형상관리 기반 설정파일 버전 무결성 감사 프레임워크

김선주*, 이석훈**, 백두권** †
*고려대학교 컴퓨터정보통신대학원 소프트웨어공학과,
**고려대학교 컴퓨터전파통신공학과
*sjking@korea.ac.kr, **{leha82, baikdk}@korea.ac.kr

Configuration Management based Configuration File Version Integrity Auditing Framework

Seonjoo Kim*, Sukhoon Lee**, Doo-Kwon Baik** †
*Dept. of Software Engineering, Korea University
**Dept. of Computer and Radio Communications Engineering, Korea University

요 약

최근 기업에서 구축하는 IT 아키텍처가 점점 복잡해져 가는 환경변화에 따라 각 시스템 구성요소의 조건이나 특성을 저장하는 설정파일(Configuration file)의 중요성이 강조되고 있지만, 대부분의 형상관리시스템이 응응 소프트웨어를 중심으로 구성되어 설정파일의 특징을 반영한 활동에 한계를 갖고 있다. 또한 품질보증 목적으로 수행하는 형상감사 활동도 응용 소프트웨어를 대상으로 품질중심 감사 활동이 이루어지면서, 등록된 개별 소프트웨어 별로 변경통제와 버전관리가 이루어져 설정파일이 포함되는 경우 버전 무결성 차원의 문제를 적발하지 못하는 문제가 있다. 이 논문에서는 추가적으로 필요한 감사활동 요구기능을 정의하고, 동일한 원본에서 배포된 설정파일들이 서로 버전이 다르게 관리 될 수 있는 문제를 감사(Audit)를 통해 탐지할 수 있는 버전 무결성 감사 프레임워크를 제시한다. 제안하는 프레임워크는 기존 형상관리 개념과 액티비티, 프로세스를 기반으로 감사 기능을 보완한다. 이를 통해 기존 감사활동에 버전 무결성 검증을 수행하는 형상감사 기능이 포함되며, 이는 설정파일의 버전 차이에 의해 사전/사후 발생될 장애, 오동작 등의 문제 해결에 활용된다. 이 논문에서 제안 프레임워크의 검증을 위하여 웹 시스템 기반 자바환경으로 구현하였고, 현장 적용한 부분을 평가 함으로써 검증하였다.

1. 서론

최근 기업의 IT 중요도는 점점 증대되고 있고 IT 가 곧 사업의 핵심 영역이며, 차별화를 통한 경쟁우위 확보의 대안으로 IT 를 활용한 비즈니스가 많이 창출되고 있다. 또한 기업의 사업영역이 IT 를 기반으로 변화되고 있다. 따라서 IT 의 중단은 곧 비즈니스의 중단을 의미하며 회사의 존폐에 영향을 미친다. 따라서 IT 아키텍쳐는 안정적인 운영, 우수한 성능 보장, 고 가용성 보장, 업무 지속성 보장등을 위해 HA 구성, 재해복구시스템 구축, 멀티 도메인 구성, 멀티 인스턴스 구성 등의 다양하고 복잡한 아키텍쳐를 구현하고 있다. 이러한 아키텍쳐의 특성을 수용하고 지원하기 위하여 응용 프로그램을 설치할 때, 사용자가 선택한 시스템 구성 요소의 조건이나 설정한 특성 등을 보관하며, 사용자가 프로그램을 시동할 때 사용하는 설정파일의 중요성이 더욱 증대되고 있다. 또한 설정파일의 유연한 형상관리가 더욱 강조되고 있는 현실이다[1].

형상관리는 IT 의 모든 형상요소를 정의, 저장하는 프로세스로서 형상관리 대상의 범위 및 통제를 담당하는 프로세스를 말한다. 조직 내부의 IT 자산 및 형상에 대한 모든 사항을 관리하고, 모든 IT 서비스 관리 프로세스를 지원하기 위한 문서와 형상에 대한 정확한 정보를 제공해 주고 있다. 이것은 정확한 정보를 검증하는 목적을 갖고 있다. 형상관리의 주요활동으로 계획, 형상식별, 형상통제, 상태 정보관리, 확인 및 감사가 있으며, 이와 같이 형상을 식별하고 변경요구를 수용, 승인, 정보추적하며 확인 및 감사를 한다[2]. 형상관리 프로세스 중 품질 보증 목적을 포함한 확인 및 감사 단계의 역할이 형상의 정합성을 증명하는데 매우 중요한 단계가 된다.

확인과 감사(Verification and audit) 활동 단계는 시스템의 요구사항에 대한 만족 여부를 검토 및 검사하고 시험하는 활동 단계로서, 다양하고 정확한 감사를 수행하여 보다 안전한 유지보수 운영과, 핵심기술에 대한 보안관리/통제가 가능한 형상관리를 할 수 있다. 하지만 많은 형상관리시스템이 어플리케이션 소프트웨어를 주 대상으로 하며, 특히 설정파일은 환경변수

* 이 연구에 참여한 연구자는 '2 단계 BK21 사업'의 지원을 받았음.
† 교신저자 : 백두권

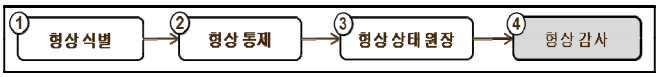
를 포함한 특징들 때문에 변경통제의 어려움과 버전 통제의 어려움이 있다. 따라서 현재 수준은 단순히 보관 기능에 한정되고, 동일한 설정파일의 그룹들 속에서 관계 있는 설정파일이 동일한 버전임을 증명하는 프로세스도 없으며, 감사하는데 한계를 갖고 있다.

이 연구에서는 기존 형상관리시스템을 이용하면서 위에서 언급된 형상감사 단계의 문제점을 보완할 수 있는 버전 무결성 감사 프레임워크를 제안하고, 동일한 형상항목이면서 일부 특성 차이를 갖는 설정파일을 비교할 수 있는 기법을 포함하여 제안한다.

이후, 2 장에서 관련연구를 통해 문제점을 도출하고 3 장에서는 문제점 해결 가능한 설정파일의 버전 무결성 검증이 가능한 감사 프레임워크를 제안한다. 4 장에서는 제시된 방안을 실제 구현하여 검증하고, 5 장에서 결론을 기술한다.

2. 관련연구

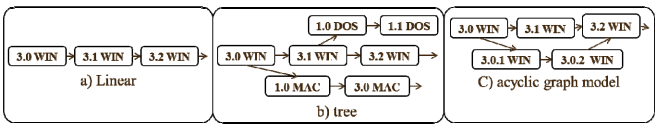
(그림 1)은 형상관리의 기본원칙을 보여준다. 형상관리의 기본원칙은 형상을 식별하고, 통제하며, 정보를 관리하며, 감사를 수행하는 활동이다[3,4].



(그림 1) 형상관리 기본원칙

①번 형상식별은 각각 개별적으로 고유한 식별이 가능한 객체를 형상항목이라 하며 하드웨어, 장비, 유형 자산, 소프트웨어 그리고 문서 등이 해당된다[5]. 하지만 형상을 관리하기 위하여 자동화 하는 과정에서 필요성이 높고 활용도가 많은 어플리케이션 소프트웨어 중심의 형상관리시스템과 Tool 들이 개발되어 보급되고 있다. 따라서 설정파일[7]과 같이 변경 통제가 어렵고 자동 추출하기 어려운 특징을 갖은 형상항목들은 형상관리 자동화 시스템에서 많은 기능이 배제되고 단순 텍스트 중심의 버전관리만 활용하고 있는 문제를 갖고 있다.

②번 형상통제는 변경요구를 받아 이를 평가하고 중요도를 판단하며 변경여부를 의사결정 하여 승인하는 일을 말한다. 실질적인 역할은 형상항목들의 기준선 설정 이후 버전관리가 주된 역할이다. 버전 관리[6]는 버전과 리비전(Revision) 으로 구성되며 버전 발생 구조는 (그림 2) 와 같다[3,8].



(그림 2) 버전통제 그래프 구조

각각 하나의 형상항목을 기준으로 변경되는 과정을 버전으로 관리하고 변경된 기록을 추적할 수 있도록 한다. 하지만 멀티 도메인, 멀티 서버, 멀티 인스턴스 환경에서 설치 배포되는 설정파일의 경우, 각각 개별적으로 버전관리가 이루어진다. 결과적으로 동일한 원본에서 배포된 파일이 환경에 관련된 특정 매개변수가 다르게 기술되어 버전이 각각 분리 관리되면서, 일정시간 경과 후 동일한 설정파일이 각기 다른 버전으로 존재하는 문제가 발생하게 된다. 또한 배포 실패나 착오 수정에 의해 특정 설정파일 수정을 누락하여 동일한 문제가 수시로 발생한다.

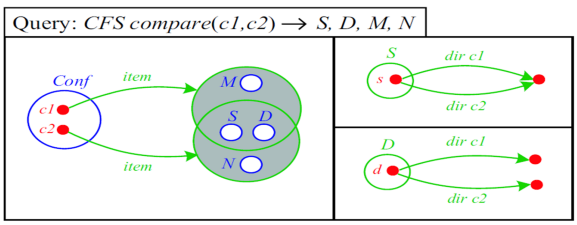
③번 형상상태 원장은 형상항목들이 어떻게 변경되 왔는지 파악할 수 있도록 형상변경에 대한 모든 정보를 추적, 기록하고 관리하는 것을 말한다. 이때 버전이 동일해야 하는 관계를 정의하고 정보를 관리하며, 버전 일치성을 판단할 수 있도록 한다. 이논문에서 설정파일을 일반화하기 위한 여러 정보들을 포함해서 관리하는 기능을 추가하도록 한다.

④번 형상감사는 변경 요청한 요구사항이 만족하는지를 검증하고 새로운 기준선을 등록하는 원칙을 말한다[4]. 또한 변경된 형상이 정상적으로 버전 일치하는지 확인하고, 나아가서 빌드(Build) 된 결과와 소스코드의 일치성을 확인하는 감사도 수행한다. 하지만 동일한 형상항목이 여러 서버 또는 인스턴스에 배포되는 경우(예:공통모듈을 처리계, 채널계, 배치계 배포) 해당 모듈들이 동일한 버전으로 존재하는지 버전 무결성 감사는 수행할 수 없으며, 설정파일과 같이 내용이 조금씩 다른 형상항목은 비교 대상에서 제외되며, 비교할 수 있는 솔루션 형태의 기술이 없는 문제가 상존한다.

3. 설정파일 버전 무결성 감사 프레임워크

3.1 설정파일 일반화 개념원리

설정파일은 정의해야 하는 내용과 해당 파일이 설치되어야 할 장소 및 환경 등의 특징을 포함해야 하는 특성이 있다. 또한 동일한 설정파일이 여러 환경에 동시에 설치되어야 하는 형상항목이기 때문에 분산 설치되는 설정파일들 속에서 공통부분과 차별화된 매개변수를 분리한다. (그림 3) 은 설정파일의 공통인자(S,D)와 한쪽에만 존재하는(M,N) 을 나타내고[7], 분리된 매개변수에서 차별화 매개변수(M,N)를 패턴 분석하여 패턴으로 정의하고 해당 패턴을 (그림 4) 의 Rule 정보에 등록한다.

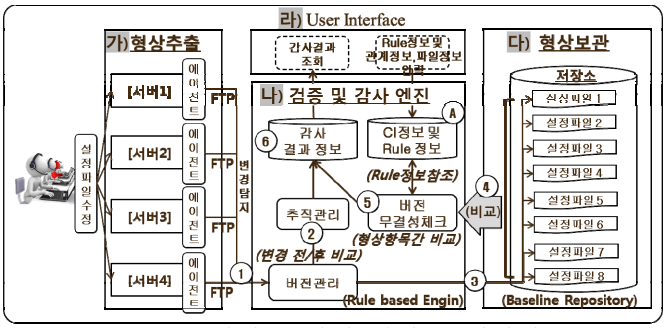


(그림 3) 설정파일 분석 기법

등록된 패턴은 설정파일들을 그룹으로 정의하여 비교할 때 동일성 여부를 판단해서 차이가 발견되어도 예외처리하기 위한 정보로 활용된다. 비즈니스 규칙을 정의하는 매개변수를 데이터베이스에 저장하고 매개변수를 이용해 규칙 처리를 하는 데이터베이스 기반의 Rule 엔진 기법을 이용하였다[9].

3.2 프레임워크 구성요소

설정파일은 일반 어플리케이션 소프트웨어와 같은 방법으로 형상관리를 수행할 수 없기 때문에 형상관리 기반으로 최적화된 감사 프레임워크를 설계하였다. 변경 내용을 탐지하여 추출하고, 추출된 내용이 기존 형상 버전과 동일한 것인지 아니면 새로운 버전을 부여해야 하는지 판단하게 된다. 변경된 내용은 새로운 기준선으로 형상 저장소에 반영하고 변경 전/후 사항을 보고서로 제공한다. 기존의 형상관리 기법을 기반으로 형상항목의 버전을 통제하고, 설정파일을 그룹으로 분류하여 관계로 맺어 해당 정보를 반영한다. (그림 4)는 프레임워크 개념도이며 중요 부분에 대해 상세한 설명을 하도록 한다.



(그림 4) 버전 무결성 감사 프레임워크

가) 형상추출은 설정파일 수정이 완료되어 저장된 설정파일을 데몬 형태의 에이전트로 상시 감시하고 있다. 감지 즉시 FTP 를 통해 버전관리 영역으로 보낸 후 접수된 설정파일을 전/후 비교하여 변경 여부를 판단 한다. 이후 변경내용을 추적관리 정보로 보내고 형상 저장소에 새로운 버전으로 저장을 한다.

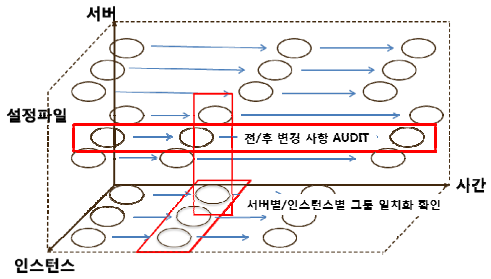
나) 검증 및 감사 엔진에서는 형상을 보관하는 설정파일의 Rule 정보를 참조하여 같은 그룹별로 버전 동일 여부를 체크하고 무결성이 위배된 설정파일의 내용을 감사결과 보고서로 제공한다. 버전 무결성 위배 건을 추출하는 엔진은 텍스트를 파싱하여 인자를 구분하고, 분리된 인자와 예외패턴 Rule 에 기 등록된 정보와 비교하는 기능이다. 중요 룰은 관계 룰과 예외패턴 룰 두 가지가 존재하며, 관계 룰은 동일한 설정파일을 의미하는 그룹을 맺어주는 관계이며 원본 성격의 설정파일이 각 환경변수만 달리하여 분산 배포되는 그룹을 정의 해 주는 것이다. 예외패턴 룰은 설정파일 원본에서 환경변수로 값을 달리 셋팅 해 주어야 하는 매개변수를 도출한 것을 말하며, 이를 패턴으로 정형화하여 패턴 룰에 등록하게 된다. 등록된 패턴 룰은 동일 그룹내 설정파일들의 동일 버전인지를 체크할 때 해당 매개변수는 비교에서 제외되도록 하는데 이용하게 된다.

다) 형상보관 장소는 DB 형태로 설계하였으며, 이미 제공된 형상관리 Tool 을 사용하여도 된다. 분산되어 등록되는 모든 설정파일 형상에 유일한 이름을 부여하여 등록하여 저장한다.

라) 인터페이스는 형상관리 대상 파일의 정보를 입력하고 대상파일들의 룰과 관계를 입력하는데 이용하며 감사 결과에 대한 보고서를 온라인을 통해 조회할 수 있다.

3.3 형상감사

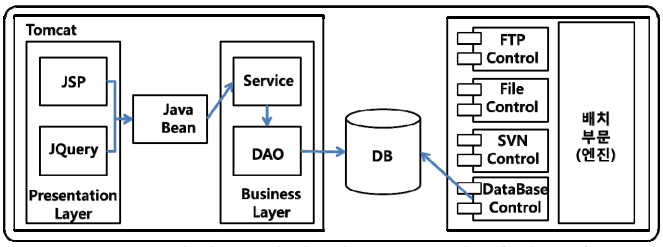
기존의 형상관리의 감사기법을 활용하여 설정파일의 각각 독립적인 버전 정합성과 변경 요구사항에 따른 변경 확인 및 감사를 수행한다. 설정파일의 동일 그룹내 설정파일들은 사전에 등록된 그룹정보를 바탕으로 그룹내 동일버전 여부를 감사한다. (그림 5) 는 제안한 기법이 시계열(과거/현재) 변경 차이를 분석하는 것과 특정시점에 동일 그룹내 설정파일들 사이의 변경 차이를 분석하여 감사할 수 있는 확장된 기능을 표현한 것이다.



(그림 5) 시계열, 객체간 입체적 다차원 감사

4. 구현 및 평가

위에서 제시된 프레임워크를 구현하기 위한 시스템 아키텍쳐는 (그림 6) 과 같다.



(그림 6) 버전 무결성 시스템 구현 아키텍쳐

웹 시스템 기반으로 프리젠테이션 레이어는 WAS 중 Tomcat 을 이용하였다. 사용자의 요청은 JSP, jQuery 에서 받아 Java Bean 에 데이터를 넣어 서비스로 요청을 하고 데이터 처리는 DAO 가 담당한다. 배치부문은 프레임워크의 핵심이며 등록된 주기별로 구동되어 설정파일을 추출하고 변경사항을 감지하여 무결성을 감사한다. 배치부문 프로그램 내에 각 역할을 수행하는 컴포넌트를 콘트롤이라 하고, 컨트롤을 통해 필요한 작업을 요청한다. FTP, File, SVN(Subversion), DB 를 담당하는 컴포넌트는 재사용이 가능하도록 설계하였다. 저장소와 Rule 정보 등을 보관하는 DB 는 오라클 DB 를 사용하며, 내부 엔진에 필요한 핵심 알고리즘은 자바로 개발하였다. 배치부문이 체크시스템의 엔진 역할을 수행하며, 텍스트를 파싱한 후 예외패턴과 비교하여 매치된 매개변수는 Space 처리 함으로써 텍스트 비교 시 제외 시키고자 하는 매개변수를 제외시킬 수 있다.

4.1 형상감사 대상 설정파일 선정

형상감사를 통해 관리해야 할 DB 설정파일, WAS 설정파일, 솔루션 설정파일 등 45 개 서버에 있는 4,500 여개의 설정파일을 선정하고, 각 파일의 디렉토리, 파일명 등 정보를 등록하였다. 등록 시 유일한 형상이름을 부여하였으며 선정된 파일을 그룹으로 분류하여 그룹관계를 확정하고 그룹정보를 형상정보에 등록하였다.

4.2 환경 구성

각 설정파일에 관련된 모든 정보를 Rule DB 에 저장하고, 설정파일의 그룹간 관계를 정의하여 등록하였으며, 각 설정파일을 최초 추출하여 저장소에 축적하여 기준선을 설정하였다. 선정된 설정파일을 분석하여 차별화 패턴을 도출하고 패턴 별 검색 조건을 정의한 후 Rule DB 에 저장하였다.

4.3 형상감사 수행

각 설정파일의 변경 발생 여부를 감사하여 변경된 내용을 도출하고 결과를 저장소에 보관하여 필요한 내용을 보고서로 제공하였다. 또한 정의된 각 그룹내부적으로 버전 상이한 건을 감사하여 각 설정파일들의 문제점을 도출하고 보고하며, 각 담당자가 원인분석 작업을 할 수 있도록 하였다. (그림 7) 은 구현되어 감사결과를 조회한 화면 사례이다.

(그림 7) 동일 그룹 내 버전 무결성 점검 결과

4.4 비교평가

기존 형상관리의 형상감사와 기능을 추가 보완한 버전 무결성 감사 기법과 비교하여 기능의 향상 여부와 운영 안정성에 대한 향상을 증명한다.

<표 1> 일반 형상감사와 무결성 형상감사 비교

구분	형상감사	무결성감사
설정파일	0 개	4,500 개 적용
위배건수	N/A	월평균 3 건
점검시간	2 시간/일(인당)	10 분/일
기능	단순비교	무결성비교
감사목적	품질보증	운영안정성

기존의 형상관리 기법은 형상의 변경에 대한 품질보증 목적으로 감사를 행한다. 기능적 감사, 물리적 감사, 프로세스 감사[10] 를 수행하는데, 이 논문에서 제안한 보완기능은 품질보증보다 수작업 업무 자동화에 따른 생산성을 개선하고, 시스템 인수 시 또는 시스템 운영 시 대량의 프레임워크 버전의 정합성을 확인하거나, 여러 파일 중 변경 누락 건을 탐지하는데 활용하여 무결성을 대폭 향상시켜 장애를 사전에 예방하는데 중점을 두었다.

5. 결론 및 향후 연구

이 연구는 소프트웨어 형상관리시스템이 지원하기 어렵고, 특징에 따라 관리대상으로 포함하기 어려운 성격의 설정파일을 형상감사할 수 있는 새로운 기법을 제안하였다.

이를 통해 장애예방과 예방점검 생산성 향상, 백업 및 복구에 활용하며, 시스템 프로그래머 또는 관리자 권한을 소유한자에 의한 변경을 탐지할 수 있게 되었다. 또한 실수에 의해 변경 누락된 설정파일 착오를 발견하여 선제적으로 장애 예방을 할 수도 있다.

향후 어느 환경에서도 신속하게 설치하고 사용할 수 있도록 유연성과 확장성에 대한 연구가 필요하며, 설정파일 추출 알고리즘이 보다 다양한 형태의 파일 형식을 수용하고, 패턴을 자동으로 판단할 수 있는 연구가 필요하다.

참고문헌

[1] http://terms.naver.com/entry.nhn?docId=846647&mobile&categoryId=2954.

[2] http://www.itsmf.or.kr/practice/support6.asp.

[3] L. Bendix, A. Dattolo, and F. Vitali. ”Software configuration management in software and hypermedia engineering: A survey” In Handbook of Software Engineering and Knowledge Engineering, volume 1, pages 523–548. World Scientific Publishing, 2001.

[4] 이원우, 권용래, "지식기반적인 소프트웨어 구성관리 시스템", 한국정보과학회, 1992 년도 봄 학술발표논문집, Vol. 19, NO. 1, 1992.

[5] CMMI for Services, Version 1.3, CMMI Product Team “Improving processes for providing better services”, Software Engineering Institute November 2010 TECHNICAL REPORT, CMU/SEI-2010-TR-034 ESC-TR-2010-034(Carnegie Mellon) , http://www.sei.cmu.edu.

[6] Ian Sommerville, “SOFTWARE ENGINEERING”, Ninth Edition, PEARSON.

[7] Howse, J., Schuman, S., Stapleton, G., “Diagrammatic formal specification of a configuration control platform”, ENTCS, 259:87-104 (2009).

[8] R. Conradi, B. Westfechtel , “Version Models for Software Configuration Management”, ACM Computing Surveys, Vol. 30, No. 2, June 1998

[9] 김성민, “BRMS(Business Rule Management System) 적용에 의한 업무프로세스 개선효과 검증”, 국민대학교 비즈니스 IT 전문대학원, 2009.

[10] Coetzee S, Cox S and Herring J., “Configuration management of a system of interdependent standards”, 7th International Conference on Standardization and Information Technology (SIIT), Berlin, Germany, 28-30 September 2011.