소프트웨어 산출물들간 논리적 결합 식별을 위한 버전관리 기법

김대엽*, 윤 청**
*, **충남대학교 컴퓨터공학과
e-mail : kdymn2@cnu.ac.kr

Version Management Technique for Detecting Logical Coupling among Software Artifacts

Dae-Yeob Kim*, Cheong Youn**
*, ** Dept. of Computer Engineering, Chung-Nam National University

요 약

산출물들의 논리적 결합은 소프트웨어의 다양한 유지보수 활동에서 유용하게 활용될 수 있다. 버전 이력은 과거의 변경을 분석하여 산출물들간의 논리적 결합을 판단하는데 사용된다. 본 논문은 한 형상항목에 포함된 산출물들의 버전 링크를 구성함으로써 논리적 결합을 찾아내고, 이를 통해 과거의 변경에 대한 추적성과 향후의 변경에 대한 기준을 제공할 수 있도록 하였다. 형상관리 시스템과 개인 작업환경을 통합함으로써 산출물들간 버전 링크를 구성하고 논리적 결합을 식별할 수 있도록 하였다.

키워드 – 논리적 결합; 버전 관리; 소프트웨어 형상관리(SCM); 워크스페이스

1. 서론

소프트웨어 산출물들의 논리적 결합(logical coupling)은 소프트웨어 시스템에 대한 변경의 예측(change prediction)[1, 2]과 릴리즈 관리(release management), 역 공학(reverse engineering)[3, 4] 등의 분야에서 유용하게 활용될 수 있다. 일반적으로 결합은 산출물들의 의존성 관계를 식별하기 위해 측정되었다[5, 6]. 주로 소스코드 산출물에 대해서 적용되는 이 방법은 프로시져 호출(procedure call)이나 포함(include)과 같은 관계를 통해 의존성을 식별한다. 이러한 방법은 큰 규모의 소스코드를 분석해야 하는 경우 많은 비용 부담이 따를 수 있으며, 구문론적(syntactic) 의존성에 기반하기 때문에 자칫 모든 의존성 정보를 식별하지 못할 수도 있다[7, 8]. 게다가 이러한 방법은 산출물들의 과거 변경 이력을 고려하지 않고 단일 버전만을 분석하기 때문에 버전 정보에 기반한 변경 패턴 추적 및 변경 예측에 있어서는 한계를 지니고 있다.

버전 이력에 기반한 산출물들의 논리적 결합 측정은 위의 방법과는 달리 과거의 버전 정보들을 분석하여 산출물들의 변경 패턴을 추적하고 향후에 있을 변경을 예측할 수 있도록 한다. 이러한 방법은 과거의 사건에 대한 패턴을 분석한다는 차원에서 데이터 마이닝(data mining) 기법과 유사하기 때문에 흔히 버전 이력 마이닝(mining version histories) 혹은 소프트웨어 저장소 마이닝(mining software repositories) 기법으로 불린다[2, 9-11]. 소프트웨어 저장소는 주로 CVS 나 Subversion 과 같은 버전관리 시스템을 의미한다[10]. 버전 이력 마이닝과 소프트웨어 저장소 마이닝은 그러한 의미에서 같은 성격의 기법으로 해석될 수 있다.

버전 이력에 기반한 논리적 결합은 시스템이 개발되는 동안 산출물들이 얼마나 자주 함께 변경되었는지를 측정함으로써 식별된다[1, 2, 7-13]. 산출물들이 함께 변경되었다는 사실은 소프트웨어 저장소로의 반영 시간(commit time (interval)), 반영 담당자(committer) 등에 대한 공통 요소를 분석함으로써 추측할 수 있다[9, 10, 16]. 함께 변경된 산출물들은 특정 변경 패턴으로 인식되어 그룹화되며, 그룹화 결과는 CVS 와 같은 버전관리 시스템에 적용되는 변경 집합(change-set)과 같은 의미를 가질 수 있다[9].

위와 같은 기법은 주로 얼마나 많은 산출물들이 논리적 결합을 갖고 있는지를 식별하는데 초점을 맞추고 있다. 본 논문은 산출물들의 논리적 결합을 정의함에 있어 소프트웨어 형상항목과 관련시킨다. 일반적인 소프트웨어 형상관리(Software Configuration Management, 향후 SCM 이라 칭함) 시스템들은 하나의 형상항목에 여러 개의 소프트웨어 산출물들을 포함시킬 수 있도록 허용한다[17]. 하나의 형상항목에 포함되는 산출물들은 SCM 계획 단계의 형상항목 식별 과정에서 정의된다[18]. 한 형상항목에 포함된 산출물들은 형상항목이 공식화될 때 동시에 SCM 시스템에 반영되기 때문에 앞서 설명한 논리적 결합의 조건을 만족한다고 볼 수 있다.

본 연구에서는 소프트웨어 형상항목과 산출물들을 포함 관계로 정의하고, 한 형상항목의 범위 안에서

산출물들의 논리적 결합을 식별하는데 초점을 맞추었다. 한 형상항목 내에서 산출물들은 각각 다른 버전 흐름에 의해 개발될 수 있으므로 버전 링크에 기반한 논리적 결합을 나타낼 수 있어야 한다. 제안하고자 하는 시스템은 형상항목의 공식화 버전(revision)과 그 안에 포함된 산출물들의 버전(version)을 구분하고, 형상항목이 기준선 문서(baselined document)로 공식화될 때 해당 revision 정보를 산출물의 version 에 연결시키도록 하였다. 동일 revision 으로 연결된 version 들을 획득하면 형상항목에 포함된 산출물들의 논리적 결합을 식별할 수 있다. 이러한 과정을 시스템적으로 자동화하고, 산출물의 버전 흐름 및 revision 연결 상태를 시각화함으로써 사용자의 이해와 편의를 제공하였다.

본 논문의 구성은 다음과 같다. 1 장의 서론에 이어 2 장에서는 형상항목과 산출물의 버전을 연계하기 위한 통합 변경 관리 시스템 구조를 설명하고, 3 장에서는 버전 정보를 이용한 논리적 결합의 식별 방법을 소개한다. 마지막으로 4 장에서 본 논문의 결론과 향후 연구를 논한다.

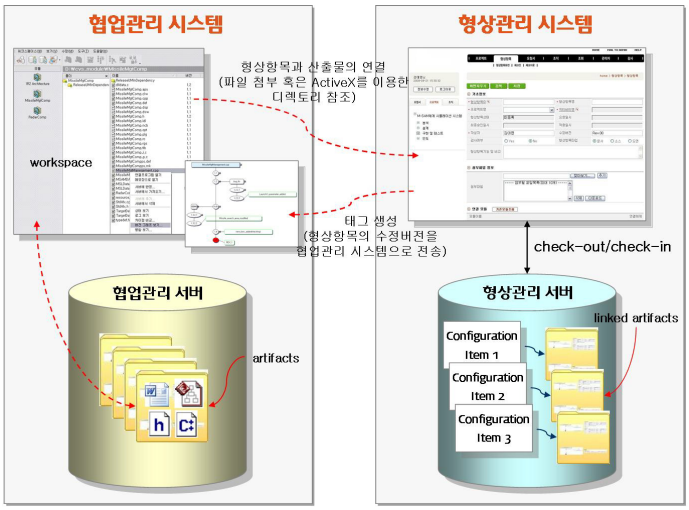
2. 통합 변경관리 시스템 구조

본 장에서 소개하는 통합 변경관리 시스템은 형상항목의 변경과 그것을 구성하는 산출물들의 변경을 연계함으로써 산출물들의 버전 링크를 구성하고 논리적 결합을 식별할 수 있도록 지원한다. 형상항목의 변경은 SCM 시스템에서 관리되며, 산출물들의 변경은 개인 작업환경에서 버전관리 및 협업관리 기법에 의해 관리된다.

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

기준선 문서를 관리하는 SCM 시스템과는 달리 개인의 작업환경에서는 각 개발자의 의도에 따라 자유로운 변경이 이루어진다. CVS 나 Subversion 과 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 가능하다. 따라서 SCM 에 반영된 공식화 버전(형상항목의 revision 이 아님)과 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 revision 에 해당되는 산출물들의 변경 이력을 추적할 수 있도록 하기 때문에 향후 유지보수 활동에 도움을 줄 수 있다.

본 논문은 형상항목의 revision 으로부터 포함 산출물들의 버전 링크를 식별하기 위해 (그림 1)과 같은 구조의 통합 변경관리 시스템을 제안한다.



(그림 1) 협업관리와 SCM 의 상호운용

(그림 1)은 개인 작업환경을 지원하는 협업관리 시스템과 SCM 시스템이 통합 운용되는 모습을 도식화한 것이다. 협업관리 시스템은 산출물의 버전관리 기능은 물론 여러 개발자들의 병렬 개발을 지원하기 위한 기능들을 포함하고 있다(공개소프트웨어인 CVS 을 기반으로 제작). SCM 시스템은 웹 기반으로 제작되었으며, 형상항목에 대한 변경 통제 기능을 포함하여 프로세스 관리 기능, 이슈 보고 기능 등을 제공하고 있다.

협업관리 시스템은 워크스페이스(workspace)라고 하는 개인 작업환경을 제공하여 버전관리 및 협업관리를 지원한다. 워크스페이스에서 산출물들은 모듈(module)이라고 하는 단위로 그룹화되고, 모듈은 협업관리 서버에 저장된다. 개발자들은 워크스페이스를 통해 협업관리 서버로부터 모듈을 다운로드하기도 하며 자신의 작업 내용을 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 나타난다.

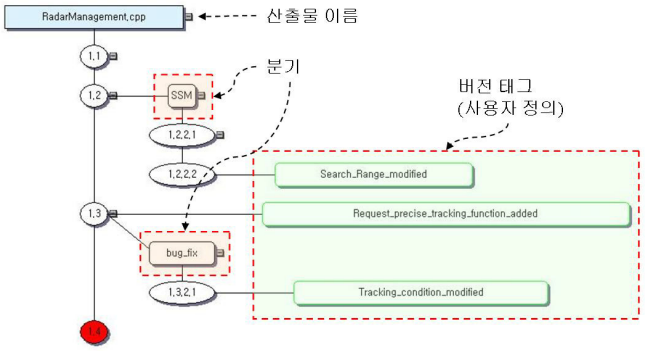
협업관리 시스템에서 생성된 산출물들의 집합을 SCM 시스템에 반영하기 위해서는 그룹화 단위인 모듈을 형상항목에 연결시키는 과정이 필요하다. 모듈과 형상항목의 연결은 웹 기반의 SCM 시스템에서 워크스페이스의 물리적 디렉토리를 참조하는 방식으로 이루어진다. 모듈과 형상항목이 연결된 상태에서 해당 형상항목이 기준선 문서로 공식화되는 과정(check in)을 거치면 형상항목의 revision 이 모듈 내에 포함된 산출물들의 버전에 태그(tag)로 연결된다. 태그는 특정 버전에 대해 간략한 정보를 제공하기 위한 것으로 개발자가 임의로 생성할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

형상항목의 revision 이 SCM 시스템에 반영된 각

산출물 버전에 태그로 연결되면 개발자들은 공식화된 버전들의 링크를 식별할 수 있으며, 이를 통해 산출물들의 논리적 결합을 획득하게 된다. 다음 장에서는 형상항목의 revision 과 산출물들의 버전이 연결된 모습을 보여주고, 그로부터 얻을 수 있는 산출물들의 논리적 결합에 대해 설명한다.

3. 버전 정보를 이용한 산출물들의 논리적 결합 식별

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

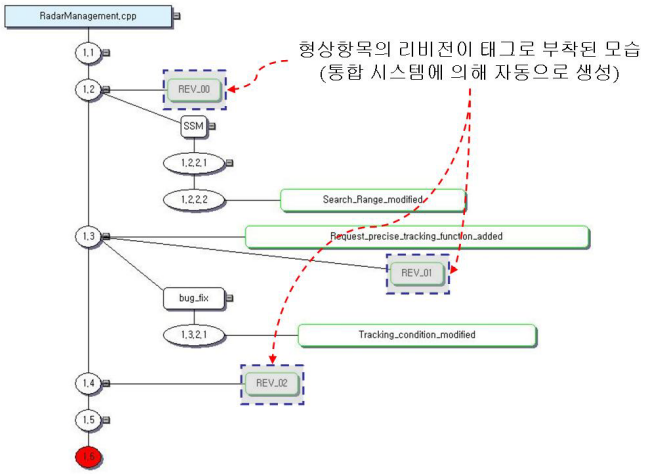


(그림 2) 산출물의 버전 그래프

위의 그림에서 산출물의 '1.2'와 '1.3' 버전에 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 병합될 수도 있다. 위와 같은 버전 그래프는 각 산출물마다 다르게 나타날 수 있으며, 따라서 모든 산출물의 변경 연관성을 추적하기 위해서는 버전들간 링크를 나타낼 수 있는 방법이 필요하다. 이를 위해 본 연구에서 제시한 방법은 형상항목의 revision 을 각 산출물의 버전(SCM 시스템에 반영된 버전)에 태그로 부착하는 것이다. (그림 3)은 특정 산출물에 형상항목의 revision 이 태그로 부착된 모습을 보여준다.

(그림 3)에서 볼 때, SCM 시스템에 반영된 버전은 '1.2', '1.3', '1.4'이며 이들은 각각 형상항목의 revision 을 나타내는 'REV_00', 'REV_01', 'REV_02'와 연결되어 있다. 형상항목의 revision 태그는 2 장에서 설명한 바와 같이 시스템이 자동적으로 생성, 연결한 것이며, 이것은 (그림 2)에서 나타난 사용자 정의에 의해 생성된 태그와는 다른 성격을 지닌다.

개발자들은 형상항목에 포함된 산출물들에 대해서 공통의 revision 태그를 갖는 버전들끼리 링크를 구성하고 있음을 파악할 수 있으며, 이를 통해 산출물들간 논리적 결합을 찾을 수 있다.



(그림 3) 산출물의 버전 그래프 (형상항목의 revision 이 태그로 부착된 모습)

개발자들은 특정 revision 정보를 이용해서 형상항목에 포함된 산출물들의 버전을 획득할 수도 있다. 워크스페이스 환경에서 제공하는 이 기능은 일일이 모든 산출물의 버전 그래프를 확인하지 않고도 특정 revision 과 연결된 버전들을 획득할 수 있도록 함으로써 버전 링크 및 논리적 결합의 식별을 위해 드는 노력을 줄일 수 있다.

<표 1>은 형상항목의 revision 으로부터 얻을 수 있는 산출물들의 버전 링크 정보를 예로 보여준다.

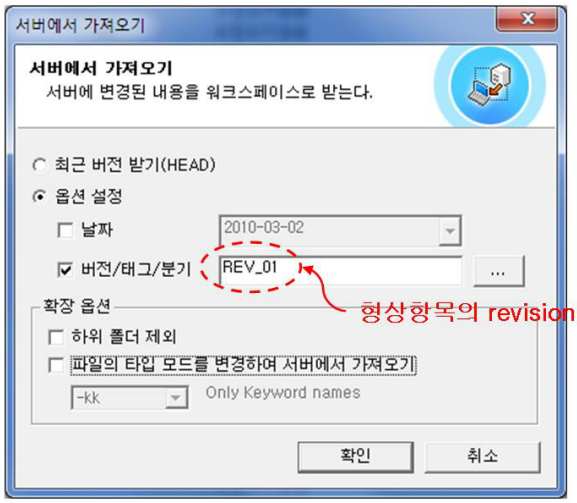
<표 1> 형상항목 revision 과 포함 산출물들의 버전 관계

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 1>의 예에서 형상항목 CI 에는 네 개의 산출물 a_1, a_2, a_3, a_4 이 포함되어 있다. 형상항목은 공식화 과정에 따라 순차적으로 revision 정보를 생성하며 그 결과는 'REV_00', 'REV_01', 'REV_02'이다. 각 revision 은 포함된 산출물들의 버전과 연관되어 있다. 'REV_01'의 경우 산출물 a1 에서 a4 까지 각각 1.2, 1.4, 1.4, 1.5 버전과 연관되어 있음을 볼 수 있다. 다시 말하면 산출물 a_1, a_2, a_3, a_4 은 'REV_01'에 대해서 <1.2, 1.4, 1.4, 1.5>와 같은 버전 링크를 구성하고 있으며, 이 버전 링크를 통해 산출물들의 논리적 결합을 식별할 수 있는 것이다.

<표 1>은 형상항목의 revision 과 포함 산출물들의 버전 정보 관계를 통해, 형상항목 내에 포함된 산출물들의 논리적 결합을 식별할 수 있다는 것을 직관적으로 나타낸 것이다. 실제로 본 연구에서 제안한 시스템은 <표 1>과 같은 형태로 논리적 결합을 표현하지는 않는다. 대신 협업관리 시스템의 워크스페이스

환경에서 개발자가 특정 revision 에 연관된 버전 링크를 획득하고자 하는 경우, revision 정보를 통한 질의(query)를 통해 해당 revision 과 연관된 버전들을 협업 관리 서버로부터 가져올 수 있는 기능을 제공한다. (그림 4)는 revision 'REV_01'으로부터 산출물들의 버전을 가져오기 위한 과정을 보여주는 화면이다. 이 기능을 통해 형상항목에 포함된 산출물들에 대해서 'REV_01'이 태그로 연결된 버전들을 획득할 수 있다.



(그림 4) revision 정보를 이용한 산출물들의 버전 획득

위의 기능을 실행함으로써 개발자는 특정 revision 과 연결된 버전들의 링크를 식별하고, 산출물들의 논리적 결합을 찾아낼 수 있다. 이를 통해 개발자들은 형상항목의 변경 이력과 더불어 그것을 구성하는 산출물들의 버전 이력을 총체적으로 인식하고 분석할 수 있게 된다.

4. 결론 및 향후 연구

산출물들의 논리적 결합은 과거의 변경에 대한 추적성을 제공할 뿐만 아니라 향후에 일어날 변경에 대한 기준을 제공하기 때문에 유지보수 활동에 매우 중요한 정보로 활용된다. 본 논문은 산출물들의 논리적 결합을 한 형상항목 내에서 식별하는데 초점을 맞추었다. 한 형상항목에 포함된 산출물들에 대해 논리적 결합을 식별하기 위해 산출물들간 버전 링크를 획득하는 방법을 제시하였다. 산출물들간 버전 링크는 형상항목의 revision 을 태그로 연결시킴으로써 획득할 수 있으며, 이러한 과정을 시스템적으로 자동화하여 사용자의 편의를 더했다.

향후 연구에서는 형상항목을 통한 시스템 전체의 baseline 을 식별하고, 그로부터 시스템을 구성하는 모든 산출물에 대한 논리적 결합을 찾아내는 기법에 대해 연구하고자 한다.

참고문헌

[1] Ying A., Murphy G., Ng R., and Chu-Carrol M., "Predicting Source Code Changes by Mining Change History", Transactions on Software Engineering, 30(9), pp. 573-586, 2004.

[2] Zimmerman T., Weibgerber P., Diehl S., and Zeller A., "Mining Version Histories to Guide Software Changes", In International Conference on Software Engineering, pp. 563-572, IEEE Computer Society Press, 2004.

[3] D'Ambros M. and Lanza M., "Reverse Engineering with Logical Coupling", In Working Conference on Reverse Engineering, pp 189-198, 2006.

[4] Pinzger M., "ArchView – Analyzing Envolutionary Aspects of Complex Software Systems", PhD thesis, Vienna University of Technology, 2005.

[5] Fenton N.E., Pfleeger S.L., "Software Metrics – A Rigorous & Practical Approach", International Thomason Computer Press, Second Edition, 1996.

[6] Offen R.J., "Establishing Software Measurement Programs", IEEE Software, pp. 45-53, March/April 1997.

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

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

[9] Kagdi H., Maletic J.I., Sharif B., "Mining Software Repositories for Traceability Links", International Conference on Program Comprehension, pp. 145-154, June 2007.

[10] Kagdi H., Maletic J.I., "Software-Change Prediction: Estimated + Actual", International Workshop on Software Evolvability, pp. 38-43, Sept. 2006.

[11] Kagdi H., Yusuf S., Maletic J.I., "Mining Sequences of Changed-files from Version Histories", International Conference on Software Engineering, pp. 47-53, 2006.

[12] Bieman J.M., Andrews A.A., and Yang H.J., "Understanding Change-Proneness in OO Software Through Visualization", Proceedings of International Workshop on Program Comprehension, pp. 44-53, 2003.

[13] German D.M., "Mining CVS Repositories, the SoftChange Experience", Proceedings of International Workshop on Mining Software Repositories, pp. 17-21, 2004.

[14] Hassan A.E. and Holt R.C., "Predicting Change Propagation in Software Systems", Proceedings of International Conference on Software Maintenance, pp. 284-293, 2004.

[15] Van Rysselberghe F. and Demeyer S., "Studying Software Evolution Information by Visualizing the Change History", Proceedings of International Conference on Software Maintenance, pp. 328-337, 2004.

[16] Robbes R., Pollet D., and Lanza M., "Logical Coupling Based on Fine-Grained Change Information", Working Conference on Reverse Engineering, pp. 42-46, 2008.

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

[18] "IEEE Guide to Software Configuration Management", ANSI/IEEE Std 1042-1987.