# 코드 이력을 이용한 버전 정보 관리 자동화에 관한 연구

김은영° 조현 최순규 이상훈
삼성전자
{marine°, hcho, skchoi, youris.lee}@samsung.com

# Study on automated version information management using code history

Eunyoung Kim° Hyun Cho Soonkew Choi Sanghoon Lee
Samsung Electronics, LTD

요 약

소프트웨어는 개발단계에서부터 유지/보수단계에 이르기까지 끊임없이 변경되며, 하드웨어와 달리 "변하기 쉬운 특성"과 "눈에 보이지 않는 특성"을 가지고 있어 소프트웨어의 형상관리를 어렵게 한다. 소프트웨어 형상 관리를 용이하게 하고 이를 통해 개발 생산성을 높이기 위해 많은 소프트웨어 하우스에서는 형상 관리 도구를 도입하여 소프트웨어 개발에 이용하고 있다. 그러나 도입된 형상 관리 도구의 활용 형태를 보면 대부분 버전 관리 부분인데 이는 개발자들이 변경 정보를 충실히 작성하지 않으면 형상 아이템의 버전만 관리될 뿐 실질적인 변경 정보는 각 버전의 비교를 통해서만 알 수 있다. 따라서 본 논문은 형상 관리 아이템중 코딩 표준에 따라 작성되는 소스 코드를 대상으로 새로운 버전이 생성될 때마다, 변경 정보를 소스 코드로부터 추출하여 자동으로 형상 관리 도구의 버전 관리 정보로 등록하여 관리함으로써 소스 코드와 형상 관리 도구의 버전 정보를 통합하여 관리하는 방안을 제시하고자 한다.

## 1. 서론

삼성전자는 휴대폰, 디지털 TV, 컴퓨터, 프린터 등과 같이 다양한 전자 제품을 생산하고 있다. 이러한 다양한 전자 제품들은 Digital Convergence 시대를 맞이 하여 하나의 제품에서 다양한 기능을 제공할 수 있도록 전자 제품들이 통합되어 가고 있다. 따라서 개발되는 제품에 탑재되는 소프트웨어의 크기 역시 새로운 제품이 출시될 때마다 기하급수적으로 증가하고 있다.

이렇게 증가하는 소프트웨어를 효율적으로 관리 하기 위하여 형상 관리 도구를 개발에 적용하고 있다. 형상 관리 도구의 도입으로 변경 프로세스가 간소화되고 개발 산출물의 버전 관리가 용이하게 되어 팀 단위 개발 작업이 가속화되었다. 그러나 소프트웨어의 버전 관리의 경우 개발자가 자신이 변경한 변경 이력을 충실히 작성하지 않는 경우 형상 관리 도구는 변경된 소스 코드의 버전만 보관하는 일종의 Backup 레파지토리로만 활용된다. 따라서 소프트웨어의 유지 보수를 위하여 변경정보가 필요한 경우, 개발자는 형상 관리 도구에 저장된 각 버전을 하나씩 비교하여 변경 내용을 추출하여 조합해야한다.

본 논문은 변경정보를 충실히 관리하는 방안으로 소스코드를 대상으로 코딩 표준과 연계하여 소스 코드로부터 변경 정보를 자동으로 추출하여 변경 정보 입력에 대한 개발자의 부담을 덜어주고 변경 정보를 일목요연하게 관리하는 방안을 제시하고자 한다.

## 2. 버전 관리의 문제점

형상 관리 도구에서 소프트웨어 버전 관리를 위한 많은 기능을 제공하고 있으나 개발자 또는 운영자가 적절한 정보를 제때 입력하지 않으면 단순히 버전을 가진 레파지토리 의미만 가질 뿐이다. 개발자들이 형상 관리 정보를, 특히 소스 코드의 버전 관리 정보를 충실히 입력하지 못하는 것은 첫째, 최초의 변경 의도와 더불어 새로운 변경 요구가 변경 과정에서 발생한 경우 즉각 형상 관리 도구에 반영할 수 없으며, 둘째 코딩 표준에 따라 소스 코드에 변경 정보를 입력하고 형상 관리 도구에 입력해야 하는 이중작업이 개발자들에게 부담으로 작용하고 있기 때문이다. 또한 전략적으로 개발과정에서는 소스 코드에 버전 정보가 존재해야 하지만 개발 완

료 후 이를 삭제하여 배포해야 되는 불편함 때문이다.

이러한 문제점으로 인해 버전 정보가 형상 관리 도구와 소스 코드 두군데 이중으로 관리되고 있으며, 정보의 이중 입력으로 인해 어느 한쪽도 버전 정보가 충분히 제공되지 못하는 경우가 발생한다.

다음의 그림1은 실제 개발중인 소스 코드의 일부분으로 소스 코드에 기록되는 변경 정보의 예이고, 그림2는 그림 1의 소스 코드의 버전 트리로 형상 관리 도구에서 제공하는 변경 정보의 예이다.

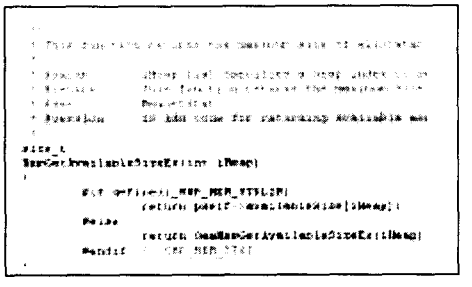

그림 1. 소스 코드의 변경 정보

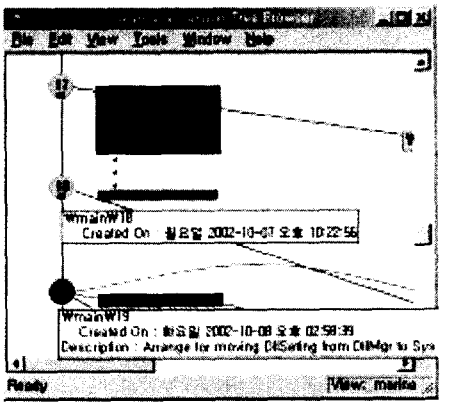

그림 2. 형상관리 도구의 변경 정보

위 그림에서 보듯이 버전 18번에 대한 변경정보는 소스코드에, 버전 19번에 대한 변경 정보는 형상 관리 도구에 기술되어져 있음을 볼 수 있으며, 두 개의 형상 관리 정보가 서로 일치하지 않음을 알 수 있다.

## 3. 버전 정보 통합 관리 시스템

위와 같은 문제를 해결하기 위해 형상 관리 도구와 소스 코드를 연계하여 소스 코드로부터 버전 정보를 추출하여 형상 관리 도구에서 버전 정보를 통합 관리하는 시스템을 제시하고자 한다.

### 코딩 표준의 개선

기존의 C 코딩 표준은 변경이 일어날 때 마다 함수 주석 부분의 Version 항목에 버전 숫자와 변경 이력을 기록하도록 규정하고 있다.

버전 정보 통합 관리 시스템을 구축하기 위하여 먼저 함수 주석 부분에 관한 코딩 표준을 변경하는데, Version 항목을 버전 숫자와 변경 내역으로 구분되어 있는 변경 정보를 자동화된 시스템에서 버전 정보를 인식 할 수 있도록 @를 붙여 @history로 변경하고, 변경 이력은 버전 숫자를 제외하고 기록하도록 변경한다. 그림 3은 코딩 표준의 변경 전후를 나타낸다.

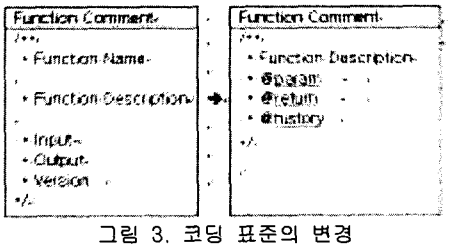


그림 3. 코딩 표준의 변경

### 버전 정보 통합 관리 시스템의 구성

소스 코드에서 @history 부분의 변경 정보를 추출하여 자동으로 형상 관리 도구의 변경 정보로 입력할 수 있도록 형상 관리 도구를 customizing한다. 특정 operation이 실행될 때마다 특정 작업을 수행할 수 있는 Trigger를 이용하여 시스템을 구성한다.

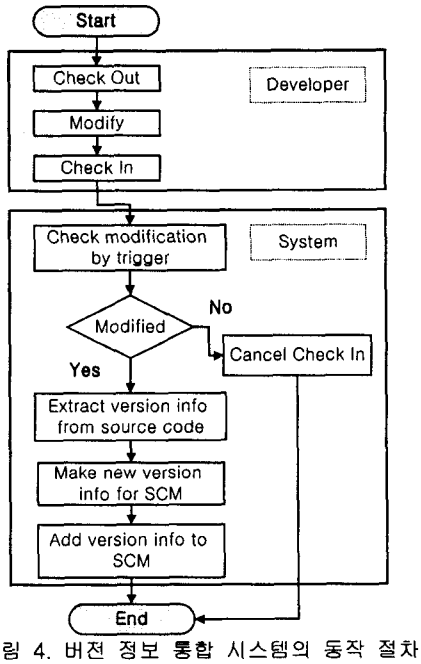


그림 4. 버전 정보 통합 시스템의 동작 절차

위 그림 4는 버전 정보 통합 관리 시스템의 동작 절차로 개발자가 형상 관리 도구에 저장된 소스 코드를 수정하는 것부터 소스 코드에서 추출된 변경 정보가

형상 관리 도구의 변경 정보로 입력되기까지의 흐름을 보여준다.

개발자는 소스 코드를 수정하기 위하여, 먼저 형상 관리 도구의 CheckOut Operation을 수행하여 소스 코드의 수정 권한을 획득한다. 권한 획득 이후, 개발자는 소스 코드를 코딩 표준에 의거하여 함수 단위로 변경하는데, 반드시 변경 정보를 함수 주석 부분중 @history 항목에 기술하여야 한다. 마지막으로 개발자는 수정된 소스 코드를 형상 관리 도구에 저장하기 위하여 CheckIn Operation을 수행한다.

이 때 시스템에서는 개발자의 CheckIn Operation을 수행하기 이전에 변경 정보를 자동으로 추출하기 위한 Trigger를 동작시킨다. Trigger는 개발자가 CheckIn 하려는 현재 버전의 소스 코드와 이미 형상 관리 도구에 저장된 이전 버전을 비교하여 소스 코드의 변경 부분을 추출한다. 추출한 변경 부분이 없다면 이 Trigger는 종료되고, 시스템은 CheckIn Operation을 취소한다. 반면 소스 코드가 변경된 경우, Trigger는 앞서 추출한 변경 부분에서 @history 패턴을 찾아 함수의 변경 정보를 분리하고, 해당 함수의 이름도 추출하여 변경 정보를 재구성한다. 이렇게 재구성된 변경 정보를 소스 코드의 수정된 부분 전체에서 추출하여 현재 버전의 변경 정보로 조합하고, 형상 관리 도구의 CheckIn Operation의 Comment로 입력하면 Trigger가 정상 종료되고 시스템은 CheckIn Operation을 허용한다.

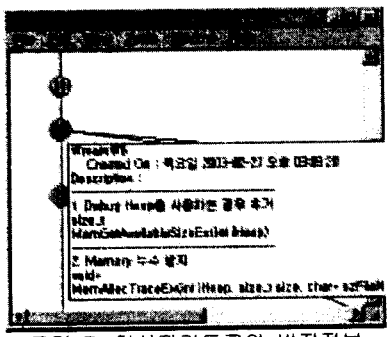

그림 7. 형상관리도구의 버전정보

형상 관리 도구에서 관리되는 버전 정보의 예인 그림 7을 보면 버전 트리에서 변경 정보와 함께 수정한 함수를 바로 볼수 있어, 개발자는 버전간의 변경 내용을 버전 비교 기능을 사용하지 않고도 일목요연하게 알 수 있게 되어 변경 정보 활용이 용이해졌다.

## 4. 결론

소스 코드의 변경 관리를 효율적으로 하기위해 코딩 표준을 준수 하는 소스 코드를 대상으로 형상 관리 도구에서 변경 정보를 자동으로 관리하도록 하였다.

코딩 표준을 강력히 준수하도록 제한하였기 때문에 상대적으로 적은 부분에 대하여 변경정보관리 자동화가 적용되었지만, 기존의 개발자 자율에 의해 변경정보를 관리하던 것보다 체계적이고 효과적인 소스코드의 변경을 관리가 이루어졌다. 개발자들은 CheckOut시 의도와 다른 부분을 수정했다 하더라도, 코딩 표준에 의해 함수 주석 부분에 변경이력만 작성해 두면 CheckIn시 소실되는 변경정보 없이 자동으로 변경정보를 형상관리도구로 관리할 수 있게 되고, 변경정보의 이중작성으로 인한 부담을 덜게 되었다.

또한 변경 정보를 추출하는 Trigger에 형상 관리 도구에 입력되는 변경 정보는 소스코드에서 삭제하도록 하는 부분을 추가함으로, 소스 코드를 타 사업부에 배포할 때 함수의 변경 정보를 삭제하기 위한 노력이 줄어들게 되었다.

## 5. 참고 문헌

[1] Rational Inc., Managing Software Projects with Base ClearCase,

[2] Fletcher J. Buckley Implementing a Software Configuration Management environment, IEEE Computer, Volume:27 Issue:2, Feb 1994

[3] Bakul Banerjee, Implementation of a SW configuration environment, Software Engineering Standards Application Workshop, 1991., Fourth, 20-24 May 1991