소프트웨어 산출물들간 변경 추적을 위한 버전 링크 식별기법

김 대 엽°, 윤 청

충남대학교 컴퓨터공학과

kdymn2@cnu.ac.kr, cyoun@cnu.ac.kr

A Method for Detecting Version Links to Trace Changes among Software Artifacts

Dae-Yeob Kim°, Cheong Youn

Dept. of Computer Engineering, Chungnam National University

요 약

소프트웨어 산출물들은 지속적으로 변경되며, 변경 이력을 관리하기 위해 버전관리 기법이 사용된다. 산출물의 버전 정보는 과거의 변경에 대한 추적성을 제공함으로써 효율적인 변경 관리를 돕는다. 여러 산출물들이 함께 변경된 경우 각 산출물들의 버전정보를 링크시킴으로써 변경에 대한 추적성을 향상시킬 수 있다. 버전 링크의 식별은 미리 정해진 산출물들의 집합을 대상으로 하며, 동일한 변경 요청에 대해서 이루어진다. 본 논문은 산출물들의 집합을 한 형상항목 내에 포함된 것으로 정하고, 형상항목에 대한 변경 요청으로부터 산출물들간의 버전 링크를 식별하기 위한 기법을 제시한다. 형상항목과 산출물의 관계로부터 버전 링크를 식별하기 위해 형상관리 환경과 산출물의 버전관리 환경을 통합하였다.

1. 서론

소프트웨어 시스템은 지속적으로 진화하며, 그 규모와 복잡성이 날로 증가한다. 여러 개선 작업이나 유지보수 활동 이후에 변경은 더욱 어려운 일이 된다. 따라서 소프트웨어 시스템을 보다 효율적으로 유지보수 할 수 있는 형태로 만들기 위한 방법이 필요하다. 소프트웨어 시스템의 진화는 그것을 구성하는 다양한 산출물들의 변경을 수반하기 때문에 산출물들의 변경을 얼마나 체계적이고 효율적으로 관리하느냐에 따라 시스템의 유지보수성이 결정된다고 볼 수 있다.

산출물의 지속적인 변경을 관리하기 위한 방법으로 버전관리 기법이 사용된다. 버전관리를 통해 개발자들은 과거의 변경 이력을 분석함으로써 변경에 대한 추적성 정보를 획득할 수 있다. 변경에 대한 추적은 단일 산출물뿐만 아니라 여러 산출물들에 대해서도 이루어져야 한다. 특정 산출물의 변경은 그것과 관계된 다른 산출물들에게 영향을 주는 경향이 있으므로, 동일 목적을 위해 변경된 여러 산출물들에 대해서 추적성을 제공하는 것은 향후 유지보수 활동에 도움이 된다.

버전 정보를 이용한 산출물들간 변경 추적 기법은 Kagdi et al. [1-3], Zimmerman et al. [4], Gall et al. [5-7] 등에 의해서 연구되었다. 이들은 산출물들간 추적성 식별을 위해 과거의 버전 이력을 분석하였으며, 이 방법이 기존의 의존성 분석에 기반한 방법에 대해 갖는 차별성을 주장하였다. 의존성 분석은 산출물들간 변경 추적성을 식별하기 위해 결합도(coupling)와 응집도(cohesion) 측정 메트릭스를 사용한다. 이러한 방법은 산출물들 사이의 구문론적(syntactic) 혹은 의미론적(semantic) 상관관계를 분석한다. 하지만 버전 이력을 고려하지 않고 단일 버전에 대해서만 추적성을 식별하기 때문에 과거의 변경 흐름을 이해하는데 한계가 있으며, 큰 규모의 시스템에 대해서 완전한 의존성 분석이 이루어지지 않을 경우 부정확한 변경이 발생할 수도 있다. 버전 이력에 기반한 추적성 식별기법은 특정 변경과 관련된 산출물들의 집합을 찾아내고 그것들간의 버전 링크를 나타냄으로써 시스템의 유지보수성을 향상시킨다. Kagdi, Zimmerman, Gall의 연구에서는 변경과 관련된 산출물들의 집합을 찾아내기 위해 산출물의 commit 시간, 변경 담당자 등에 대한 공통적 속성을 분석한다. 분석의 결과로 찾아낸 산출물들의 집합에 대해서는 논리적 결합(logical coupling) 혹은 변경 결합(change coupling)이란 표현을 사용한다. 이 경우 추적성 식별의 대상인 산출물들의 수가 가변적일 수 있다.

본 연구는 추적성 식별의 대상인 산출물들의 범위를 한 형상항목에 포함된 것으로 정의한다. 소프트웨어 형상관리(Software Configuration Management)에서 한 형상항목은 여러 개의 산출물들을 포함할 수 있다[8]. 형상항목에 포함된 산출물들은 특정 변경 요청에 대해 각각 다른 흐름의 버전들을 가질 수 있으며, 그 결과 형상항목이 공식화될 때 서로 다른 버전이 형상관리 서버에 반영될 수 있다. 한 형상항목 내에서 여러 산출물들이 서로 다른 버전으로 반영된다면, 산출물들간 버전의 링크를 통해 변경에 대한 추적성을 나타낼 필요가 있다. 형상항목이 기준선 문서로서 형상관리

서버에 공식화될 때 포함된 산출물들의 버전 링크를 구성함으로써 향후 변경에 대한 기준을 제시할 수 있으며, 이는 곧 유지보수 활동에 도움이 될 수 있다. 한 형상항목에 포함된 산출물들의 버전 링크를 나타내기 위해 제안한 시스템은 형상항목의 수정버전(revision)과 포함된 산출물들의 버전을 연결시킨다. 형상항목의 수정버전은 소프트웨어 시스템의 특정 릴리즈를 구성하는 베이스라인의 요소로서, 형상항목이 기준선 문서로 공식화될 때 증가한다. 이 정보를 포함된 산출물들의 버전에 태그로 부착함으로써 산출물들의 버전 링크를 나타낼 수 있다. 형상항목에 대한 변경 요청과 변경 후 공식화 과정까지의 모든 절차는 형상관리 시스템을 통해 이루어지며, 형상항목 내에 포함된 산출물들에 대한 변경 업무는 개별적인 버전 관리를 지원하는 협업관리 시스템을 통해 이루어진다. 따라서 형상항목의 수정버전과 산출물들의 버전을 연결하기 위해서는 형상관리 시스템과 협업관리 시스템을 통합해야 한다. 통합 환경에서 형상항목의 공식화된 변경 절차와 산출물들의 개별적인 변경이 유기적으로 연계될 수 있다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 형상항목의 수정버전과 산출물들의 버전을 연결하기 위한 통합 변경관리 시스템 구조를 설명한다. 3장에서는 통합 변경관리 시스템의 구현 결과를 통해 산출물들간 버전 링크를 시각적으로 나타내는 과정을 설명한다. 마지막으로 4장에서 본 논문의 결론과 향후 연구를 논한다.

2. 통합 변경관리 시스템 구조

본 장에서 소개하는 통합 변경관리 시스템은 형상항목의 변경과 그것을 구성하는 산출물들의 변경을 연계함으로써 산출물들의 버전 링크를 식별할 수 있도록 지원한다. 형상항목의 변경은 형상관리 시스템에서 이루어지며, 산출물들의 변경은 개인 작업환경에서 협업관리 시스템의 지원을 받아 이루어진다.

형상항목은 기준선 문서로서 형상관리 시스템에 등록되며, 형상항목을 변경하기 위해서는 공식화된 결재 절차를 거쳐야만 한다. 형상항목의 변경 결과가 형상관리 시스템에 반영되면 기준선 문서로서 새로운 공식화 버전이 생성되며 이것을 수정버전(revision)이라 부른다.

기준선 문서를 관리하는 SCM 시스템과는 달리 개인의 작업환경에서는 각 개발자의 의도에 따라 자유로운 변경이 이루어질 수 있다. CVS나 Subversion과 같은 버전관리 도구들은 이러한 개인 작업환경을 지원하기 위한 것으로 산출물들의 버전뿐만 아니라 여러 개발자들의 협업을 관리할 수 있도록 한다. 개인 작업환경에서 산출물들은 형상항목의 공식화 상태와 상관없이 지속적인 변경이 이루어질 수 있으므로 형상관리 시스템에 반영된 버전과 임의로 생성된 버전들을 구분할 수 있어야 한다. 임의로 생성된 버전과 공식화된 버전을 구분하는 이유는 기준선 문서와 연결된 산출물들의 버전 링크를 식별하기 위함이다. 형상항목에 포함된 산출물들의 버전 링크는 특정 수정버전에 해당되는 산출물들의 변경 이력을 추적할 수 있도록 하기 때문에 향후 유지보수 활동에 도움을 줄 수 있다.

본 논문은 형상항목의 수정버전으로부터 포함된 산출물들의 버전 링크를 식별하기 위해 그림 1과 같은 구조의 통합 변경관리 시스템을 제안한다.

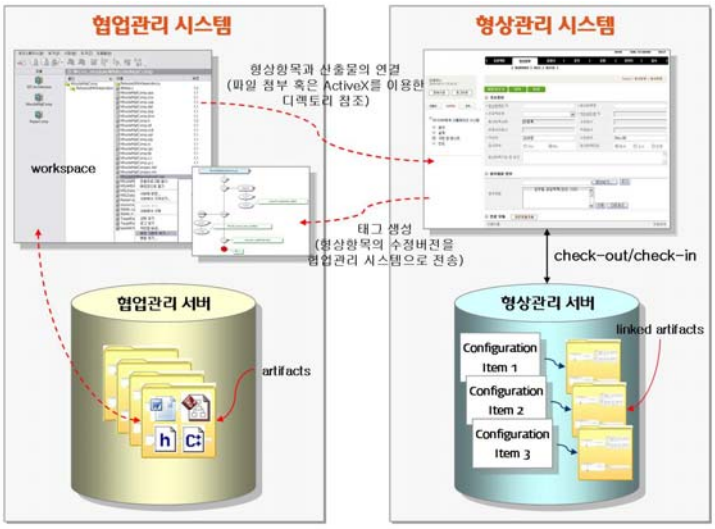

그림 1 협업관리와 형상관리의 상호 운용

그림 1은 개인 작업환경을 지원하는 협업관리 시스템과 형상관리 시스템이 통합 운용되는 모습을 도식화한 것이다. 협업관리 시스템은 산출물의 버전관리 기능은 물론 여러 개발자들의 병렬 개발을 지원하기 위한 기능들을 포함하고 있다(공개 소프트웨어인 CVS를 기반으로 구현). 형상관리 시스템은 웹 기반으로 구현되었으며, 형상항목에 대한 변경 통제 기능을 포함하여 프로세스 관리 기능, 이슈 보고 기능 등을 제공하고 있다.

협업관리 시스템은 워크스페이스(workspace)라고 하는 개인 작업환경을 통해 버전관리 및 병렬개발을 지원한다. 워크스페이스에서 산출물들은 모듈(module)이라고 하는 단위로 그룹화되고, 모듈은 협업관리 서버에 저장된다. 개발자들은 워크스페이스를 통해 협업관리 서버로부터 모듈을 다운로드하기도 하며 자신의 작업 결과를 서버에 반영(모듈 혹은 개별 산출물에 대해서 모두 가능)하기도 한다. 이러한 과정을 반복함으로써 각 산출물들의 버전이 지속적으로 증가하게 되며 그 결과는 버전 그래프(version graph)로 표현된다.

협업관리 시스템에서 생성된 산출물들의 집합을 형상관리 시스템에 반영하기 위해서는 그룹화 단위인 모듈을 형상항목에 연결시키는 과정이 필요하다. 모듈과 형상항목의 연결은 웹 기반의 형상관리 시스템에서 워크스페이스의 물리적 디렉토리를 참조하는 방식으로 이루어진다. 모듈과 형상항목이 연결된 상태에서 해당 형상항목이 기준선 문서로 공식화되는 과정(check in)을 거치면 형상항목의 수정버전이 모듈 내에 포함된 산출물들의 버전에 태그(tag)로 연결된다. 태그는 특정 버전에 대해 간략한 정보를 제공하기 위한 목적으로 생성된다. 태그는 개발자가 원하는 작업을 수행한 후 특정 버전에 임의로 생성할 수도 있지만, 형상항목의 수정버전 태그는 형상항목이 공식화되는 시점에 시스템이 자동으로 생성하여 산출물의 버전에 연결한다. 이것은 개발자가 일일이 모든 산출물의 버전에 형상항목의 수정버전 태그를 생성하는 부담을 덜어준다.

형상항목의 수정버전이 형상관리 시스템에 반영된 각 산출물 버전에 태그로 연결되면 개발자들은 공식화된 버전들의 링크를 확인할 수 있으며, 이를 통해 산출물들의 변경 추적성을 식별할 수 있게 된다. 다음 장에서는 형상항목의 수정버전과 산출물들의 버전이 연결된 모습을 보여주고, 그로부터 얻을 수 있는 산출물들의 버전 링크 및 추적성 정보에 대해 설명한다.

3. 산출물들간 버전 링크의 시각화와 추적성 식별 방법

산출물들의 버전 흐름은 버전 그래프로 표현된다. 버전 그래프를 통해 개발자들은 현재까지 진행된 산출물들의 변경 흐름을 파악할 수 있으며 필요할 경우 새로운 분기(branch)를 생성할 수도 있다. 분기는 기존 변경 흐름으로부터 새로운 목적의 변경 흐름을 진행할 필요가 있을 때 생성한다. 그림 2는 산출물에 대한 버전 그래프의 예를 보여준다.

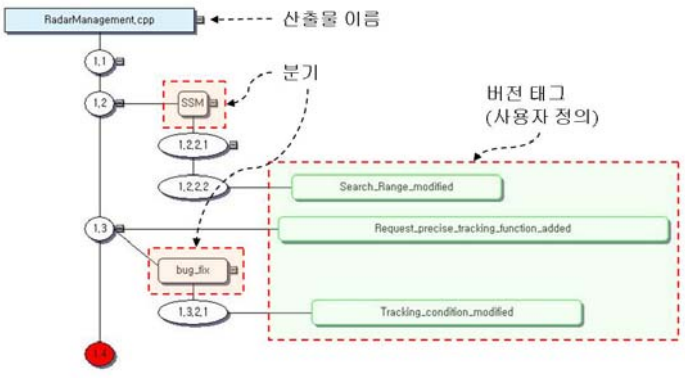

그림 2 산출물의 버전 그래프 예

위의 그림에서 버전 '1.2'와 1.3'에 각각 다른 목적의 분기를 생성한 모습을 볼 수 있으며, 특정 버전('1.2.2.2', '1.3', '1.3.2.1')에 사용자가 정의한 태그가 부착된 모습을 볼 수 있다. 분기에 의해 생성된 버전은 필요할 경우 다시 기본 흐름의 버전으로 병합(merge)될 수도 있다. 위와 같은 버전 그래프는 각 산출물마다 다르게 나타날 수 있으며, 따라서 모든 산출물의 변경 추적성을 나타내기 위해서는 버전들간 링크를 식별할 수 있는 방법이 필요하다. 이를 위해 본 연구에서 제시한 방법은 형상항목의 수정버전을 각 산출물의 버전(형상관리 시스템에 반영된 버전)에 태그로 부착하는 것이다. 그림 3은 특정 산출물에 형상항목의 수정버전이 태그로 부착된 모습을 보여준다.

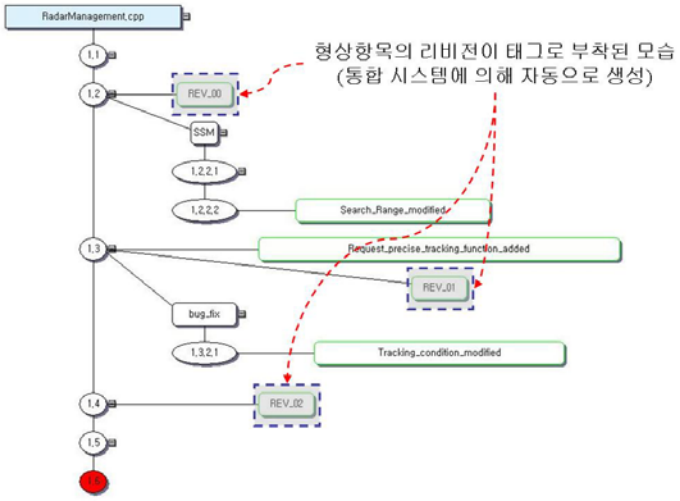

그림 3 산출물의 버전 그래프 (형상항목의 수정버전이 태그로 부착된 경우)

그림 3에서 볼 때, 형상관리 시스템에 반영된 버전은 '1.2', '1.3', '1.4'이며 이들은 각각 형상항목의 수정버전을 나타내는 'REV_00', 'REV_01', 'REV_02'와 연결되어 있다. 형상항목의 수정버전 태그는 2장에서 설명한 바와 같이 시스템이 자동적으로 생성, 연결한 것이며, 이것은 그림 2의 사용자 정의에 의해 생성된 태그와 구별된 개념이다.

개발자들은 형상항목에 포함된 산출물들에 대해서 공통의 수정버전 태그를 갖는 버전들끼리 링크를 구성하고 있음을 확인할 수 있으며, 이를 통해 산출물들간 변경 추적성을 식별할 수 있다. 개발자들은 또한 특정 수정버전 정보를 이용해서 형상항목에 포함된 산출물들의 버전을 획득할 수도 있다. 워크스페이스 환경에서 제공하는 이 기능은 일일이 모든 산출물들의 버전 그래프를 확인하지 않고도 특정 수정버전과 연결된 버전들을 획득할 수 있도록 함으로써 버전 링크 및 추적성 식별을 위해 드는 시간적 비용을 줄일 수 있다.

표 1은 형상항목의 수정버전으로부터 얻을 수 있는 산출물들의 버전 링크를 개념적으로 보여준다.

표 1 형상항목 수정버전과 포함 산출물들 버전의 관계

artifacts / CI's revision	a_1	a_2	a_3	a_4
REV_00	1.1	1.2	1.2	1.3
REV_01	1.2	1.4	1.4	1.5
REV_02	1.4	1.5	1.5	1.7

표 1의 예에서 형상항목 CI에는 네 개의 산출물 a_1, a_2, a_3, a_4가 포함되어 있다. 형상항목은 공식화 과정에 따라 순차적으로 수정버전을 생성하며 그 결과는 'REV_00', 'REV_01', 'REV_02'이다. 각 수정버전은 포함된 산출물들의 버전과 연결된다. 'REV_01'의 경우 산출물 a_1에서 a_4까지 각각 1.2, 1.4, 1.4, 1.5 버전과 연결되어 있음을 알 수 있다. 다시 말해 산출물 a_1, a_2, a_3, a_4은 'REV_01'에 대해서 <1.2, 1.4, 1.4, 1.5>와 같은 버전 링크를 구성하고 있으며, 이 버전 링크를 통해 산출물들간의 변경 추적성을 식별할 수 있는 것이다.

표 1은 형상항목의 수정버전과 포함 산출물들의 버전 관계를 통해 형상항목 내에 포함된 산출물들의 버전 링크를 식별할 수 있다는 것을 개념적으로 보여준 것이다. 실제로 본 연구에서 제안한 시스템은 표 1과 같은 형태로 버전 링크를 표현하지는 않는다. 대신 협업관리 시스템의 워크스페이스 환경에서 개발자가 특정 수정버전에 연관된 버전 링크를 획득하고자 하는 경우, 수정버전을 통한 질의(query)를 통해 해당 수정버전과 연관된 버전들을 협업관리 서버로부터 가져올 수 있는 기능을 제공한다.

그림 4는 수정버전 'REV_01'과 연관된 산출물들의 버전을 가져오기 위한 질의 화면을 보여준다. 이 기능을 통해 형상항목에 포함된 산출물들에 대해서 'REV_01'이 태그로 연결된 버전들을 획득할 수 있다.

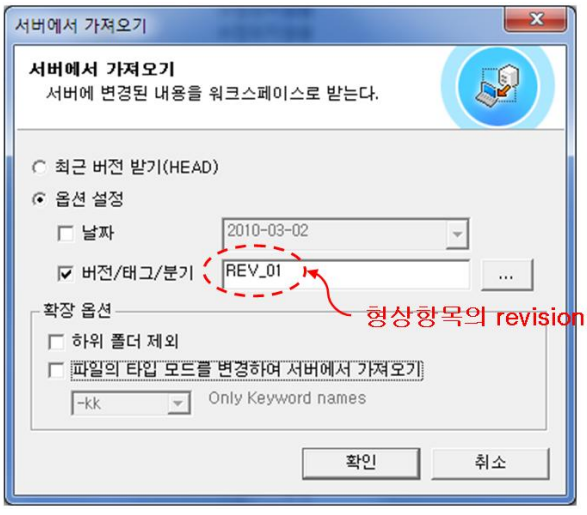

그림 4 수정버전을 이용한 산출물들의 버전 획득

위의 기능을 실행함으로써 개발자들은 특정 수정버전과 연결된 버전들의 링크를 식별하고, 산출물들간 변경 추적성을 파악할 수 있다. 그림 5는 그림 4에 나타난 화면의 처리 결과를 보여준다.

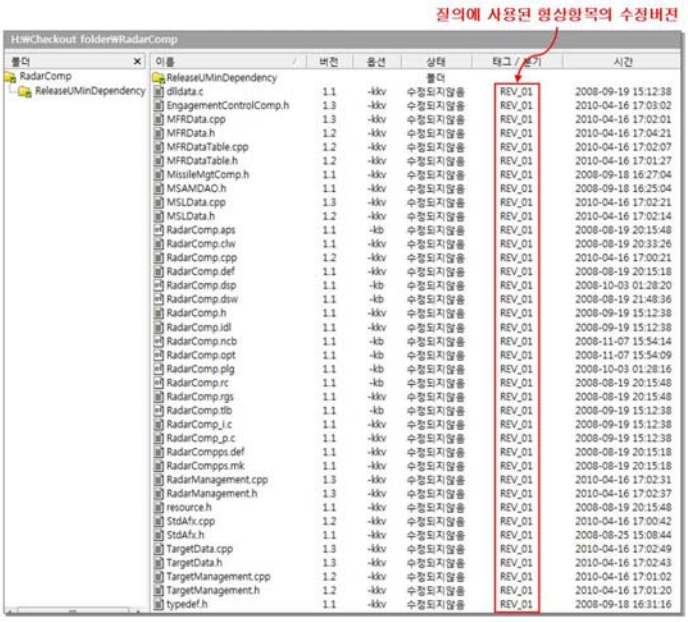

그림 5 형상항목의 수정버전을 이용한 버전 링크 획득 화면

그림 5에서 각 산출물들의 버전은 형상항목의 수정버전인 'REV_01'을 이용해 획득한 것들이다. 형상항목의 공식화와 함께 형상관리 시스템에 반영된 버전들이 각각 다르다는 것을 확인할 수 있으며, 이 정보를 통해 형상항목의 특정 수정버전과 산출물들의 버전이 어떻게 연결되어 있는지를 확인할 수 있다.

산출물과 형상항목은 다양한 변경 요청에 의해 지속적인 변경이 이루어지므로, 형상항목이 공식화될 때 산출물들의 어떤 버전들이 반영되는지를 확인할 수 있어야 향후 유지보수 활동에 어려움이 없게 된다. 본 연구에서 제시한 통합 변경관리 환경과 버전 링크 획득 방법은 과거의 변경에 대한 산출물들의 변경 패턴 및 논리적 연관성을 제공하므로 소프트웨어 시스템의 지속적인 변경에 효과적으로 대처할 수 있도록 지원한다.

4. 결론 및 향후 연구

산출물들의 버전 링크를 통한 추적성 정보는 과거의 변경에 대한 이해를 제공하고 향후에 일어날 변경에 대해 기준을 제공하기 때문에 시스템의 유지보수 활동에 유용하게 활용될 수 있다. 본 논문은 산출물들의 버전 링크 및 추적성 정보를 한 형상항목 내에서 식별할 수 있도록 하는데 초점을 맞추었다. 산출물들간 버전 링크는 형상항목의 수정버전을 태그로 연결시킴으로써 획득할 수 있으며, 이러한 과정을

시스템적으로 자동화하여 개발자들의 편의를 더했다.

향후 연구에서는 형상항목을 통한 시스템 전체의 기준선(baseline)을 식별하고, 그로부터 시스템을 구성하는 모든 산출물에 대해 추적성 정보를 제공하고자 한다. 시스템 전체를 구성하는 산출물들은 그 형태나 표현 방법들이 매우 다양하므로, 의존성 분석과 같은 방법으로 추적성 정보를 표현하기에는 많은 어려움이 존재한다. 따라서 특정 추상화 단계에 해당되는 산출물들을 그룹화하고 이를 형상항목과 연결하여 관리한다면 시스템 전반에 걸친 추적성 정보를 효율적으로 유지 관리할 수 있을 것이다.

참고문헌

[1] H. Kagdi and J. I. Maletic, “Software-Change Prediction: Estimated + Actual”, 2nd International IEEE Workshop on Software Evolvability, pp. 38-43, 2006.

[2] H. Kagdi, S. Yusuf and J. I. Maletic, “Mining Sequences of Changed-files from Version Histories”, Proceedings of the International Workshop on Mining Software Repositories, pp. 47-53, 2006.

[3] H. Kagdi, J. I. Maletic and B. Sharif, “Mining Software Repositories for Traceability Links”, International Conference on Program Comprehension, pp. 145-154, 2007.

[4] T. Zimmerman et al., “Mining Version Histories to Guide Software Changes”, IEEE Transactions on Software Engineering, Vol. 31, pp. 429-445, 2005.

[5] H. Gall, M. Jazayeri and J. Krajewski, “CVS Release History Data for Detecting Logical Couplings”, Proceedings of the International Workshop on Principles of Software Evolution, pp. 13-23, 2003.

[6] H. Gall, K. Hajek and M. Jazayeri, “Detection of Logical Coupling Based on Product Release History”, Proceedings of International Conference on Software Maintenance, pp. 190-198, 1998.

[7] B. Fluri, H. C. Gall and M. Pinzger, “Fine-Grained Analysis of Change Couplings”, IEEE International Workshop on Source Code Analysis and Manipulation, pp. 66-74, 2005.

[8] C. Schwaber, “The Forrester Wave™: Process-Centric Software Configuration Management, Q4 2005”, Forrester Research, Inc., Nov. 14 2005.