상인 경우는 상으로 평균 2.0~3.5 미만은 중으로 분류하였다. 평가결과값의 상은 만족하는 경우이고 중의 경우는 보완이 필요한 것을 반영한 것이다.

5. 결 론

실무에서 리포팅의 중요성이 높음에도 불구하고 기존의 형상관리시스템은 각각의 사용용도나 범위에 맞추어 한정적인 리포팅 기능만을 제공하고 있어 다양한 프로젝트와 조건에 맞는 리포팅 시스템에 대한 요구가 절실하다.

본 논문에서는 기존의 형상관리 시스템들이 가지고 있는

〈표 11〉 설문에 의한 평가 결과

대분류	항 목	시간 효율	정보 효율	주요 관련항목
릴리즈 & 빌드 관리	릴리즈 내용 및 상태 리포팅에 대해 만족하는가?	상	상	3-2, 3-3, 4-3
	릴리즈에 관련된 문제점 정보에 대해 만족하는가?	상	상	1-7, 3-2, 4-3
	릴리즈 결과물에 포함된 내용이 김증되는가?	상	상	1-5, 1-6, 3-4
의존성 추적	릴리즈간 변경사항 리포팅 문서가 제공되는가?	중	중	1-3,
	코드간 의존상태 정보가 제공되는가?	상	상	1-6,3-4
재현성	문제발생시에 이전 작업환경으로 복원 가능한가?	중	중	1-6,3-4
작업영역 관리	릴리즈별 작업환경 설정이 별개로 가능한가?	상	상	2-1,3-2
변경영향 파악	변경에 따라 생겨나는 영향이 예상되는가?	중	중	4-4,
변경요구 관리	수시로 생겨나는 명확한 변경 요구가 관리되는가?	상	상	4-1,4-2
문제점 관리	문제점의 상태 파악이 되는가?	상	상	4-1
프로젝트 정보	관련 프로젝트 정보 찾기가 쉬운가?	상	상	1-4,3-2, 4-4
	프로젝트, 변경요구의 메트릭 정보 제공에 만족하는가?	중	중	1-4,4-3
	프로젝트 상태 리포팅	중	중	1-4,4-3

리포팅 기능을 보완, 개선하기 위한 새로운 리포팅 시스템을 제안하였다. 제안 시스템을 통해서 개발 단계상에서 관련자인 개발자, 형상관리자, 빌더, 프로젝트 관리자에게 필요로 하는 정보를 제공함으로 인해 불필요한 문제점의 감소와 불필요한 패키지 제작의 횟수가 줄어들게 하는 성과를 거두었다. 각 역할별로 얻게 된 이점을 간략히 정리하면 다음과 같다. 개발자 입장에서는 자신의 진행현황을 다른 사람의 작업내역과 비교해서 확실하게 알 수 있으며, 형상관리자는 개발자의 작업상의 문제점 파악 및 형상관리 운영현황에 대한 정보를 기반으로 업무처리를 할 수 있게 되었다. 빌더는 패키지 관리에 대해 투명한 업무처리를 제시할 수 있게 되었고 이의 공유를 통해 불필요한 패키지 제작의 횟수를 줄일 수 있게 되었고 프로젝트 관리자는 현재 개발상의 문제점과 형상관리상의 변경정도를 통해 프로젝트 수행에 대한 우선순위를 조정할 수 있게 되었다. 또한 형상관리 관련자들에게 유용한 정보를 제공함으로 인해 업무에 소요되는 시간과 정보의 질적인 면에서 효과가 있음을 알게 되었다. 그러나 문제점 감소 부분에 있어서 리포팅 시스템 외의 변수요인으로 인한 것도 고려할 수 있기에 향후 장기적인 데이터를 기준으로 추가적인 비교 평가를 수행할 필요가 있다.

더불어 4개 항목에 대한 세부리포트 기능을 필요에 따라 추가로 제공할 필요도 있다.

향후 형상관리시스템 내에서 형상관리 지표에 해당되는 항목을 선정하여 산출하는 것과 소스 코드 분석도구 및 테스트 도구와의 통합을 통해 소프트웨어 품질을 올리는 것을 고려하고 있다.

참 고 문 헌

[1] Anne Mette Jonassen Hass, "Configuration Management Principles and Practice", pp.3-27, Addison Wesley, 2002.

[2] Dennis M. Ahern, Aaron Clouse and Richard Turner, "A Practical Introduction to Integrated Process Improvement", pp143-148, Addison Wesley, 2003.

[3] David M. Dikel, David Kane and James R. Wilson, "Software Architecture Organizational Principles and Patterns", pp.17-37, Prentice Hall, 2001.

[4] Belllab, "Sablime User's Reference Manual", Lucent Technologies, 2002.

[5] Brian A. White, "Software Configuration Management Strategies and Rational ClearCase", pp.51-93, Addison Wesley, 2001.

[6] Telelogic AB, "Guidelines for Evaluating a Change & Configuration Management System", pp.3-25, Telelogic, 2003.

[7] Michael E. Bays, "Software Release Methodology", pp. 127-205, Prentice Hall PTR, 1999.

[8] Ahmed E. Hassan, Richard C. Holt, and Audris Mockus, "MSR 2004 international workshop on mining software repositories Software Engineering", IEEE. ICSE Proceedings, pp.770-771, 2004.

[9] Thomas Zimmermann "Mining Version Histories to Guide Software Changes", IEEE. ICSE Proceedings, pp.563-572, 2004.

[10] Vesperman, Jennifer, "Essential CVS : Version Control and Source Code Management", O'Reilly, 2003.

[11] Chuck Walrad and Darrel Strom, "The Importance of Branching Models in SCM", IEEE Transactions on Computers, Vol.35, Issue 9, pp.31-38, Sept., 2002.

[12] Brad Aooleton and Darry A.Hahn, "Selection The Right Branching Solution : Techniques, Strategies and Trade-offs", Rational User Conference, 2003.

[13] Susan A. Dart, "The Urgent Need for Configuration Management and Benefits of Automation", CM Crossroads Journal, 2000.

[14] Ronald van der Linger and Andre van der Hoek, "An Experimental, Pluggable Infrastructure for Modular Configuration Management Policy Composition", IEEE. ICSE 2004 Proceedings, pp.573-582, 2004.

[15] Mario E. Moreira, "ABCs of Release Management", CM Crossroads Journal, August, 2004.

[16] Stephen H. Kan, "Metrics and models in software quality engineering", Addison Wesley, 2001.

김 정 일

e-mail : solar9@samsung.com

1996년 중앙대학교 컴퓨터공학(학사)

1996년~현재 삼성전자 책임연구원

2005년 성균관대학교 컴퓨터공학(공학석사)

관심분야 : 형상관리, 변경관리, SW품질관리, SW Build/Release Management 등

이 은 석

e-mail : eslee@ece.skku.ac.kr

1985년 성균관대학교 전자공학과(학사)

1988년 일본 Tohoku(동북)대학교 정보공학과(석사)

1992년 일본 Tohoku (동북)대학교 정보공학과(박사)

1992년~1993년 일본 미쯔비씨 정보전자연구소 특별연구원

1994년 일본 Tohoku (동북)대학교 Assistant Prof.

1995년~현재 성균관대학교 정보통신공학부 교수

관심분야 : 소프트웨어공학, 유비쿼터스컴퓨팅, 오토노믹컴퓨팅, 에이전트지향지능형시스템등