DOI: 10.3745/KIPSTD.2009.16D.6.899

소프트웨어 형상관리와 작업공간의 통합을 통한 산출물의 추적성 향상 기법

김 대 엽[†] · 윤 청[††]

요 약

소프트웨어 형상관리를 통한 소프트웨어 개발 생산성 및 품질의 향상은 체계적이고 일관성 있는 변경 관리에 기반한다. 변경 관리에 있어 형상항목으로 식별된 산출물은 조직 구성원들에게 그 변경 이력을 제공해야 하며 이를 통해 구성원들은 해당 산출물에 대한 변경을 정확히 추적할 수 있어야 한다. 일반적인 소프트웨어 형상관리 시스템들은 산출물에 대한 추적 정보를 형상관리 시스템 내에서만 제공하고 있으며 개인의 작업공간에서 발생한 개별적인 변경까지 추적하는 기능을 제공하지 못하고 있다. 본 논문은 형상관리 시스템과 개인의 작업공간을 통합함으로써 형상항목뿐만 아니라 개인에 의한 산출물의 변경까지 추적할 수 있도록 하였다. 통합 환경을 통해 형상항목의 수정버전(revision)을 작업공간의 산출물 버전에 태그로 연결시켰으며 두 환경 사이의 연계된 변경 과정에서 추적성 정보의 효율적인 관리를 가능하게 하였다.

키워드 : 추적성, 소프트웨어 형상관리, 버전관리

Traceability Enhancement Technique Through the Integration of Software Configuration Management and Personal Working Space

Dae-Yeob Kim[†] · Cheong Youn[††]

ABSTRACT

Software productivity and quality improvement through software configuration management is based on organized and consistent change management. In change management, artifact identified as configuration item should be able to provide its changed history to the members in group and the members should be able to track down the changes made for the corresponding artifact. General software configuration management systems provide tracking information for artifacts only within the configuration management system, and it does not go further to changes that occur within personal working space. This paper provides a solution that helps tracking down changes that occur not only in configuration management, but also personal artifact's changes through the integration of configuration management system and personal working space. A revised version in configuration management system is connected to the artifact version of the working space by the tagging technique, and traceability can be managed more effectively by sharing the tracking information.

Keywords : Traceability, Software Configuration Management, Version Control

1. 서 론

소프트웨어 개발 과정에서 발생하는 다양한 종류의 산출물들(요구사항 명세서, 설계문서, 소스코드 등)은 사용자 환경과 개발기술 환경의 변경, 새로운 요구사항의 추가, 오류 수정, 기능 개선 등의 이유로 지속적인 변경이 일어난다. 현대의 소프트웨어 운영 환경은 더욱 복잡해지고 있으며 개발 방법론 또한 다양해지면서 소프트웨어 산출물들을 효율적으로 관리하기 위한 노력이 확산되고 있다.

소프트웨어 산출물들의 변경을 체계적으로 관리하고자 하는 활동으로 소프트웨어 형상관리(SCM: Software Configuration Management, 이후 'SCM'으로 칭함)가 있다. SCM은 소프트웨어의 전체 생명주기를 통해 형상을 인식하여 체계적으로 변경 사항을 관리하고, 소프트웨어의 추적 용이성과 통일성을 유지하기 위한 활동으로 제품의 품질을 보증하며 개발 생산성을 향상시킬 수 있도록 한다. SCM은 이미 대부분의 소프트웨어 개발업체들 사이에서 반드시 필요한 활동으로 자리잡고 있으며 이를 지원하기 위해 수많은 SCM 도구들이 개발 및 사용되고 있다. SCM은 과거 버전관리라는 개념에서부터 출발하였으며 따라서 대부분의 SCM 도구들이

† 준 회 원 : 충남대학교 컴퓨터공학과 석·박사 통합과정
†† 정 회 원 : 충남대학교 전기정보통신공학부 교수(교신저자)
논문접수 : 2009년 9월 4일
수 정 일 : 1차 2009년 11월 16일
심사완료 : 2009년 11월 21일

기본적으로 버전관리 기능을 포함하고 있다. 현대의 SCM 도구들은 여기에 병렬개발 및 변경 추적 등과 같은 기능을 포함, 전체 개발 및 유지보수 프로세스를 제어하고 통제하는 기능들을 제공하고 있다.

버전관리 및 병렬개발 지원 기능 등은 생산성 중심의 차원에서 제공되는 것으로 개발자들의 작업공간에서 발생하는 변경을 관리한다. 개발 및 유지보수 과정에서 전체의 변경 프로세스를 통제하고 관리하는 것은 프로세스 중심의 차원에서 제공되어야 할 기능으로 관리자 및 개발자들 간의 유기적인 업무 흐름(work-flow)을 지원한다. 현재의 SCM 도구들은 생산성 중심의 기능들로 구성된 도구와 여기에 프로세스 중심의 기능들이 추가 통합된 도구로 구분할 수 있다.

생산성 중심의 SCM 도구들은 협업 관리 차원에서 기본적인 버전관리 및 병렬개발 기능을 포함하고, 보다 향상된 분기/병합 기능, Build 자동화 기능 등을 지원함으로써 조직 내 구성원들의 자원 공유 문제, 동시 작업 문제들을 처리할 수 있도록 한다. 생산성 중심 SCM 도구들의 경우 논리적 형상의 제어에 필요한 업무 프로세스를 지원하지 않기 때문에 대규모의 프로젝트를 관리해야 하는 조직에게는 적용하기에 무리가 따른다. 논리적 형상항목의 변경에 대해서 관련 추적성 정보를 충분히 제공하지 못한다는 점 또한 한계로 지적될 수 있다.

프로세스 중심의 SCM 도구들은 작업공간에서의 변경권한을 SCM 서버를 통해 제어한다. 특정 산출물에 대한 변경의 요구가 있을 경우 그것을 실현할 담당자들을 설정하고 변경업무를 할당한다. 즉, 작업권한을 일부의 개발자들에게만 할당함으로써 그 외의 개발자들이 해당 시점에 동일 산출물에 대한 작업에 참여할 수 없도록 한다. 변경의 안정성과 일관성을 위한 정책이지만 이는 개발 생산성을 향상시키는데 있어 하나의 제약 요소로 인식되고 있다[1].

또한 프로세스 중심의 SCM 도구들은 변경에 대한 추적성 정보의 관리가 SCM 영역에 국한된다는 한계가 있다. 일반적인 프로세스 중심 SCM 도구들은 'Change Set'이나 'Task'와 같은 논리적 단위로 변경업무를 할당하고 모든 변경이 실현되면 SCM 서버에 반영(Check In)하도록 하여 이들의 변경 이력을 관리한다. 결국 SCM 환경에서의 논리적 변경에 대한 추적성은 다양한 내용(논리적 대상의 버전, 변경 담당자, 변경 사항, 변경 일자 등)으로 제공하고 있지만, 개인 작업공간에서 이루어지는 변경 흐름에 대해서는 상대적으로 추적성 정보의 관리가 부족한 것이 사실이다.

본 연구는 생산성 중심의 관점에서 작업공간에서의 변경업무에 제한을 두지 않으면서 프로세스 중심의 형상제어 흐름과 연계될 수 있는 환경을 제시한다. 또한 각각의 작업공간에서 이루어진 여러 흐름의 변경에 대해 논리적 형상항목과 연계된 추적성 정보를 나타낼 수 있도록 하는 방법을 제시한다. 추적성 정보는 각 변경에 대한 의미를 이해하는데 도움을 주고 전체적인 변경 흐름을 파악하게 함으로써 과거의 문제나 혹은 앞으로 있을 작업에 대한 방향을 제시해주기 때문에 개발 생산성 및 일관성 유지에 직접적인 관련이 있다고 볼 수 있다. 따라서 개인 작업공간을 통해 이루어진 산출물의 변경 흐름과 논리적 형상항목의 변경을 통합한 추적성 정보는 매우 중요한 의미를 지닌다. 만약 임의의 작업공간에서 일어난 변경에 대해 추적성 정보를 제공하지 못한다면 변경을 수행한 사용자 외에는 해당 산출물의 변경 흐름을 파악하기 어려우며, 그 결과 작업의 중복으로 인한 변경의 일관성 저하, 변경의 통합을 위한 추가적 업무의 발생, 변경 결과에 대한 정확성 보장의 어려움과 같은 부작용을 초래할 수 있다.

앞서 제시한 바와 같이 작업공간과 SCM의 변경을 연계시키고, 작업공간에서의 변경 흐름을 나타내는 추적성 정보를 제공하기 위해 본 연구는 먼저 관리의 대상을 물리적 산출물과 논리적 형상항목으로 구분한다. 물리적 산출물은 사용자 개개인의 작업공간에서 변경의 대상이 되며, 논리적 형상항목은 SCM 영역에서 승인절차에 따른 변경의 대상이 된다. 작업공간에서 변경의 대상이 되는 산출물들은 버전관리, 및 병렬개발 기능을 지원하는 협업관리 서버에 저장된다. 조직의 모든 사용자들은 협업관리 서버에 저장된 산출물들을 필요에 따라 획득하고 원하는 작업을 수행할 수 있다. 이 과정에서 발생하는 모든 변경 흐름은 버전그래프를 통해 기록된다. 또한 논리적 관리 대상인 형상항목은 SCM 서버의 데이터베이스에 기록되고, 향후 변경에 있어 작업공간의 변경과 연계되기 위해 협업관리 서버에 저장된 산출물을 참조한다. 이로써 작업공간에서 발생하는 모든 변경 이력은 SCM의 형상 제어 과정에서 추적 가능해진다. 또한 작업공간에서의 변경 결과가 논리적 형상항목과 연결되고 이것이 기준선(baseline) 문서로 공식화될 때 형상항목의 수정버전(revision)을 협업관리 서버에 존재하는 산출물의 버전 그래프에 태그로 연결시킴으로써 작업공간의 변경 추적성과 SCM 영역의 변경 추적성을 통합하여 나타낼 수 있게 된다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 SCM 도구들의 발전 과정과 그로 인한 상용 도구들의 기능적 분류, 그리고 추적성 관리에 대한 관련 연구를 소개한다. 3장에서는 SCM 환경과 개인 작업공간의 변경을 연계함으로써 이루어지는 변경 시나리오에 대해 기술하고, 4장에서는 SCM 환경과 작업공간이 통합된 시스템 구조 및 그 구현 결과를 소개한다. 5장에서는 제안한 시스템 상에서 추적성 정보를 나타내기 위한 방법을 설명한다. 6장에서는 기존 프로세스 중심 SCM 도구들로부터 제시한 시스템의 구조적 차이점에 대해 구체적으로 기술하고, 마지막으로 7장에서 본 논문의 결론과 향후 연구에 대해 제시한다.

2. 관련 연구

2.1 SCM 도구의 분류

소프트웨어 산출물에 대한 변경의 관리는 과거 1970년대 최초 Unix 기반의 RCS, SCCS을 시작으로 버전관리라는 개념에서 출발하였다. 당시의 버전관리 개념은 단순히 관리대상이 되는 파일들을 일정한 저장소에 모아두고 Check

〈표 1〉 기능에 따른 SCM 도구의 분류

분류		제품 주요 기능	제품군
생산성 중심	Lower-End	기본적인 버전관리 기초적인 병렬개발 지원	Visual Source Safe(MS)
	Higher-End	향상된 투명성 제공 강력한 Build 자동화 기능 지원 통합 환경의 문제/변경 추적 이기종 환경 지원 향상된 분기/병합 기능 지원 변경사항 공지 기능	Visual Enabler(SoftLab) PVCS(InterSolv) SourceIntegrity(MKS) StarTeam(StarBase) ClearCase(IBM Rational) Deliverables Manager(LBMS)
프로세스 중심	생산성 중심 도구의 기능 포함 프로세스 관리 기능 사용자의 역할과 책임 명시 관리를 위한 승인 절차 변경요구(기능 보강, 응급 오류 수정, 변경 등)에 의한 Work-Flow관리		CM Synergy(IBM Rational) CCC/Harvest(CA) PVCS Dimensions (SQL Software) ClearCase UCM(IBM Rational)

In/Check Out을 통하여 대상 파일들에 대한 버전을 관리하는 목적으로 사용되었다. 그 후 80년대에 이르러 병렬개발 환경에서 발생하는 문제점(충돌, 분기, 병합 등)들을 지원할 수 있는 도구들이 출현하기 시작했으며, 이를 통하여 변경관리라는 말이 사용되기 시작했다. 80년대 말에 이르러서는 복잡한 프로그램 운영 환경들을 더욱 강력하게 지원하기 위하여 프로세스라는 개념을 접목시킬 수 있는 변경관리 도구들이 모습을 드러내었다. 현대의 SCM 도구들은 그 종류가 매우 다양하지만 크게 협업관리 차원에서 생산성 향상을 위한 생산성 중심의 도구와 SCM 차원에서의 관리 프로세스를 지원하기 위한 프로세스 중심 도구로 구분할 수 있다. 생산성 중심의 도구는 개발 생산성에 초점을 맞춘 기능들로 구성되는데 버전관리, 병렬개발 지원 기능 등이 대표적이며 보다 높은 수준의 생산성 향상을 위해 Build 자동화 기능이나 변경 추적 기능, 분기 및 병합 기능 등이 제공된다. 프로세스 중심의 도구들은 여기에 SCM 차원의 관리 프로세스를 추가 통합시킨 것으로 개발 각 단계별로 작업에 대한 내용들을 미리 정의하고 작업권한을 제어하며, 승인절차 등을 통해 전체 작업을 통제하는 부분까지 포함한다. 또한 이러한 작업들이 얼마나 사용자들의 요구사항에 맞도록 변화하고 있는지 여부를 검증하고 필요한 부분들을 리포팅하는 것 또한 프로세스 중심 SCM 도구의 중요한 기능이다.

<표 1>은 현재 일반적으로 사용되고 있는 SCM 도구들을 생산성 중심과 프로세스 중심의 관점에서 분류한 결과이다.

2.2 추적성 관리

추적성(traceability)에 대한 정의는 연구 및 산업 분야에 따라 매우 다양하게 표현되고 있다. 기계 산업의 측정 표준, 재료 공학, 소프트웨어 공학, 식품 처리 등의 각 분야마다 나름대로 추적성에 대한 정의를 적용하고 있다. 가장 정형화된 형태의 정의에서 추적성이란 고유하게 식별될 수 있는 개체들을 시간의 개념과 함께 연관시키고 그 관계를 검증할 수 있는 능력으로 간주된다[2]. ISO와 같은 인증기관에서는 고객에게 제품의 품질을 보장하기 위한 수단으로 추적성 관리의 중요성을 강조하고 있으며 이에 따라 각 산업 분야마다 추적성 관리를 위한 지침과 절차서를 개발하여 적용하고 있다[3].

소프트웨어 분야에서 추적성에 대한 정의는 그것의 적용 범위와 목적에 따라 다소 다르게 나타나고 있다. 따라서 적용하고자 하는 범주를 먼저 설정하고 그에 따라 추적성에 대한 정의를 제시해야 한다[4]. 일반적으로 소프트웨어 분야에서는 요구사항과 그로부터 파생되는 다른 산출물들의 관계를 관리하는 능력으로 추적성을 정의하고 있다[4-8].

요구사항으로부터 파생된 산출물들(설계문서, 각종 구현물(소스코드, 실행파일 등), 테스트 케이스 계획서 등)이 요구사항 명세서에 기재된 모든 기능들을 만족시키고 정확하게 반영하였는지를 검증(validation) 및 확인(verification)하는 것은 앞서 정의한 추적성을 뒷받침할 수 있는 활동들이다. 검증은 요구사항 분석의 결과가 사용자의 요구를 만족시킬 수 있는 것인지, 사용자가 원하는 올바른 제품을 만들었는지 검사하는 활동이며, 검사의 결과를 기록하기 위해서 검증 테이블을 사용할 수 있다. 확인은 개발 주기 상에서 현 단계의 산출물이 바로 이전 단계의 산출물과 일치하는가를 결정하는 과정으로, 사용자가 바라는 대로 소프트웨어가 만들어졌음을 확신할 수 있도록 수행하는 활동이다. 요구사항의 기능적 요소들을 기준으로 다음 단계의 산출물들이 그것을 정확하게 반영하고 있는지 나타내기 위해서 확인 행렬(verification matrix)이 사용되며, 이러한 확인 행렬을 통해 요구사항 명세서에 기재된 기능들에 대한 추적을 수행할 수 있다.

요구사항의 기능들에 초점을 두고 관리되는 검증 테이블이나 확인 행렬 외에도 단순히 산출물들 간의 관련성을 표현하기 위해 추적성 매트릭스(traceability matrix)나 추적성 네트워크(traceability network)와 같은 기법들이 사용되기도 한다. 추적성 매트릭스는 요구사항 명세서, 설계문서, 소스코드, 테스트 케이스 계획서 등과 같은 산출물들 간의 대응관계를 테이블 형태로 표현하며, 추적성 네트워크는 이러한 산출물들의 관계를 다양한 형태의 링크로 표현한다.

앞서 설명한 일반적인 추적성의 정의가 소프트웨어 개발과 관련된 추적성의 모든 관점을 포괄할 수 있다고 보기는 힘들다[9]. 서로 다른 산출물들 간의 추적성 뿐만 아니라 특정 산출물 내부에 대한 추적성, 또는 시간의 변화에 따른 산출물의 변화에 대한 추적성 등 다양한 범주에 적용할 수 있는 추적성의 정의가 이루어져야 한다.

본 논문에서는 단일 산출물의 변화 과정을 기록하고 분석할 수 있는 능력으로 추적성을 정의하고 있다. 이 경우 추적성은 근본적으로 단일 산출물의 변경과 그 영향 정도를 분석하기 위한 목적으로 사용될 수 있으므로 서로 다른 산출물들 간의 추적성과 함께 변경의 이력을 나타내는 중요한 정보로 사용될 수 있다.

Berczuk는 단일 산출물의 추적성을 두 가지로 구분하였다. 개인적인 작업공간에서의 변경에 대한 지역 추적성(local traceability)과 모든 사용자들이 변경을 공유할 수 있는 전역 추적성(global traceability)이 그것이다[10]. 지역 추적성은 작업공간에서 발생하는 산출물의 변경을 개인적으로 관리하고 유지하기 위해 필요한 개념이며, 전역 추적성은 SCM 환경을 통해 여러 사용자들이 형상항목에 대한 변경 정보를 공유하고 그 과정을 추적할 수 있도록 하기 위한 개념이다. 그가 제시한 SCM 패턴에서 이러한 두 가지 추적성의 개념은 따로 분리될 수 있는 것이 아니며, 소프트웨어 제품을 개발하는 과정에서 개인적인 작업으로부터 특정 릴리즈(release)를 구성해 나가는 환경의 흐름을 효율적으로 관리하기 위한 것이다. 다시 말하면 개인적으로 관리되는 산출물의 추적성 정보와 여러 사용자들이 SCM 환경을 통해 공유하는 형상항목의 추적성 정보를 통합 관리할 때 보다 효율적인 개발 환경을 구성할 수 있다는 것이다.

<표 1>에서 생산성 중심 SCM 도구들은 SCM 차원의 전역 추적성을 제공하지 못하고 개인 작업공간의 지역 추적성만을 제공한다는 점에서 한계로 지적될 수 있다.

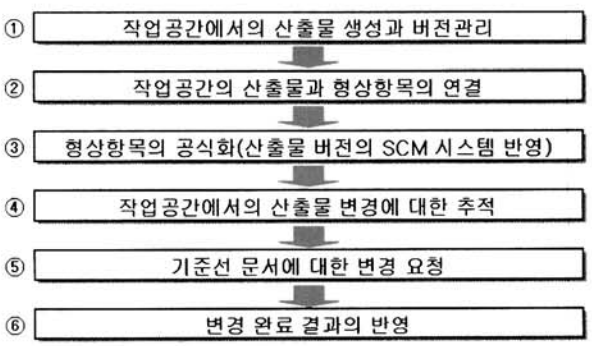
프로세스 중심 SCM 도구들은 지역 추적성과 전역 추적성을 따로 구분하지 않고 SCM 수준의 전역 추적성만을 제공하는 경향이 있다고 볼 수 있다. 대부분의 프로세스 중심 SCM 도구들은 개인의 작업공간과 SCM 환경에서 모두 동일한 개념의 논리적 단위를 변경의 대상으로 삼고 있기 때문에 모든 변경의 결과는 SCM 관점에서만 의미를 가질 뿐 개인 작업공간에서의 임의의 변경 흐름은 중요하게 여겨지지 않는다. 따라서 이 경우도 지역 추적성과 전역 추적성을 통합 제공한다고 보기 힘들다.

본 연구는 이러한 문제점을 해결하기 위해 개인의 작업공간과 SCM 환경을 구조적으로 구분하되 전체적인 변경 과정에서 작업공간과 SCM의 변경 결과를 통합 추적할 수 있는 방안에 대해 제시한다. 먼저 작업공간에서의 변경 이력은 해당 산출물이 논리적 형상항목에 참조방식으로 연결됨으로써 추적 정보의 투명성을 제공할 수 있다. 또한 논리적 형상항목의 공식화된 버전 정보는 작업공간의 산출물 버전에 태그로 결합됨으로써 산출물의 버전 흐름 가운데 기준선과 관련된 버전과 임의로 추가 생성된 버전들을 구분할 수 있도록 한다. 이를 통해 SCM 환경에서의 형상항목에 대한 전역 추적성 뿐만 아니라 개인 작업공간에서의 산출물에 대한 지역 추적성을 모두 제공하고 보다 확장된 추적성 정보를 활용할 수 있게 된다.

3. SCM 환경과 개인 작업공간의 통합 변경 시나리오

이 장에서는 SCM 시스템과 개인 작업공간을 통합했을 때 이루어지는 전반적인 변경 과정에 대해 기술하고자 한다. 먼저 개인 작업공간에서 생성된 산출물은 SCM의 기준선 문서와 연결되어 변경이 이루어지며 변경의 결과는 기준선 문서의 공식화 과정에 따라 SCM 서버에 반영된다. 사용자들은 기준선 문서를 통해 SCM에 반영된 산출물 버전을 획득하게 되고 획득한 버전과 이미 개인 작업공간에서 새롭게 생성된 버전들을 연계하여 변경에 대한 추적 정보를 얻을 수 있다.

(그림 1)은 작업공간에서 만들어진 산출물이 SCM의 기준선 문서에 연결되는 것부터 변경 결과를 SCM 서버에 반영하는 과정까지의 흐름을 나타내고 있다.



(그림 1) SCM 환경과 작업공간의 통합 변경과정

3.1 작업공간에서의 산출물 생성과 버전관리

통합 환경에서의 변경관리는 개인 또는 공동(협업)의 작업공간에서 산출물을 생성함으로써 시작된다. 작업공간에서는 생성된 산출물이 형상항목으로 식별되어 SCM 서버에 신규 등록되기까지, 혹은 SCM 서버에 등록된 이후의 변경 요청에 대해 그것을 만족시키기까지 수차례의 변경이 일어날 수 있으며 이러한 변경을 관리하기 위해 버전관리 기법을 사용한다. 작업공간에서의 버전관리는 산출물에 대해 다양한 흐름의 변경을 가능하게 하고 개인적인 변경 이력을 보관할 수 있도록 지원한다[11-16]. 작업공간의 버전관리는 비공식적인 영역에서 개인 작업의 추적성(local traceability)을 유지하기 위한 목적으로 이루어진다.

3.2 작업공간의 산출물과 형상항목의 연결

작업공간에서 산출물에 대한 작업이 완료되면 사용자는

완료된 버전을 SCM 시스템에 반영하기 위해 결재 요청을 수행해야 한다. SCM 시스템에서 결재 요청은 형상항목을 통해 이루어지기 때문에 사용자는 작업공간의 산출물과 형상항목을 연결시켜야 한다. 산출물과 형상항목의 연결은 파일 첨부 방식이나 작업공간의 디렉토리 구조를 참조시키는 방식을 통해 이루어진다. 작업공간의 디렉토리 구조는 정확히 말하면 협업관리 서버에 존재하는 산출물의 구조이다. 사용자들은 협업관리 서버에서 산출물을 획득하여 자신의 작업공간을 형성하기 때문에 실제로는 SCM 서버와 협업관리 서버의 연결을 통해 두 대상이 연결되는 것이다. 이러한 연결 관계를 통해 작업공간의 산출물 버전은 향후 형상항목이 기준선 문서로 공식화될 때 SCM 시스템에 반영된다.

3.3 형상항목의 공식화(산출물 버전의 SCM 시스템 반영)

작업공간의 산출물과 형상항목이 연결된 상태에서 사용자는 형상항목의 공식화를 위해 결재 요청을 수행한다. 결재 요청이 최종 승인되면 사용자는 체크인(check-in) 기능을 실행하여 형상항목을 공식화하고 그 결과 해당 형상항목은 SCM 시스템 상에서 기준선 문서가 된다. 형상항목이 기준선 문서로 공식화되었다는 것은 연결된 산출물의 버전이 SCM 시스템에 반영되고 동결되었음을 의미한다. 이를 나타내기 위해 SCM 시스템은 자동적으로 기준선 문서의 버전을 작업공간으로 보내어 해당 산출물의 버전에 태그로 붙인다. 기준선 문서의 버전은 형상항목의 수정버전(revision)으로서 'REV_00'과 같은 형태로 표현할 수 있으며 이는 개인 작업공간에서 산출물에 부여하는 버전과는 구별되는 개념이다. 기준선 문서의 버전은 SCM 시스템 상에서 전역 추적성을 나타내기 위한 정보로서 활용되며, 태그를 통해 기준선 문서의 버전과 작업공간의 산출물 버전을 연결시키는 것은 SCM 시스템의 전역 추적성과 작업공간의 지역 추적성을 통합 관리하기 위한 방법이다. 사용자들은 이러한 태그 정보를 통해 산출물의 공식화된 버전을 파악할 수 있게 된다.

3.4 작업공간에서의 산출물 변경에 대한 추적

형상항목이 기준선 문서로 공식화된 이후에도 사용자들은 자신의 작업공간 내에서 연결된 산출물에 대해 자유롭게 변경을 수행할 수 있다. 사실 일반적인 SCM 도구들은 산출물이 한번 동결되면 그에 대한 임의의 변경을 불허하며 철저한 결재 절차를 통해서만 변경이 가능하도록 지원한다. 본 연구에서 제시한 통합 환경은 사용자들이 자신의 작업공간에서 임의의 작업을 수행할 수 있도록 하되 그 결과가 SCM 영역의 공식화된 버전에 영향을 주지 않도록 함으로써 보다 나은 개발 생산성을 지원한다[12, 16-17].

작업공간에서의 자유로운 변경은 기존 SCM 서버에 반영된 산출물의 버전에 추가적인 흐름의 버전들을 생성한다. 사용자들은 SCM 서버에 반영된 버전에 대해 같은 흐름으로 버전을 생성하기도 하며 때로는 분기(branch)를 생성하여 별도의 흐름으로 작업을 수행할 수도 있다. 사용자들은 분기를 통해 기본 흐름(mainline)에 영향을 주지 않고 독립적인 흐름 가운데 각자가 의도한 작업을 자유롭게 수행할 수 있다. 분기는 특정 버전에 대한 버그를 별도의 흐름으로 수정하고자 하는 경우, 상이한 고객의 요구로부터 별도의 릴리즈를 개발하고자 하는 경우, 특정 플랫폼에 맞는 버전을 개발하고자 하는 경우 등에 의해 생성될 수 있다. 분기에 의한 작업은 그 결과에 따라 기본 흐름에 병합되어 새로운 버전을 생성할 수도 있다.

작업공간에서의 다양한 변경 흐름은 SCM의 기준선 문서와 연계되어 추적될 수 있다. 사용자는 기준선 문서와 함께 동결된 산출물의 버전을 확인하고 그로부터 추가적으로 발생한 버전들을 추적함으로써 작업공간에서의 변경이 얼마나 수행되었는지 확인할 수 있다. 작업공간에서 발생한 산출물의 변경을 추적하는 것은 향후 기준선 문서의 변경 요구에 대한 작업의 경과를 확인하기 위해 반드시 필요한 활동이다.

3.5 기준선 문서에 대한 변경 요청

기준선 문서의 변경은 이전의 공식화 과정과 마찬가지로 결재의 과정을 거쳐 이루어진다. 기준선 문서의 변경 요청에 대해 최종 승인이 이루어지면 지정된 변경 담당자는 기준선 문서를 체크아웃(check-out)하여 변경 권한을 획득한다. 기준선 문서의 변경 권한을 가진 변경 담당자는 기준선 문서에 연결된 산출물을 획득하고 통합 환경을 통해 현재 다른 작업공간에서 해당 산출물에 대해 수행 중인 변경을 추적할 수 있다.

일반적으로 산출물에 대한 다른 사용자의 작업 내용을 공유하기 위해서는 공유 사용자들의 정보와 산출물의 버전 정보를 관리하는 협업관리 시스템이 필요하다. 변경 담당자를 비롯한 모든 공유 사용자들은 협업관리 시스템을 통해 서로의 작업 내용을 공유하고 추적할 수 있다. 특히 변경 담당자는 작업공간의 버전 흐름 가운데 자신이 체크아웃 한 기준선 문서의 버전을 태그로 가지는 버전에 대해 확인하고 그로부터 추가로 발생한 여러 버전들을 추적함으로써 그 동안의 변경 내용을 분석할 수 있게 된다.

3.6 변경 완료 결과의 반영

작업공간에서 산출물의 변경을 추적하고 요구를 만족시키기 위한 모든 작업이 완료되면 기준선 문서의 변경 담당자는 그 결과를 다시 SCM 서버에 반영해야 한다. 산출물의 변경 결과를 SCM 서버에 반영하기 위해서는 ③번 흐름에서와 같은 결재 과정을 거쳐야 하며, 결재의 승인이 이루어지면 다시 체크인 기능을 실행하여 공식화하는 과정을 수행한다. 그 결과 기준선 문서에 새로운 버전이 생성되고 이는 또 다른 태그 정보로서 작업공간의 산출물 버전에 연결된다.

지금까지 작업공간의 산출물이 기준선 문서와 연결되어 SCM 서버에 반영되는 과정에 대해 설명하였다. 작업공간에서의 변경에 대한 산출물 추적성은 개인의 필요에 의해서라기보다는 조직 구성원들이 서로의 작업을 이해하고 파악함으로써 보다 나은 개발 생산성을 지원하기 위해 더욱 중요한 의미를 갖는다. 따라서 SCM을 통한 변경 관리 업무와

개인 작업공간을 통합하는 일은 보다 성숙된 개발 환경을 제공하기 위해 반드시 필요하다.

4. 시스템 통합 환경

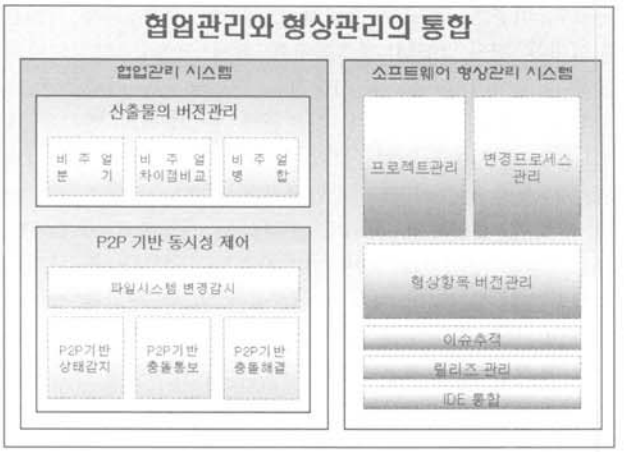
본 장에서는 개인 작업공간을 SCM 환경과 통합 운영하기 위한 구조를 제시한다. SCM의 전역 추적성뿐만 아니라 개인 작업공간에서의 지역 추적성을 통합 관리하고자 하는 이유는 산출물이 여러 사용자들에 의해 공유될 수 있는 대상이기 때문이다. 기존 SCM 서버에 반영된 버전에 대해서 추가로 생성된 버전들은 해당 산출물을 변경하고자 하는 또 다른 사용자들에게 공개되어야 하며 이를 기반으로 한 추적 정보에 의해 여러 사용자들의 중복된 작업을 예방할 수 있다.

개인 작업공간에서의 자유로운 변경작업과 산출물의 공유에 의한 문제들을 다루기 위해 본 연구는 협업관리 시스템을 제공하며, 협업관리 시스템과 SCM 시스템의 통합을 바탕으로 총체적인 변경관리 및 추적성 관리를 지원하고자 한다.

(그림 2)는 통합의 대상인 협업관리 시스템과 SCM 시스템이 갖춰야 할 기능적 요소들을 나타내고 있다.

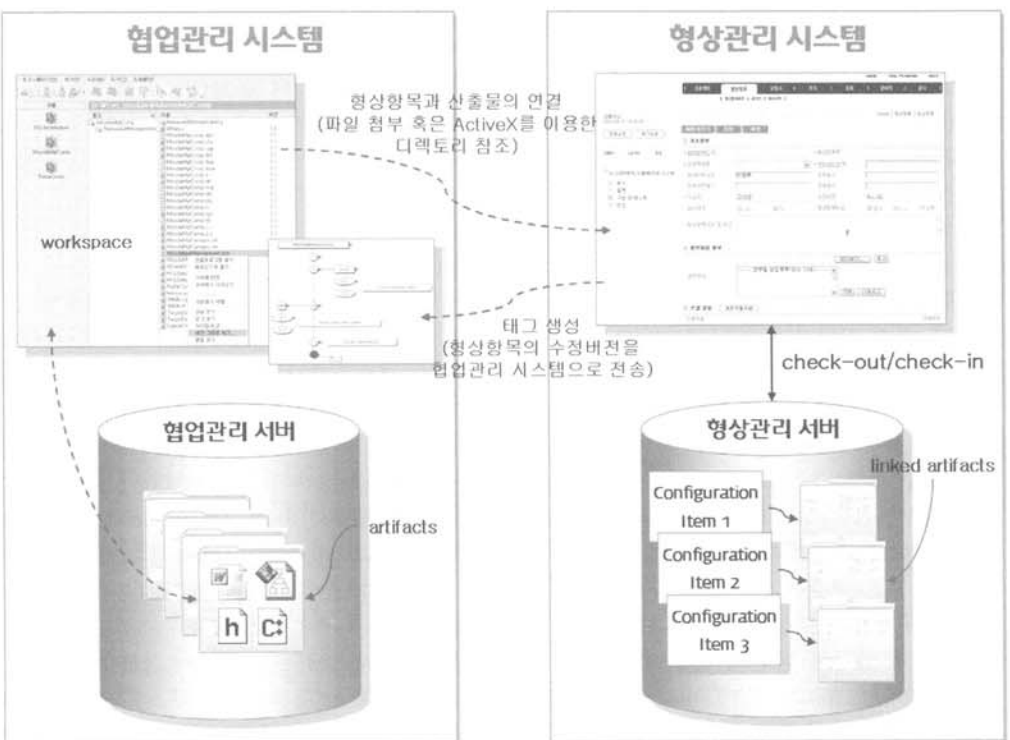
협업관리 시스템은 개인 및 공동의 작업공간을 지원하기 위해 설계되었으며 SCM 시스템은 일반적인 형상관리 업무를 지원하기 위한 기능들로 구성되어있다.

(그림 3)은 통합 환경에서 각 시스템의 상호운용 과정을 도식화한 것이다. 협업관리 시스템의 산출물을 웹 기반



(그림 2) 협업관리 시스템과 SCM 시스템의 기능

SCM 시스템의 형상항목과 연결시키는 방법 중 작업공간의 디렉토리 구조를 참조시키는 방식을 지원하기 위해 ActiveX 형태의 라이브러리 파일을 각 사용자의 레지스트리에 등록하는 과정이 필요하다. 이러한 과정은 SCM 시스템의 형상항목 페이지 실행 시 자동으로 이루어지며 이를 통해 각 사용자의 작업공간에 존재하는 산출물 버전이 SCM 시스템에 반영될 수 있다. 또한 등록된 형상항목의 수정버전을 작업환경에 보내어 산출물의 버전에 연결시키기 위해 JNI(Java Native Interface) 프로그래밍 기법을 사용하였다. Java 기반의 웹 페이지에서 형상항목의 수정버전을 매개변수로 하여 협업관리 시스템의 태그 생성 함수를 호출하면 산출물의 버



(그림 3) 협업관리 시스템과 SCM 시스템의 상호 운용

전에 형상항목의 수정버전이 태그로 붙게 된다. 이들 기법은 서로 다른 개발 플랫폼에서 구현된 두 시스템의 상호운용을 위해 적용된 것이다.

협업관리 시스템은 개인 작업공간에서 산출물의 변경을 관리하기 위한 버전 관리 기능과 공유 사용자들 간의 협업 시 동시성 제어를 다루기 위한 기능 등으로 구성된다. 이것은 생산성 중심의 기능에 해당된다. 산출물의 버전 관리는 분기, 병합, 차이점 비교 등의 기능을 편리하게 수행할 수 있도록 시각화된 화면을 제공한다. 협업관리 시스템은 버전 관리를 위해 CVS의 버전제어 기법과 저장소 관리 메커니즘을 기반으로 구현되었으며 동시 작업을 지원하기 위해 별도의 충돌해결 기법을 제공한다.

(그림 4)는 협업관리 시스템의 워크스페이스(workspace) 환경을 보여준다. 워크스페이스는 작업공간을 독립적으로 구성하여 개인의 자유로운 변경 작업을 지원하기 위한 공간이다. 워크스페이스에 존재하는 산출물은 각각의 버전정보를 갖고 있으며 사용자는 원하는 산출물의 버전정보를 버전 그래프에 의해 확인할 수 있다. 각 산출물의 변경을 위해 사용자는 해당파일 포맷에 맞는 응용 프로그램을 실행하여 작업할 수 있다.

SCM 시스템은 접근의 용이성을 위해 웹 기반으로 개발되었으며 여러 기능들이 있지만 본 논문에서는 기준선 문서의 변경 제어 및 버전 관리 기능에 초점을 두어 설명한다. 이것은 앞선 형상관리 도구의 분류에서 프로세스 중심의 기능에 해당된다. 변경 제어는 특정 기준선 문서에 대해 변경 요청이 있을 경우 그것을 평가하고 적용하는 등의 절차를 의미한다. 정해진 결재 절차를 거치지 않고서는 공식화(check-in)된 기준선 문서를 변경할 수 없으며, 변경권한을 획득하기 위해서는 그에 대한 요청이 선행되어야 한다. SCM 시스템의 버전 관리는 이미 기준선 문서로 공식화된 형상항목에 대한 것으로 수정버전의 관리를 의미한다. 형상항목의 수정버전은 제품의 기준선을 결정짓는 중요한 정보이다. 기준선은 모든 검토가 완료된 형상항목들의 집합으로 정의할 수 있으며 소프트웨어 공학의 개발 프로세스 관점에서 보면 다음 단계의 업무를 위한 입력 자료로서 의미를 가진다.

(그림 5)는 웹 기반의 SCM 시스템을 보여주고 있다. 해당 화면은 특정 형상항목에 대한 화면으로 선택한 형상항목의 기본 정보가 나타난다.

위의 그림에서 현재 형상항목의 수정버전은 'REV_02'로 표시되어 있다. 형상항목의 수정버전은 형상항목이 공식화될 때마다 변경 담당자에 의해 지정되며 꼭 위와 같은 형태로 표시될 필요는 없으나 다른 사용자들이 이해할 수 있는 내용으로 기록되어야 할 것이다. 형상항목이 기준선 문서로 공식화될 때 수정버전은 화면 아래 연결된 산출물의 버전그래프에 태그로 연결된다. 연결된 산출물은 기본적으로 협업관리 시스템의 워크스페이스 환경에서 관리되지만 형상항목과의 연계성을 지니기 위해 위와 같이 관계를 맺는다. 만약 (그림 5)의 상태에서 새로운 변경 요청이 발생하여 형상항목과 그 안의 산출물들의 변경이 이루어졌다면 형상항목의 수정버전은 'REV_03'이 되고 해당 정보가 다시 산출물의 버전그래프에 태그로 붙게 될 것이다. 이러한 과정이 시스템적으로 자동화됨으로써 변경 업무에 관련된 사용자들은 효율적으로 통합 추적성 정보를 관리할 수 있게 된다.

(그림 4) 협업관리 시스템의 워크스페이스 환경

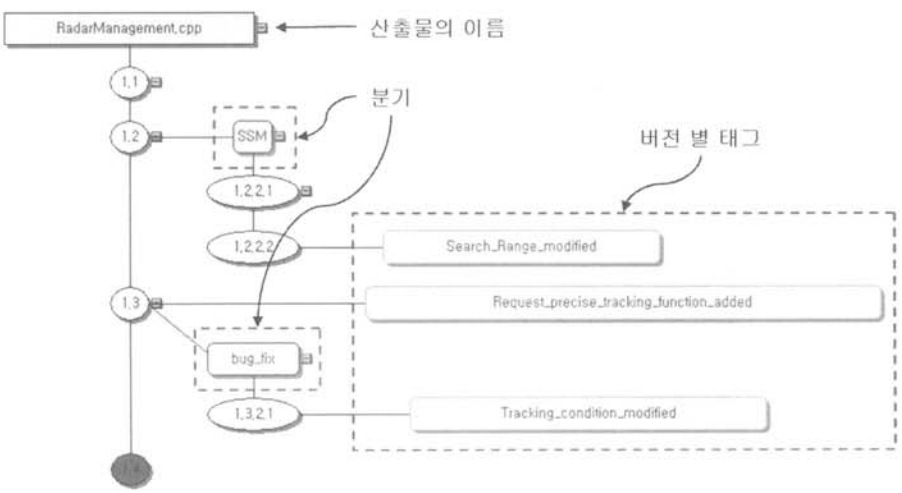
(그림 5) SCM 시스템의 형상항목 화면

5. 산출물의 버전그래프와 추적성의 표현

본 장에서는 작업공간에서 산출물의 버전 흐름을 나타내기 위한 버전그래프를 소개하고, 형상항목의 수정버전이 버전그래프에 태그로 붙음으로써 보다 수월하게 변경을 추적할 수 있도록 하는 방법에 대해 설명한다. 형상항목의 수정버전이란 기준선 문서가 SCM 시스템에 공식화된 횟수를 의미한다.

작업공간에서의 산출물에 대한 반복적인 변경을 관리하고 버전의 흐름을 표현하기 위해서 다양한 형태의 버전모델이 사용된다. 본 연구에서는 버전그래프를 사용하여 작업공간에서의 버전 흐름을 표현하였다. 버전그래프는 트리 구조의 형태를 띠며 버전의 이름은 시스템마다 다르게 정의될 수 있다. (그림 6)은 본 연구를 통해 표현한 버전그래프의 모습이다.

버전그래프는 산출물의 버전정보 외에 분기(branch)나 태그(tag)와 같은 정보들을 포함한다. 분기는 특정 버전에 대해 버그를 수정한다거나 다른 응용분야에 적용하기 위한 새로운 흐름의 개발을 진행하고자 하는 경우 생성할 수 있다. 위의 그림에서 보듯이 각 분기는 그 목적이나 의도를 나타내는 분기 이름에 의해 구분할 수 있으며 생성된 분기는 기본 흐름과 별도의 흐름으로 진행된다. 분기에 의해 수행된 변경 결과는 필요할 경우 다시 기본 흐름에 병합(merge)되어 반영되기도 한다. Berczuk가 제시한 SCM 패턴에서 분기는 기본 흐름(mainline)에 영향을 주지 않는 개인의 codeline으로서 개발의 효율성을 극대화시킬 수 있는 요소이다[10].



(그림 6) 산출물의 버전그래프

태그는 각 버전에 추가적인 의미를 부여하고자 할 때 사용된다. 태그를 통해 사용자들은 각 버전에 어떠한 변경이 일어났는지 개략적으로 이해할 수 있으며 이는 산출물의 변경 흐름에서 이정표(milestone)와 같은 의미를 지닌다.

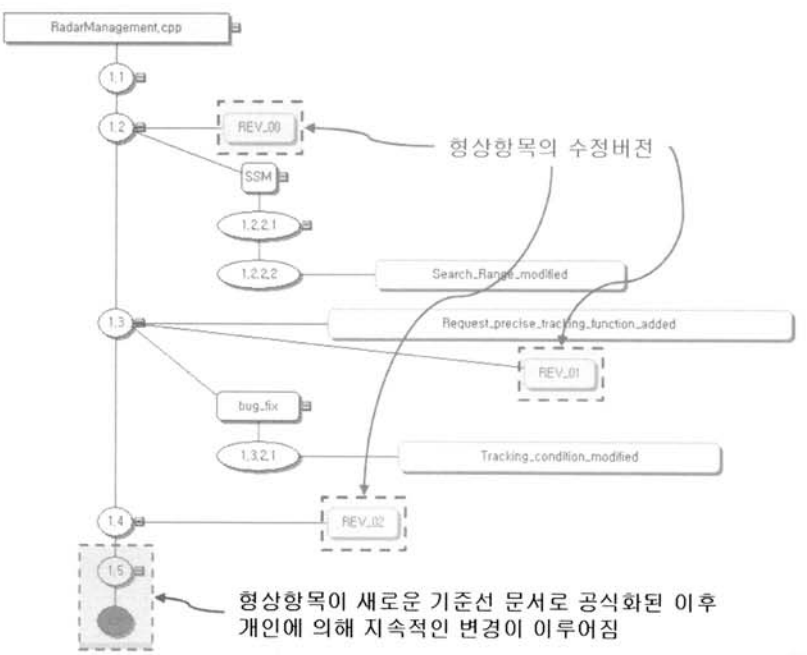
산출물의 버전그래프에 기준선 문서와 관련된 추적성 정보를 보다 확실히 나타내기 위해 본 연구에서는 형상항목의 수정버전을 태그로 붙이는 기법을 사용하였다. (그림 6)에서 나타난 태그 정보가 개인에 의해 임의로 생성된 것이라면 형상항목의 수정버전은 그것이 공식화되는 시점과 맞물려 시스템이 자동적으로 생성한다. 임의로 생성된 태그와 구분하기 위해 SCM 시스템은 형상항목의 수정버전을 약속된 규칙(3장에서 정의한 'REV_01'과 같은 형태)에 의해 정형화하고 이를 버전그래프에 결합시킨다. 자동화된 태그 생성 기능과 규칙에 의한 태그 명명 방식은 사용자들 사이에서 발생할 수 있는 혼란을 막고 업무의 생산성을 향상시킬 수 있다는 장점을 지니고 있다. (그림 7)는 형상항목의 수정버전이 개인에 의해 변경된 산출물의 버전그래프에 태그로 결합된 모습을 보여준다.

(그림 7)에서 형상항목의 수정버전을 의미하는 'REV_00', 'REV_01', 'REV_02'가 버전그래프에 태그로 결합된 모습을 볼 수 있다. 그림에서 볼 수 있듯이 산출물의 버전 '1.4'가 형상항목의 수정버전 'REV_02'에 반영된 이후에도 개인 작업공간에서는 지속적으로 변경이 수행되어 새로운 버전을 생성할 수 있다(버전 '1.5'와 '1.6'의 경우). 이 경우 사용자들은 'REV_02'라고 하는 형상항목의 수정버전 태그를 통해 SCM에 최종적으로 반영된 이후 산출물의 변경이 얼마나 진행되고 있는지를 직관적으로 파악할 수 있게 되는 것이다.

6. 기존 SCM 시스템들과의 비교 평가

본 장에서는 생산성 중심 SCM 도구와 프로세스 중심 SCM 도구의 기능적 특성을 기반으로 제안한 시스템의 차별성을 평가하고자 한다. 현재 일반적으로 사용되고 있는 SCM 도구들은 작업공간에서의 버전관리와 SCM의 변경관리 업무를 통합 운영하는데 한계를 지니고 있다. 생산성 중심 도구의 경우 작업공간에서의 협업 관련 기능들을 통해 생산성을 향상시키는데 초점을 맞추고 있으나 SCM의 형상제어와 같은 변경관리 기능은 제공하지 않고 있다. 프로세스 중심 도구의 경우는 작업공간과 SCM을 통합함으로써 안정적인 형상 제어를 가능하게 하지만 작업공간에서의 효율적인 협업 가능성을 제한하기 때문에 생산성 측면에서 한계를 지니고 있다.

Source Integrity(SI)와 StarTeam은 생산성 중심의 SCM 도구로 작업공간에서의 버전관리와 분기 및 병합 기능을 이용한 병렬개발 등을 지원한다. Source Integrity는 작업공간에서 단일 변경 업무 혹은 여러 변경들의 집합을 change packages라고 하는 논리적 단위로 설정하고 저장소인 project와 연관시킨다. Source Integrity의 작업공간은 sandbox라 불리는데 sandbox에서 변경하고자 하는 파일을 check out하게 되면 해당 파일은 project에서 lock 상태가 된다. 이것은 check out한 사용자 외에 다른 사용자들은 해당 파일에 대한 변경을 수행할 수 없음을 의미한다. SCM의 형상제어 흐름을 지원하지 않기 때문에 개인 작업공간에서는 누구나 원하는 파일에 대해 변경을 수행할 수 있지만, 이러한 locking 알고리즘은 여러 명의 사용자가 산출물을 공유하고 변경할 수 있는 가능성을 제한하게 된다[18].



(그림 7) 형상항목의 수정버전이 태그로 연결된 모습

StarTeam은 StarTeam view라고 하는 작업공간을 통해 저장소의 파일을 check out하고 원하는 변경을 수행한다. StarTeam의 저장소는 데이터베이스의 구조를 가지고 있으며 변경에 관련된 여러 정보들(변경 파일, 변경 내용, 요구사항 등)을 관리한다. 이러한 정보들을 통해서 각 변경이 지니는 이력을 파악할 수 있으며 이는 곧 작업공간에서의 변경 추적성을 의미한다. 생산성 중심의 도구이기 때문에 SCM의 관리 프로세스는 제공하지 않으며 따라서 SCM 차원의 전역 추적성은 지원하지 못한다[19].

CM Synergy는 프로세스 중심 도구로서 변경의 논리적 단위를 task로 정의하고 개발자들은 할당된 task에 대해서 work areas 혹은 work projects라고 불리는 자신의 작업공간을 설정한다. 할당된 task의 변경이 완료되면 complete task 오퍼레이션을 실행하여 프로젝트에 그 결과를 반영하고 이때부터 다른 개발자들이 변경의 결과를 확인 할 수 있게 된다. 프로젝트에 반영된 task에는 baseline의 구성요소를 의미하는 label이 기록되며 이것은 SCM 차원의 전역 추적성 정보를 제공한다. Task에 대한 변경의 결과가 반영되기 전에는 다른 개발자들은 개인의 작업공간에서 이루어지는 변경 과정을 추적할 수 없기 때문에 지역 추적성은 제공되지 않는다. 즉 변경의 결과가 SCM 환경에 반영된 이후에야 그 결과를 추적할 수 있으므로 전역 추적성은 제공하지만 지역 추적성을 제공하는데 있어 한계가 있다고 볼 수 있다[20].

CCC/Harvest 역시 프로세스 중심의 SCM 도구로서 작업공간에서 이루어지는 모든 변경은 중앙에서 제어된다. 즉 개인의 필요에 의해 임의의 변경이 발생하지 않도록 원천 봉쇄한다. 변경에 대한 요구 및 변경 결과의 반영은 온라인 승인절차를 거쳐 이루어지고 이와 같은 제어 프로세스는 작업의 안정성을 보장한다. 승인절차를 거쳐 반영된 변경 결과는 변경 담당자, 변경 내용, 변경 일시, 수정 버전 등의 내용과 함께 SCM 서버에 기록된다. CCC/Harvest의 경우 작업의 권한을 중앙에서 제어하고 모든 변경 이력은 SCM 차원에서 관리되기 때문에 작업공간에 발생하는 변경 이력은 따로 관리하지 않는다. 결론적으로 제공되는 변경 이력은 전역 추적성으로서의 의미만을 가진다[21].

PVCS Dimensions는 일반적인 프로세스 중심의 SCM 도구와 마찬가지로 work package라고 하는 논리적 단위의 변경 대상을 변경 담당자에게 할당한다. 변경된 work package는 SCM 서버에 반영될 경우 변경 문서(공식화된 문서)로서의 의미를 가지며, CM Synergy와 마찬가지로 산출물의 버전에 baseline을 의미하는 label이 부착된다. 이 역시 전역 추적성의 의미를 갖는다. 기본적으로 작업권한에 대한 제어 흐름은 다른 프로세스 중심 도구와 동일하며 작업권한의 여부에 따라 작업공간에서의 변경이 제한적이기 때문에 지역 추적성에 대한 관리는 따로 이루어지지 않고 있다[22].

ClearCase는 작업환경에서의 버전관리 및 병렬개발을 위해 기본적인 기능들을 제공하는 시스템(non-UCM)과 SCM 환경에서 형상제어 업무를 지원하기 위한 시스템(UCM, Unified Change Management)로 구성된다. 이는 본 논문에서 제시한 작업공간과 SCM 시스템의 통합 관점에서 볼 때 가장 유사한 환경을 제공하며 추적 정보를 표현하기 위한 기법 또한 크게 다르지 않다. 지적하고자 하는 부분은 개인 작업공간에 대한 기능적 차이점으로 환경 및 추적성의 통합에 대한 내용과는 다소 거리가 있다. 하지만 본 연구에서 제시한 작업공간이 기본적으로 여러 명의 사용자가 동일 산출물을 공유하고 모든 변경 작업에 참여할 수 있도록 가능성을 제공하는 반면 ClearCase의 경우는 그렇지 않다는 점에서 문제로 지적될 수 있다. ClearCase에서는 산출물의 공유를 허용하지만 실제 변경을 수행할 수 있는 사용자는 단 한 명으로 제한되어 있다. 따라서 먼저 변경 권한을 획득하지 못한 사용자들은 읽기 권한(read-only)만을 가지게 되어 변경 작업에 참여할 수 없게 된다. 이는 산출물에 대한 변경의 일관성을 확실히 보장하기 위한 제어 기법이지만, 일관성을 저해하지 않고 여러 명의 사용자가 변경에 참여할 수 있도록 하는 본 연구의 작업공간과 비교했을 때 개발 생산성에 있어 어느 정도 한계를 지니고 있다고 볼 수 있다[23].

살펴본 바와 같이 생산성 중심의 SCM 도구와 프로세스 중심의 SCM 도구들은 각각의 기능적 범위나 정책적 특성에 따라 지역 추적성과 전역 추적성을 모두 제공하지 못한다는 것을 알 수 있다. 작업공간에서의 지역 추적성은 공식화된 형상항목에 대해서 자유로운 변경이 가능하도록 하고 변경의 결과에 대한 투명성을 제공함으로써 업무 효율성 및 신뢰성을 보장할 수 있는 근거가 된다. 이러한 지역 추적성이 SCM의 전역 추적성, 즉 공식화된 문서의 변경 이력과 통합될 때 시스템은 더욱 확장된 추적성을 제공할 수 있으며 이를 통해 개발 생산성과 신뢰성을 보다 향상시킬 수 있게 된다.

<표 2>는 본 논문을 통해 핵심적으로 다룬 기능적 요소들을 바탕으로 기존 SCM 도구와 제안한 시스템을 비교한 결과이다. <표 2>의 비교 항목들 중 '개인 작업공간에서 형상항목에 대한 자유로운 변경 보장'은 일반적인 프로세스 중심 도구들이 갖고 있는 작업 제한의 한계를 지적하기 위한 것으로, 생산성 중심 도구나 본 연구에서 제안한 시스템은 보다 나은 개발 생산성을 위해 지원하는 기능이다.

'형상제어 프로세스 지원'과 '개발 프로세스 관리 지원'은 기본적으로 프로세스 중심 도구들이 갖춰야 할 핵심적인 기능으로, SCM의 관리적 요구사항을 만족시키고 있는가를 평가하는 항목이다.

'지역 추적성과 전역 추적성의 통합'은 앞선 두 항목과 관계된 것으로 작업공간에서의 자유로운 변경과 SCM의 형상제어 프로세스를 동시에 만족시키기 위한 것이다.

'시각화된 버전 모델'은 작업공간이나 SCM 환경에서의 변경 과정을 시각적으로 표현함으로써 보다 수월한 변경 추적을 가능하게 한다. 일반적으로 버전 그래프와 같은 모델이 주로 사용되는데, 버전 그래프는 다양한 흐름의 분기, 병합, 태그(라벨)와 같은 요소들을 표현하기에 매우 적합한 모델이다.

〈표 2〉 제안한 시스템과 타 시스템들의 기능성 비교

지원기능 \ 제품군	Source Integrity	StarTeam	Synergy/CM	CCC/Havest	PVCS Dimensions	ClearCase (UCM)	제안한 시스템
개인 작업공간에서 형상항목의 자유로운 변경 보장	○	○	×	×	×	×	○
형상제어 프로세스 지원	×	×	○	○	○	○	○
개발 프로세스 관리 지원	×	×	○	○	○	○	○
지역 추적성과 전역 추적성의 통합	×	×	×	×	×	○	○
시각화된 버전 모델	○	○	○	×	○	○	○
산출물 공유와 동시작업 지원	×	×	×	×	×	Limited	○

마지막으로 '산출물 공유와 동시작업 지원'은 여러 사용자들이 특정 산출물을 공유하고 각자의 작업공간에서 동시에 작업을 수행할 수 있는지 여부에 대한 평가 항목이다. 일반적인 SCM 도구들은 형상제어 프로세스와 상관없이 특정 사용자가 먼저 해당 산출물을 변경하는 경우에 다른 사용자들에게는 변경을 허용하지 않는 반면, 제안한 시스템은 동시작업을 허용하고 이에 대해서 적절한 충돌해결 기법을 제공하기 때문에 보다 나은 개발 생산성을 지원하게 된다.

지금까지 설명한 비교 항목들을 기준으로 봤을 때 제안한 시스템은 기존 SCM 도구들이 갖고 있는 부분적인 한계를 모두 만족한다고 평가할 수 있다. 가장 중요한 것은 확장된 추적성 정보를 이용한 개발 생산성의 향상에 있다고 볼 수 있다. 본 논문에서 중요하게 다루지는 않았지만 제안한 시스템이 다른 도구들에 비해 부족한 부분으로는 Build 자동화 기능이나 통합개발환경(IDE)과의 연계, 이기종 환경에 대한 지원 등이 있다. 이러한 부분들은 앞으로 더욱 보완해 나갈 것이며, 더욱 성숙된 개발 환경을 지원할 수 있도록 노력할 것이다.

7. 결론 및 향후 연구

본 논문은 일반적인 SCM이 지향하는 목적을 준수하면서 보다 높은 차원의 개발 생산성을 제공하기 위한 산출물의 추적성 향상 방안을 제시하였다. SCM 시스템은 공식화된 형상항목에 대해서 조직 내 모든 구성원들에게 전역 추적성을 제공하며, 개인 작업공간에서는 각자가 산출물에 대한 변경을 관리함으로써 지역 추적성을 유지한다. 이러한 전역 추적성과 지역 추적성을 통합하여 관리할 때 보다 일관된 변경 업무를 지원할 수 있으며 개발 생산성의 향상을 뒷받침 할 수 있다.

형상항목의 수정버전은 SCM 시스템 상에서 전역 추적성을 나타내기 위한 기본적인 요소로서 이를 작업공간의 산출물 버전에 연결시키는 것은 앞서 언급한 전역 추적성과 지역 추적성의 통합을 위한 방안이다. SCM 시스템에서 기준선 문서로 공식화된 형상항목은 그것과 연결된 산출물의 변경에 있어 하나의 기준을 제공한다. 이전에 공식화된 형상항목의 수정버전이 산출물의 어떤 버전과 연결되어있는지를 나타내는 것은 향후에 있을 변경에 논리적 근거를 제공해줄 수 있으며 그 동안의 변경흐름을 정확하게 추적할 수 있도록 해준다.

본 연구를 바탕으로 산출물들 간의 연관성을 밝힐 수 있는 더 넓은 의미의 추적성을 제공하기 위한 연구가 수행되어야 한다. 본 연구에서의 추적성이 단일 산출물과 형상항목 간의 관계를 통해 얻어진 것이었다면 앞으로는 여러 산출물들 간의 추적성과 나아가 프로젝트 전반에 걸친 산출물들의 추적성을 관리할 수 있는 방안에 대해 연구가 필요하다. 소프트웨어 개발에서 완성된 제품이 만들어지기까지는 매우 다양하고 많은 산출물들이 발생하기 때문에 모든 산출물들의 연관성을 규명하고 변경에 대한 추적성을 관리하는 일은 매우 복잡하고 까다로운 일이나 갈수록 크고 복잡해지는 소프트웨어의 품질과 생산성을 보장하기 위해서는 반드시 수행해야 할 과제이다.

참 고 문 헌

[1] Carey Schwaber, "The Forrester Wave™: Process-Centric Software Configuration Management, Q4 2005", Forrester Research, Inc., November, 14, 2005.

[2] "Glossary," ASME Boiler and Pressure Vessel Code, Section III, Article NCA-9000.

[3] http://www.iso.org/iso/iso_catalogue/management_standards/iso_9000_iso_14000/iso_9001_2008.htm

[4] Gotel, O. and Finkelstein, A., "An Analysis of the Requirements Traceability Problem", Proc. of the IEEE International Conference on Requirements Engineering (ICRE), 1994.

[5] Spanoudakis, G., "Plausible and Adaptive Requirement Traceability Structures", in the proceedings of SEKE'02, July, 15-19, 2002.

[6] Ramesh, B. and Edwards, M., "Issues in the Development of a Requirement Traceability Model", IEEE In Proc. of the 1st Intl. Symposium on Requirements Engineering, San Diego, CA, January, 1993.

[7] Murray, L., Griffiths, A., Lindsay, P. and Strooper, P., "Requirements Traceability for Embedded Software - an Industry Experience Report", www.itee.uq.edu.au/~pal/SVRC/tr00-41.pdf, 2000.

[8] "IEEE Guide to Software Requirements Specification", IEEE 830-1998.
[9] Bashir, M. F., Qadir, M. A., "Traceability Techniques: A Critical Study", INMIC'06, IEEE Multitopic Conference, pp.265-268, December, 23-24, 2006.
[10] Berczuk, S. P. and Appleton, B., "Software Configuration Management Patterns, Effective Teamwork, Practical Integration", Addison Wesley, 2002.
[11] Kögel, M., "Towards Software Configuration Management for Unified Models", CVSM'08, May, 17, 2008.
[12] Conradi, R. and Westfechtel, B., "Version Models for Software Configuration Management", ACM Computing Surveys 30, 2 pp.232-282, 1998.
[13] Estublier, J., Clemm, G., et al., "Impact of Software Engineering Research on the Practice of Software Configuration Management", ACM Trans on Software Engieering and Methodolog, Vol.14, No.4, pp.383-430, Oct., 2005.
[14] Mohan, K., Xu, P. and Ramesh, B., "Improving the Change Management Process", Communications of the ACM, Vol. 51, No.5, pp.59-64, May, 2008.
[15] Sarma, A., Redmiles, D., et al., "Empirical Evidence of thebenefits of workspace awareness in software configuration management", Proceedings of the 16th ACM, pp.113-123, 2008.
[16] Junqueira, D., Bittar, T. and Fortes, R., "A fine-grained and flexible version control for software artifacts", SIGDOC'08, September, 22-24, 2008.
[17] Estublier, J. and Garcia, S., "Process Model and Awareness in SCM", SCM 2005, September, 5-6, 2005.
[18] http://www.mks.com/products/sie
[19] http://www.borland.com/us/products/starteam/index.html
[20] http://www-01.ibm.com/software/awdtools/synergy/
[21] http://www.ca.com/us/products/product.aspx?ID=255
[22] http://www.serena.com/products/pvcs/index.html
[23] http://www-01.ibm.com/software/awdtools/clearcase

김 대 엽

e-mail : kdymn2@cnu.ac.kr
2005년 충남대학교 정보통신공학부(학사)
2005년~현 재 충남대학교 컴퓨터공학과 석·박사 통합과정
관심분야 : 소프트웨어 형상관리, 추적성 관리, 컴포넌트 기반 개발방법론 등

윤 청

e-mail : cyoun@cnu.ac.kr
1979년 서울대학교 물리학과(학사)
1983년 Illinois State University 전산학과 (석사)
1988년 Northwestern University 전산학과 (박사)
1983년~1985년 Wayne State University 전산학과 전임강사
1985년~1987년 Northwestern University 전산학과 전임강사
1988년~1993년 Bell Communications Research 선임연구원
1993년~현 재 충남대학교 전기정보통신공학부 교수
관심분야 : 소프트웨어공학, 객체지향 개발방법론 등