http://dx.doi.org/10.3745/KIPSTD.2011.18D.6.451

소프트웨어 형상관리 시스템과 개인 작업공간 통합 기반 산출물 의존 관계 추적성 개선 기법

김 대 엽[†] · 윤 청[††]

요 약

지속적으로 변경되는 소프트웨어 산출물들의 변경 이력과 의존 관계를 추적하는 능력은 소프트웨어 시스템 개발 프로세스에서 확인 및 검증, 시험과 같은 품질관리 활동들을 지원하는 중요한 요소로 인식되고 있다. 일반적인 소프트웨어 형상관리 시스템들은 산출물에 대한 추적 정보를 형상관리 시스템 내에서만 제공하고 있으며 개인의 작업공간에서 발생한 개별적인 변경까지 추적하는 기능은 지원하지 못하고 있다. 본 연구를 통해 형상관리 시스템과 개인의 작업공간을 통합함으로써 형상항목에 대한 변경뿐만 아니라 개인의 작업공간에 존재하는 산출물들의 변경 이력까지 추적할 수 있도록 하였다. 더 나아가 소프트웨어 산출물들이 동시에 생성되어 하나의 형상항목에 포함되는 경우 이들 산출물들의 버전 링크를 식별함으로써 산출물들 사이의 의존 관계를 추적할 수 있도록 하였다. 형상관리 시스템과 개인 작업공간의 통합을 위해 형상항목의 수정버전(revision)을 작업공간에 존재하는 산출물들의 버전에 태그로 연결시켰으며, 통합 변경 과정에서 산출물들에 대한 변경 이력 및 의존 관계의 효율적인 추적이 가능하도록 하였다.

키워드 : 소프트웨어 형상관리, 작업공간, 추적성, 버전 관리

Traceability Enhancement Technique for Dependency Relations of Software Artifacts based on the Integration of Software Configuration Management System and Personal Workspace

Dae-Yeob Kim[†] · Cheong Youn[††]

ABSTRACT

The ability to trace change history and dependency relations of software artifacts which are continuously changed has been recognized as an important factor that support quality management activities such as verification, validation, and testing in software system development process. General software configuration management systems provide tracing information for artifacts only within the configuration management system, and it does not go further to changes that occur within personal workspace. This paper provides a solution that helps tracing down not only changes of configuration items but also change history of artifacts existent in personal workspace through the integration of configuration management system and personal workspace. Furthermore, in cases of artifacts which are included in a configuration item, this paper provides a solution that support tracing dependency relations between the artifacts by identifying their version links. For the integration of configuration management system and personal workspace, a revision of configuration item is connected to the artifact's version of the workspace by the tagging mechanism, and traceability for change history and dependency relations of artifacts can be managed more effectively through integrated change process.

Keywords : Software Configuration Management, Workspace, Traceability, Version Control

1. 서 론

추적성은 재료, 부품, 혹은 최종 제품에 대한 이력과 서로 다른 요소들 간의 관계를 식별하고 추적할 수 있는 능력으로 정의된다. 추적성은 제품의 품질을 유지하고 문제 발생 시 그에 대한 정확한 분석을 수행하는데 도움을 준다. 추적성의 관리가 제대로 이루어지기 위해서는 개발에서 인도까

※ 이 연구는 2009년도 충남대학교 학술연구비에 의해 지원되었음.
† 준 회 원 : 충남대학교 컴퓨터공학과 대학원생
†† 정 회 원 : 충남대학교 컴퓨터공학과 교수(교신저자)
논문접수 : 2011년 4월 26일
수 정 일 : 1차 2011년 6월 11일, 2차 2011년 6월 29일
심사완료 : 2011년 6월 29일

지 전 공정과정에 걸쳐 관련 절차가 수립되어야 하며, 제품을 구성하는 요소들의 상태와 그것들의 관계에 대한 정보들이 유지되어야 한다. ISO 9000 시리즈의 "제품 식별 및 추적성(product identification and traceability)" 규격에서도 제품의 식별과 추적에 대한 정보가 정확하게 기록되고 유지될 때 고객에게 품질을 보증할 수 있다고 강조하고 있다[1]. 이미 전 세계 여러 산업분야에서 추적성 관리의 필요성을 인식하고 있으며, 관련 절차나 기법들을 적용하고 있다.

소프트웨어 분야에서의 추적성은 "소프트웨어 시스템이 개발되는 동안 생성된 산출물들을 연결시키는 능력"으로 정의되고 있으며, 소프트웨어 시스템의 개발 및 유지보수 과정에서 품질보증과 관련된 여러 활동들(확인 및 검증, 시험, 영향분석 등)을 지원하는 중요한 요소로 인식되고 있다[2]. 소프트웨어 추적성은 크게 요구사항 기반의 추적성과 변경 이력 기반의 추적성으로 구분할 수 있다.

요구사항 기반의 추적성은 요구사항 분석 이전과 이후에 대한 것으로 구분된다. 요구사항 분석 이전의 추적성은 이해 당사자(stakeholder)들의 요구와 식별된 요구사항에 대한 관계를 나타냄으로써 요구사항 검증(validation)[1]과 같은 활동을 지원한다. 요구사항 분석 이후의 추적성은 요구사항과 그것을 기반으로 생성되는 다른 산출물들 간의 관계를 나타냄으로써 확인(verification)[2]이나 시험(testing)과 같은 활동을 지원한다[2-9].

변경 이력 기반의 추적성은 산출물들의 변경에 대한 이력 정보를 통해 소프트웨어 시스템의 진화 과정을 추적할 수 있도록 지원한다. 변경 이력 기반의 추적성은 소프트웨어 형상관리 내에서의 추적성과, 개인 작업공간 내에서의 추적성으로 구분될 수 있다.

소프트웨어 형상관리 시스템은 기준선(baseline)을 구성하는 형상항목의 변경에 대해 공식적인 절차를 요구하며, 개발자들은 형상항목의 변경에 대한 정보(수정버전, 변경 담당자, 변경 사유, 변경 시점 등)를 공유한다. 기준선 문서로 동결된 형상항목의 변경 내용을 모든 개발자들이 공유하고 추적하는 것은 전역 추적성(global traceability)에 해당된다.

형상관리 시스템과는 달리 개인 작업공간에서는 형상항목으로 식별된 산출물에 대해 자유로운 변경이 이루어진다. 개발자들은 일반적으로 버전 관리 기법을 통해 산출물에 대한 각자의 변경 이력을 유지하고 추적한다. 버전 관리에서 산출물에 대한 변경 이력은 버전 그래프와 같은 모델로 표현될 수 있으며, 개발자들은 이러한 모델을 통해 과거의 작업 내용을 쉽게 추적할 수 있다. 버전 관리 기법을 이용한 개인적인 변경 이력의 추적은 지역 추적성(local traceability)에 해당된다.

형상항목이 기준선 문서로 공식화된 이후에도 개발자들은 그것과 연결된 산출물들에 대해 자신의 작업공간에서 자유롭게 변경을 수행할 수 있다. 그 결과 형상관리 시스템에 반영된 산출물에 대하여 개인 작업공간에서는 추가적인 새로운 버전들이 생성된다. 형상관리 시스템에 반영된 산출물에 대해서 새롭게 추가된 버전들을 다른 개발자들이 공유하고 추적할 수 없다면 서로 중복된 작업을 수행할 가능성이 있다. 중복 작업으로 인해 개발자들은 시간과 노력을 낭비할 수 있으며 이는 곧 개발의 효율성과 생산성에 좋지 않은 영향을 끼칠 수 있다. 이러한 현상은 형상관리 시스템을 통하여 개발자들 사이에 공유할 수 있는 전역 추적성과 개발자들의 고유 영역에서 관리되는 지역 추적성의 관리가 분리되어 이루어질 때 나타나게 된다.

이와 같은 부작용을 해소하기 위해 형상관리 시스템과 개인 작업공간의 통합이 요구된다. 형상관리 시스템과 개인 작업공간의 통합은 형상항목이 기준선문서로 공식화될 때 생성되는 수정버전을 개인 작업공간에 존재하는 산출물의 버전에 태그로 연결함으로써 이루어진다. 개발자들은 태그 정보를 통해 형상관리 시스템에 반영된 산출물에 대해서 새로운 변경 작업이 얼마나 진행되었는지 추적할 수 있다. 형상관리 시스템에 반영된 산출물에 대해 자신이 원하는 작업을 수행하기 전에 다른 개발자들이 수행한 작업 결과를 미리 파악함으로써 중복 작업으로 인해 야기되는 시간과 비용의 중복 투자를 예방할 수 있으며 시스템의 일관성을 유지할 수 있다.

또한 소스코드와 같이 여러 개의 산출물들이 한 형상항목에 포함되는 경우 이들 산출물들 사이의 의존 관계가 형성되고, 그 의존 관계는 산출물들의 버전 링크를 식별함으로써 추적할 수 있다. 버전들의 링크를 통해 산출물들의 의존 관계를 추적하는 것은 보다 효율적이고 일관성 있는 작업을 수행하기 위해 필요하다. 형상항목이 기준선 문서로 공식화되면 그 것과 연결된 산출물들의 버전에는 동일한 수정버전 태그가 생성되므로 형상항목의 수정버전을 이용하면 산출물들의 버전 링크를 식별할 수 있다.

본 연구를 통해 형상항목의 수정버전이 산출물들의 버전에 태그로 연결되는 과정을 자동화하고, 수정버전을 이용한 산출물들의 버전 링크 식별 기능을 제공함으로써 빠르고 간편한 추적성 관리가 가능하도록 하였다. 이는 수동적인 추적성 관리에서 나타나는 비효율적인 문제들을 해소하고 변경에 대한 일관성을 유지할 수 있도록 지원하기 위한 방안이다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 추적성 관리에 대한 기존 연구들을 간략하게 소개한다. 3장에서는 단일 산출물의 진화에 대한 추적 정보의 표현 방법을 설명하고, 4장에서는 한 형상항목에 여러 산출물들이 포함되는 경우 버전 링크를 통해 산출물들 간 의존 관계를 나타내는 방법에 대해 설명한다. 5장에서는 추적성 관리 도구가 갖춰야 할 요구사항들을 기준으로 본 논문의 통합 환경과 기존 형상관리 시스템들을 비교 평가한다. 마지막으로 6장에서 본 논문의 결론을 맺는다.

1) 고객이 원하는 올바른 제품을 만들었는지, 요구사항 분석의 결과가 고객이 바라는 요구를 반영한 만족할만한 것인지를 검사하는 것. 관련 기법으로는 시제품 개발(prototyping), 기술적인 검토(technical review), 시뮬레이션(simulation) 등이 있음.

2) 개발 주기 상에서 현 단계의 산출물이 바로 이전 단계의 산출물과 논리적인 일관성을 유지하고 있는가를 확인하는 과정으로, 고객이 바라는 대로 소프트웨어가 만들어졌음을 확신할 수 있도록 수행하는 활동. 확인 행렬(verification matrix)과 같은 형태로 그 결과를 기록할 수 있음.

2. 관련 연구

지난 수년간 소프트웨어 및 시스템 공학 연구자들은 추적성의 다양한 측면들을 다루기 위해 많은 기법들을 연구해왔다. 소프트웨어 추적성에 대한 연구들은 주로 추적 관계의 유형, 추적성 관리의 자동화 방안, 추적 관계를 표현하기 위한 기법들과 관련되어있다. 추적성에 대한 기존 연구들을 간략하게 정리하면 <표 1>과 같다.

2.1 추적 관계의 유형

산출물들 간 추적 관계의 유형은 여러 연구 논문에서 다양하게 정의되어 왔다[8-11]. 이 장에서는 본 논문에서 다루는 진화와 의존 관계에 대해 초점을 두어 설명한다.

지속적인 변경 작업에 의해 산출물에 여러 버전들이 생성되는 경우 산출물의 진화에 대한 추적 관리가 요구된다. 산출물의 진화에 대한 관리는 과거의 개발 흐름이나 변경의 배경 등을 이해하고 추적하기 위해 필요하다[12]. 진화에 대한 추적성이 관리되지 않으면 산출물의 현재 상태가 어떠한 의도나 목적에 의해 만들어졌는지 명확하게 이해할 수 없으며, 이로 인해 효율적인 변경 관리가 어려워지게 된다.

특정 산출물의 존재가 다른 산출물의 존재에 의존하는 경우나 특정 산출물의 변경이 다른 산출물에 영향을 주는 경우 두 산출물들 사이에는 의존 관계가 형성된다. 예를 들어, 요구사항 명세서는 설계 명세서가 존재하기 위해 필요한 산출물이며 요구사항 명세서의 변경은 설계 명세서의 변경을 야기할 수 있으므로 이 두 산출물은 서로 의존 관계에 놓여 있다고 볼 수 있다. 이러한 의존 관계는 주로 시간의 흐름에 따라 생성된 산출물들 사이에서 나타난다. 시간의 흐름에 따라 서로 다른 개발 단계에서 생성된 산출물들의 의존 관계를 추적하는 능력을 수평 추적성(horizontal traceability)이라 부른다[11, 13].

반면 한 개발 단계(혹은 동일 모델)에서 동시에 생성된 산출물들 간의 의존 관계를 추적하는 능력을 수직 추적성(vertical traceability)이라 부른다. 수직 추적성을 지원하는 대부분의 기법들은 서로 다른 요구사항들 간의 의존 관계를 나타내거나 소스 코드와 같이 동시에 생성된 산출물들 사이의 의존 관계를 나타내는데 초점을 맞추고 있다[14-17]. 수평 추적성과 수직 추적성 모두 시스템의 개발 및 유지보수 과정에서 일관성 있는 작업을 진행하는데 필요하다. 사용자들은 이러한 추적성을 통해 영향 분석이나 확인 및 검증과 같은 품질 관리 활동들을 효율적으로 수행할 수 있다.

2.2 추적성 관리의 자동화 방안

추적성 관리의 자동화 방안은 사용자들의 수고를 덜기 위한 목적으로 연구되어 왔다. 수동적인 기법들(스프레드시트, 데이터베이스, 워드 프로세싱 소프트웨어 등을 이용한 분석 및 유지)은 분석가나 개발자들에게 많은 노력을 요구한다.

〈표 1〉 추적 관계의 유형과 추적성 지원에 대한 기존 연구들의 분류

추적 관계의 유형 분류	진화 (evolution)	산출물의 진화 과정을 나타내는 것으로, 흔히 동일 산출물에 여러 개의 버전이 존재하는 경우에 해당
	의존 (dependency)	한 산출물의 존재 및 변경이 다른 산출물에 영향을 주는 경우
	일반화 (generalization)	시스템의 복잡한 요소를 컴포넌트 단위로 세분화하거나, 여러 요소들을 하나의 새로운 요소로 조합하는 경우
	만족 (satisfaction)	특정 산출물의 내용이 다른 산출물의 요구나 조건에 부합하는 경우
	중복 (overlap)	여러 개의 산출물들이 시스템 내에서 공통적인 속성에 의해 관련되어 있는 경우
	충돌 (conflicting)	서로 다른 산출물들이 비 일관적인 내용으로 인해 충돌을 맺는 경우
	근거 (rationalization)	산출물의 생성과 진화에 대한 이유 혹은 근거를 나타내고 유지하기 위해 사용
	기여 (contribution)	요구사항 관련 산출물과 이해 당사자(stakeholder)들 간의 관계를 나타내기 위해 사용
추적성 관리의 자동화 방안	요구사항 기반 추적성의 자동화	주로 정보검색(IR: Information Retrieval) 기법(확률적 정보검색 혹은 벡터공간 정보검색으로 구분)이나 Latent Semantic Indexing(LSI) 기법, 규칙 기반(rule-based) 추적 기법 등이 사용됨
	변경 이력 기반 추적성의 자동화	릴리즈 이력 혹은 변경 이력 정보를 이용해 산출물들의 추적 관계를 식별
추적 관계의 표현 기법	마크업(mark-up) 방식	각 산출물에 의미 있는 식별자를 부여함으로써 다른 문서들에 대한 의존성을 나타냄
	하이퍼미디어(hypermedia) 방식	이질적인 도구들에 의해 생성된 산출물들의 추적 관계를 기록, 유지, 탐색할 수 있도록 지원
	이벤트 기반(event-based) 방식	사건 통지 메커니즘을 이용하여 산출물들 간 변경의 영향을 추적할 수 있도록 지원

분석가들은 추적성 관계를 수동적으로 구성하고 유지해야 할뿐만 아니라 산출물들이 어느 정도의 수준에서 추적되어야 하는지 까지도 결정해야 하는 어려움을 겪는다. 수동적인 추적성 관리 기법의 어려움을 해소하기 위해 그 동안 다양한 형태의 자동화 기법들이 제시되어왔다. 요구사항 기반의 추적성 관리를 자동화하는데 있어서는 주로 정보검색(information retrieval)과 같은 기법들이 사용된다[18-20]. 정보검색 기법은 특정 질의어를 사용하여 문서들 사이의 유사성을 찾아낸다. 질의어와 문서들의 연관성은 확률적(probabilistic) 정보검색 기법 및 벡터 공간(vector space) 기법 등에 의해 식별된다. 정보검색 기법의 일종인 유사도 측정 알고리즘(similarity calculation algorithms)은 요구사항들 간의 유사도를 계산함으로써 추적 관계의 분석을 지원한다[21]. 정보검색 기법을 이용한 추적성 관리의 효율성은 recall[3]과 precision[4]의 표준 메트릭스를 통해 평가된다.

변경 이력에 기반 한 추적성 관리 도구들은 버전 관리 시스템으로부터 변경 패턴들(변경 시간, 변경 담당자에 대한 정보 등)을 획득함으로써 산출물들의 의존 관계를 분석하도록 지원하지만, 충분히 자동화된 기능들을 제공하지는 못하고 있다. 또한 기본적인 버전관리 시스템에 국한된 추적성을 다루고 있다는 것도 이러한 기법들의 한계로 지적될 수 있다[22-27].

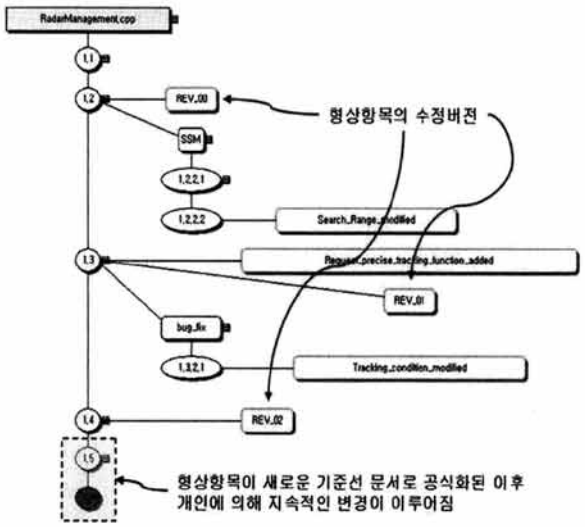
2.3 추적 관계의 표현 기법

추적 관계의 표현 기법에는 마크업 방식, 하이퍼미디어 방식, 이벤트 기반 방식 등이 있다. 마크업 방식은 각 산출물에 의미 있는 식별자를 부여함으로써 다른 문서들에 대한 의존성을 나타낸다[6, 12, 28]. 본 논문에서 형상관리 시스템과 개인 작업공간을 통합하는 데 사용된 태깅 메커니즘이 이 방식에 해당된다. 식별자로 사용된 태그(형상항목의 수정버전) 정보를 통해 산출물들의 변경 이력은 물론 의존 관계를 나타낼 수 있다. 하이퍼미디어 방식은 이질적인 도구들에 의해 생성된 소프트웨어 산출물들의 추적 관계를 기록, 유지, 탐색할 수 있도록 지원한다[29]. 이벤트 기반 방식은 이벤트 통지 메커니즘을 사용하여 산출물들 간 변경의 영향을 추적할 수 있도록 지원한다. 이 기법은 미리 정해진 산출물들의 의존 관계를 기반으로 하며, 특정 산출물에 변경이 발생할 경우 그것과 관련된 다른 산출물들이 영향을 받을 수 있음을 통보한다[30].

3. 단일 산출물의 진화 과정 추적

개발자들은 개인 작업공간에서 버전 관리 기법을 통해 산출물들의 변경 이력을 관리한다. 작업공간에서의 버전 관리는 워크스페이스(workspace) 시스템을 통해 이루어진다. 워크스페이스 시스템은 작업공간을 독립적으로 구성하여 개인의 자유로운 변경 작업을 지원한다. 워크스페이스에 존재하는 산출물들은 각각 변경 이력을 나타내는 버전 정보를 갖고 있으며, 개발자는 원하는 산출물의 버전 정보를 (그림 1)과 같이 버전 그래프로 확인할 수 있다. 버전 그래프는 각 산출물들이 그 동안 어떠한 진화 과정을 거쳐 왔는지 직관적으로 이해할 수 있도록 돕는다[31-32].

버전 그래프는 산출물의 버전 정보뿐만 아니라 분기(branch)나 태그(tag)와 같은 정보들도 포함한다. 분기는 특정 버전에 대해 버그를 수정하거나 다른 응용 분야에 적용하기 위해 별도의 흐름으로 개발을 진행하고자 하는 경우 생성할 수 있다. 분기에 의해 진행된 변경 결과는 필요할 경우 다시 기본 흐름에 병합(merge)될 수도 있다. Berczuk는 자신의 저서에서 분기를 기본 흐름(main line)에 영향을 주지 않는 개인적인 code line이라 표현하고 있으며, 개발의 효율성과 일관성을 위해 사용될 수 있다고 설명한다[33]. 태그는 각 버전에 대해 특별한 의미를 나타내고자 할 때 사용될 수 있다. 태그를 통해 사용자들은 각 버전의 생성 배경이나 결과를 개략적으로 이해할 수 있다.



(그림 1) 산출물의 버전 그래프

산출물의 변경 이력 가운데 형상관리 시스템에 반영된 버전을 구분하기 위해서 형상항목의 수정버전을 이용하였다. 형상항목의 수정버전은 형상항목이 기준선 문서로서 형상관리 서버에 등록될 때마다 새롭게 생성된다. 이 때 형상항목에 연결된 산출물의 버전(형상관리 시스템에 반영되는 버전)에 형상항목의 수정버전이 태그로 연결된다. 결과적으로 형상항목을 구성하는 각 산출물들의 어느 버전이 시스템 릴리즈와 관련되어 있는가를 추적할 수 있게 된다.

본 연구에서 개발한 형상관리 시스템은 형상항목의 수정버전을 'REV_XX'와 같은 형태로 정형화 하였다. (그림 1)의 버전 그래프에서 형상항목의 수정버전이 산출물의 버전 그래프에 태그로 연결된 모습을 볼 수 있다. 버전 그래프에서 태그는 개인 작업공간과 형상관리 시스템의 통합에 의해 자

3) 정보검색 결과에 대한 정확도를 나타내는 수치. 검색된 결과 중에서 실제 질의와 관련된 문서가 얼마나 포함되어 있는지를 나타냄.

4) 정보검색 결과에 대한 재현률을 나타내는 수치. 실제로 질의와 관련된 문서를 얼마나 많이 검색했는지를 나타냄.

동적으로 생성된 것과 사용자가 임의로 생성한 것으로 구분된다. 정형화된 수정버전과 자동화된 태그 생성 기능은 사용자들 사이에서 발생할 수 있는 혼란을 막고 업무의 생산성을 향상시킬 수 있다.

(그림 1)에서 볼 수 있듯이 산출물의 버전 '1.4'가 형상항목의 수정버전 'REV_02'에 반영된 이후에도 개인 작업공간에서는 지속적으로 변경이 진행될 수 있으며 그 결과로 새로운 버전들이 생성된다(버전 '1.5'와 '1.6'의 경우). 이 경우, 사용자들은 'REV_02'라고 하는 태그를 통해 산출물이 형상관리 시스템에 최종적으로 반영된 이후 새로운 변경 작업이 얼마나 진행되고 있는지를 직관적으로 추적할 수 있게 된다.

4. 한 형상항목 내 산출물들의 의존 관계에 대한 추적

개인 작업공간의 산출물들은 모듈(module)이라는 단위로 그룹화 되어 형상항목에 연결된다. 모듈은 개인 작업공간에서 디렉토리 구조를 띠고 있으며, 형상관리 시스템은 이 디렉토리 구조를 참조함으로써 산출물들의 변경 결과를 서버에 등록한다. 한 모듈 내에는 한 개의 산출물만이 포함될 수도 있고 여러 개의 산출물들이 포함될 수도 있다. 본 장에서는 모듈 내의 산출물이 여러 개인 경우, 즉 한 형상항목에 여러 산출물들이 연결될 때 이들 산출물들의 의존 관계를 추적하는 방법에 대해 설명하고자 한다.

한 형상항목에 포함된 산출물들의 의존 관계를 추적하기 위해서는 형상관리 시스템에 동시에 반영된 산출물들의 버전 링크를 식별하는 과정이 필요하다. 버전 링크의 식별을 통한 산출물들의 의존 관계 추적을 위해 형상항목의 수정버전이 질의어로 사용된다. 형상항목이 형상관리 시스템 상에서 공식화될 때 형상항목의 수정버전이 산출물들의 버전 그래프에 태그로 연결되기 때문에 태그 이름을 질의어로 사용하면 수정버전과 관련된 산출물들의 버전 링크를 식별하는 것이 가능하다.

(그림 2)는 한 형상항목에 포함된 산출물들에 대해서 형상항목의 수정버전 'REV_01'이 태그로 연결된 버전들을 식별한 결과를 보여준다. 화면의 '태그/분기' 탭을 보면 산출물들의 버전이 'REV_01'라는 태그와 관련되어 있음을 알 수 있다. 결과적으로 한 형상항목에 포함된 산출물들에 대해서 버전 링크를 식별하고 이를 통해 산출물들의 의존 관계를 추적할 수 있게 된다. (그림 3)은 버전 정보에 기반 한 산출물들의 의존 관계가 실제로 어떠한 의미를 갖는지 버전 그래프를 통해 표현한 것이다. (그림 2)의 버전 목록은 (그림 3)과 같이 동일한 수정버전에 의해 연결된 버전들을 나타내고 있다. 개발자들은 이러한 버전 링크를 통해 기준선 문서를 구성하는 산출물들이 어떠한 의존 관계를 맺고 있는지 파악할 수 있다. 이러한 형태의 의존 관계는 시스템에 대한 릴리즈 관리나 유지보수 활동을 수행하는데 유용하게 사용될 수 있기 때문에 매우 중요하다. <표 2>는 (그림 3)에서 형상항목의 수정버전에 따라 식별될 수 있는 산출물들 간 의존 관계를 정리한 것이다.

(그림 2) 태그 정보를 이용한 버전 링크 검색 결과

〈표 2〉 형상항목 수정버전에 의한 산출물들의 의존 관계

산출물 / 수정버전	MFRData.cpp	TargetData.h	MFRDataTable.h
REV_00	1.1	1.2	1.1
REV_01	1.2	1.3	1.2
REV_02	1.4	1.6	1.5

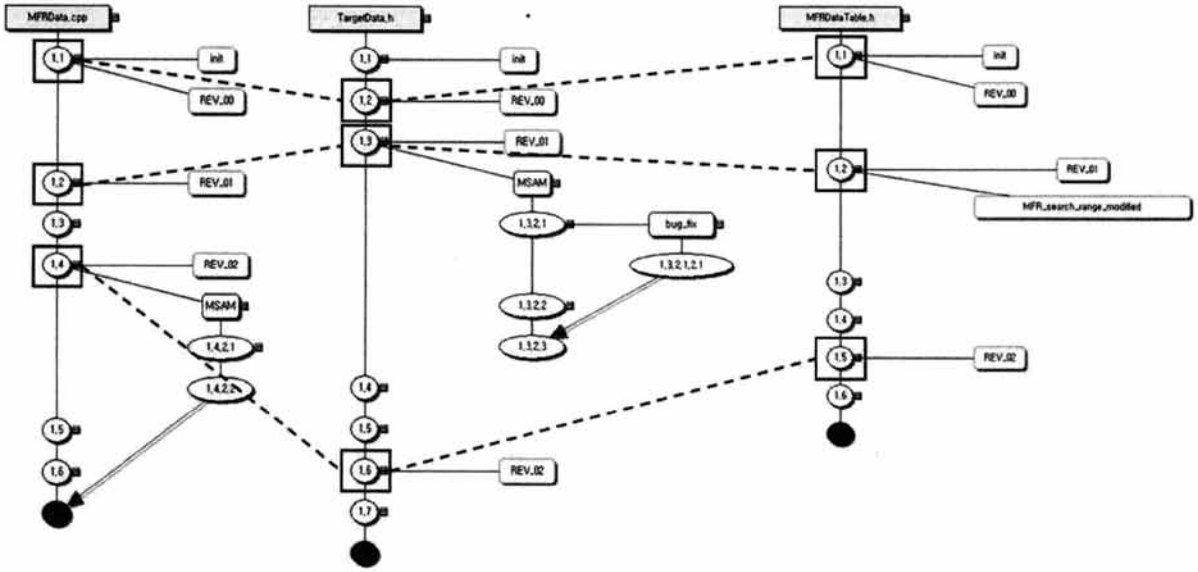
5. 추적성 관리도구의 요구사항과 만족도 평가

본 장에서는 추적성 관리에 대한 고수준의 요구사항들을 기준으로 본 논문에서 제시하는 통합 환경과 기존 형상관리 시스템들을 비교 평가하고자 한다. 비교 대상 시스템으로 IBM Rational ClearCase(UCM), Borland StarTeam, AccuRev Enterprise Edition, Perforce Software, Serena Software PVCS, CA Harvest를 선정하였다. 이들 형상관리 시스템들은 상용화된 제품으로서 전 세계적으로 높은 시장 점유율을 차지하고 있다.

5.1 추적성 관리도구의 요구사항에 대한 정성적 평가

Marcus et al.은 추적성 관리 도구들이 갖춰야 할 고수준의 요구사항들을 <표 3>과 같이 제시하고 있다[34]. <표 3>에는 추적성 관리 도구가 갖춰야 할 고수준의 요구사항들과 함께 본 논문에서 제시하는 통합 환경이 각각의 요구사항을 어떻게 만족하는지 간략하게 기술되어 있다. <표 4>는 앞서 언급한 형상관리 시스템들이 <표 3>의 요구사항을 만족하는지 여부를 나타내고 있다.

추적성의 시각화에 대한 요구사항은 대부분의 시스템들이 version graph, version tree, code line 등과 같은 모델을 이용하여 지원한다. Harvest의 경우 특정 산출물의 변경 이력을 추적하기 위해서는 텍스트 형태의 로그 기록을 조회해야 하므로 다소 불편함이 있다.



(그림 3) 버전 그래프를 통해 본 산출물들 간의 의존 관계

〈표 3〉 추적성 관리 도구의 요구사항 및 제안한 통합 환경의 만족여부 평가

Req. no.	요구사항	제시한 통합 환경의 만족여부 평가
1	추출 방법에 상관없이 다양한 산출물들에 대한 추적성을 시각화하고 저장할 수 있어야 한다.	산출물 작성에 사용된 도구의 종류에 상관없이 산출물들에 대한 추적 정보 제공
2	추적성 관계식별 도구와 통합될 수 있어야 한다.	제시한 환경 자체가 추적성 관계의 식별을 지원
3	추적성 관계에 대해 사용자가 브라우징할 수 있어야 한다.	작업공간의 워크스페이스 시스템을 통해 단일 산출물의 변경이력 및 산출물들의 의존 관계를 확인할 수 있음
4	다른 소프트웨어 공학 도구들과 상호 운용될 수 있어야 한다.	버전관리 시스템과 형상관리 시스템을 통합함으로써 상호 운용성을 만족시키고 있음
5	형상관리와 함께 변경 추적 기능을 제공해야 한다.	형상관리 시스템과의 통합으로 형상관리 프로세스 및 변경 추적 기능을 제공
6	추적성 관계에 대한 사용자의 질의 및 필터링 기능을 지원해야 한다.	단일 산출물에 대한 버전 그래프 보기와 산출물들간 의존 관계의 추적을 위한 버전 링크 식별 기능을 지원
7	추적성 프로세스와 관계에 대한 데이터를 분석하고 요약해줄 수 있어야 한다.	제시한 환경은 추적성 관계의 정보를 추출하고 시각화할 수는 있지만, 정보에 대한 분석 및 요약 기능은 제공하지 않음
8	지역 추적성과 전역 추적성을 통합 관리할 수 있어야 한다.	개인 작업공간에서의 변경뿐만 아니라 형상관리 시스템 내에서의 공식화된 변경까지 추적할 수 있도록 지원

〈표 4〉 추적성 관리 도구의 요구사항에 대한 형상관리 시스템들의 만족도 평가

(○ - 만족 / △ - 부분적으로 만족 / × - 만족하지 않음)

SCM tools / Req. No.	ClearCase	StarTeam	AccuRev	Perforce	PVCS	Harvest	제안한 시스템
1	○	○	○	○	○	×	○
2	△	△	△	△	△	△	△
3	△	△	△	△	△	△	○
4	○	△	△	△	○	○	○
5	○	×	×	×	○	○	○
6	×	×	×	×	×	×	△
7	×	×	×	×	×	×	×
8	○	×	△	△	×	×	○

추적성 관계식별 도구와의 통합에 대한 요구사항은 대부분의 형상관리 시스템들이 완벽하게 만족시키지 못하고 있다. 관련 연구에서 설명한 다양한 형태의 추적성 관계를 표현하기 위해서는 여러 가지 기술적 접근 방법이 요구되지만 대부분의 형상관리 시스템들은 버전 관리 수준에서 산출물들의 변경 이력에 대한 추적성만을 지원하고 있다. 본 논문에서 제시한 통합 환경 역시 개별 산출물에 대한 변경 이력과 한 형상항목에 포함된 산출물들 간의 의존 관계를 추적할 수 있도록 지원하지만 그 외의 추적성 관계를 식별하거나 표현하는 데는 한계를 지니고 있다. 단, 다른 형상관리 시스템들의 경우 한 형상항목에 포함된 산출물들의 의존 관계를 식별하고 표현하는 기능을 지원하지 않고 있기 때문에 본 논문의 통합 환경이 보다 개선된 추적성 관리 기능을 지원한다고 볼 수 있다. AccuRev나 Perforce, Harvest와 같은 시스템들이 여러 산출물들을 그룹화 하여 change list나 change package 단위로 관리하지만, 변경에 대한 이력 정보를 change list와 change package단위로 나타내기 때문에 그 안에 포함된 개별 산출물들의 진화 과정이나 의존 관계를 추적할 수는 없다.

추적성 관계의 브라우징에 대한 요구사항 역시 2번 요구사항과 마찬가지로 대부분 형상관리 시스템들이 제한된 기능만을 지원하고 있다. 앞서 설명한 바와 같이 개별 산출물이나 change package 단위의 진화 과정을 추적하는 기능은 지원하고 있으나, 그룹화 된 산출물들 간의 의존성 관계를 추적하고 브라우징 할 수 있도록 하는 기능은 지원되지 않고 있다. 본 논문의 통합 환경은 한 형상항목에 포함된 산출물들의 의존 관계를 리스트 형태로 조회할 수 있도록 함으로써 보다 향상된 브라우징 기능을 지원하고 있다.

다른 소프트웨어 공학 도구들과의 상호 운용에 대한 요구사항에 대해서는 비교 대상 시스템들과 제안한 통합 환경이 기본적으로 버전 관리 시스템 혹은 형상관리 시스템으로서의 역할을 수행하고 있기 때문에 대부분 만족하고 있다고 볼 수 있다. 단, StarTeam, AccuRev, Perforce와 같은 시스템들은 형상관리 프로세스를 지원하지 않고 버전 관리 수준에서 추적성을 지원하고 있다. 다섯 번째 요구사항인 형상관리 기능의 지원 여부 역시 이러한 맥락과 함께한다. 단순 버전 관리 기능은 변경에 대한 공식적인 승인 절차와 같은 변경 통제 프로세스를 지원하지 않기 때문에 위의 시스템들은 해당 요구사항을 만족시킨다고 보기 어렵다.

추적성 관계에 대한 사용자의 질의 및 필터링 기능에 대한 요구사항은 대부분의 형상관리 시스템들이 변경 이력 조회나 버전 트리 혹은 버전 그래프 보기와 같은 기능을 제공함으로써 일부 만족하고 있으나, 여러 산출물들의 의존 관계에 대한 질의 및 필터링 기능을 제공하지는 않고 있다. 본 논문의 통합 환경은 한 형상항목에 포함된 산출물들에 대해서 특정 수정버전을 이용해 그와 관련된 산출물들의 버전을 필터링하여 얻을 수 있도록 지원함으로써 해당 요구사항을 만족시키고 있다.

추적성 프로세스와 추적성 관계에 대한 데이터를 시스템이 분석하고 요약해줄 수 있어야 한다는 요구사항에 대해서는 모두가 만족시키지 못하고 있다. 여러 가지 유형의 추적성 관계를 식별하고 그 정보를 해석하여 사용자들에게 알기 쉽게 전달하는 능력은 현재의 형상관리 시스템들이 보완해야 할 부분이다.

마지막으로 전역 추적성과 지역 추적성에 대한 통합 관리 요구사항은 개인적인 변경 이력과 형상관리 상에서의 공식적인 변경 이력을 통합 관리할 수 있어야 함을 의미한다. 본 논문의 통합 환경과 ClearCase가 이러한 요구사항을 만족하고 있다. AccuRev와 Perforce는 통합 변경 관리를 지원하나 공식적인 형상관리 프로세스를 지원하지 않기 때문에 해당 요구사항을 부분적으로만 만족하고 있다. StarTeam은 지역 추적성만을, PVCS와 Harvest는 전역 추적성만을 지원하고 있으므로 추적성의 통합에 대한 요구사항은 만족시키지 못한다고 평가된다.

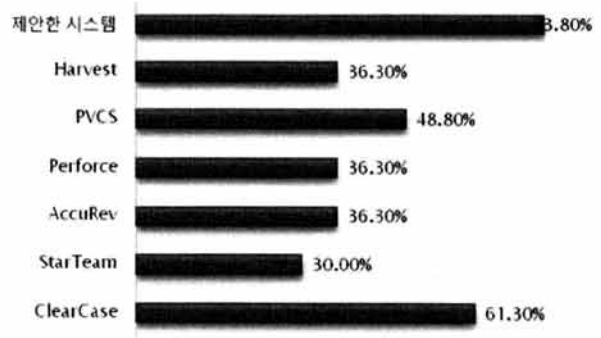
5.2 추적성 관리도구의 요구사항에 대한 정량적 평가

본 장에서는 추적성 관리도구의 요구사항에 대한 정성적 평가 결과를 토대로 각 형상관리 시스템들을 정량적으로 평가한다. <표 3>의 8가지 요구사항을 기준으로 한 정량적 평가 모델은 다음과 같다.

$$SCM_{tool} = \frac{10\sum_{i=1}^{n} w_i \times p_i}{n}\%$$

- SCM_{tool} : 추적성 관리도구 요구사항에 대한 형상관리 시스템의 전체 평가 지수(%)
- n : 요구사항 항목의 전체 개수
- i : 요구사항 번호
- w : 각 요구사항에 대한 가중치, $0 < w \leq 1$
  (본 연구의 평가에서는 모든 요구사항에 대해 가중치 1로 설정)
- p : 각 요구사항 항목에 대한 평가 점수, 각 요구사항에 대한 만족도에 따라 1~10까지의 범위를 가짐

위의 평가 모델을 토대로 비교 대상인 형상관리 시스템들과 본 논문의 통합 환경을 평가한 결과는 (그림 4)과 같다.



(그림 4) 추적성 관리도구의 8가지 요구사항에 대한 각 형상관리 시스템의 만족도 평가

평가 모델에서 각 요구사항 항목에 대한 평가 점수 p는 해당 요구사항을 만족하는 수준에 따라 결정하였으며, (그림 4의 평가 결과는 각 형상관리 시스템들이 <표 3>의 8가지 요구사항을 어느 정도 수준으로 만족하고 있는지 나타내고 있다.

6. 결론 및 향후 연구

소프트웨어 산출물들에 대한 추적성은 고품질의 소프트웨어를 생산하는 데 있어 매우 중요한 요소이다. 개발자들은 소프트웨어 산출물들이 어떻게 진화해 왔는지, 또는 서로 어떠한 의존 관계를 맺고 있는지 추적함으로써 변경을 효율적으로 관리할 수 있다. 과거의 많은 연구들에 의해서 다양한 형태의 추적성 관계가 다루어졌으며, 추적성 관계의 표현 및 추적성 관리의 자동화 방안에 대한 많은 기법들이 제시되어왔다.

추적성 관리의 중요성과 더불어 추적성 관리 활동과 형상관리 활동이 서로 연계되어 이루어져야 한다는 필요성이 제기되었다. 이것은 보다 개선된 변경 관리 환경을 구축하기 위한 조건이 된다. 추적성 관리와 형상관리가 연계되어 이루어지기 위해서는 개발자들의 작업공간과 형상관리 시스템의 통합이 필요하다. 개발자들은 자신의 작업공간에서 산출물에 대한 개인적인 변경 이력을 관리하며 이것은 지역 추적성에 해당된다. 반면 형상관리 시스템 상에서 형상항목의 변경을 공유하고 추적하는 것은 전역 추적성에 해당된다. 개인 작업공간과 형상관리 시스템을 통합하는 것은 추적성 관리와 형상관리를 통합한다는 의미와 함께 지역 추적성과 전역 추적성을 통합 관리한다는 의미도 포함한다.

본 논문에서는 개인 작업공간과 형상관리 시스템을 통합한 환경을 제시하였다. 개발자들은 통합 환경을 통해 산출물들에 대한 진화 관계와 형상항목에 대한 진화 관계를 연결시켜 관리할 수 있다. 이것은 앞서 말한 지역 추적성과 전역 추적성이 통합 관리될 수 있다는 것을 의미한다. 또한 한 형상항목에 여러 산출물들이 포함되는 경우 개발자들은 이들 산출물들의 의존 관계도 추적할 수도 있다. 산출물들의 의존 관계를 추적하는 능력은 과거의 변경을 정확하게 이해하는 데 도움을 주며 향후에 있을 변경 작업에 일관성을 더해줄 수 있다.

본 논문에서 제시한 통합 환경은 개발자들의 편의를 위해 시각화된 버전 그래프와 함께 자동화된 태깅 메커니즘, 효율적인 질의 검색 기능을 제공한다. 이들은 수동적인 추적성 관리에서 나타나는 어려움들을 상당 부분 해소시키기 위한 것으로 추적성 관리에 필요한 정보들을 손쉽고 정확하게 획득할 수 있도록 지원한다.

본 논문에서 다룬 진화 관계와 의존 관계는 산출물들과 형상항목의 변경 이력을 기반으로 관리된다. 특히 의존 관계의 관리는 한 형상항목에 포함된 산출물들을 그 대상으로 하며 이것은 수직 추적성과 관련이 있다고 볼 수 있다. 향후 연구에서는 서로 다른 형상항목들 간의 의존 관계까지 추적할 수 있도록 하는 방법에 대해 다루고자 한다. 이는 수평 추적성에 해당되는 것으로 수직 추적성과 함께 반드시 관리되어져야 한다.

참 고 문 헌

[1] http://www.iso.org

[2] Pohl K., "PRO-ART: Enabling Requirements Pre-Traceability", Proceedings of the 2nd IEEE International Conference on Requirements Engineering (ICRE 1996), 1996.

[3] Cleland-Huang J., Schmelzer D., "Dynamic Tracing Non-Functional Requirements through Design Pattern Invariants", Proceedings of the 2nd International Workshop on Traceability in Emerging Forms of Software Engineering(TEFSE 2003), Canada, Oct., 2003.

[4] Cleland-Huang J., Chang C.K., Sethi G., Javvaji K., Hu H., Xia J., "Automatic Speculative Queries through Event-based Requirement Traceability", Proceedings of the IEEE Joint International Requirements Engineering Conference, Germany, Sept., 2002.

[5] Egyed A., Gruenbacher P., "Automatic Requirements Traceability: Beyond the Record and Replay Paradigm", Proceedings of the 17th IEEE International Conference on Automated Software Engineering, Edinburgh, UK, Sept., 2002.

[6] Song X., Hasling B., Mangla G., Sherman B., "Lessons Learned from Building a Web-Based Requirements Tracing System", Proceedings of 3rd International Conference on Requirements Engineering, pp.41-50, 1998.

[7] Stout G.A., "Requirements Traceability and the Effect on the System Development Lifecycle(SDLC)", http://www.reveregroup.com/articles/137_005-RevereThoughtLeadership.pdf

[8] Ramesh B., Jarke M., "Toward Reference Models for Requirements Traceability", IEEE Transactions on Software Engineering, 27(1), Jan., 2001.

[9] Gotel O., Finkelstein A., "An Analysis of the Requirements Traceability Problem", Proceedings of the 1st International Conference in Requirements Engineering, pp.94-101, 1994.

[10] Spanoudakis G., Zisman A., "Software Traceability: A Roadmap", Handbook of Software Engineering and Knowledge Engineering, Vol.3, pp.395-428, Aug., 2005.

[11] Lindvall M., Sandahl K., "Practical Implications of Traceability", Software Practice and Experience, Vol.26, No.10, pp.1161-1180, 1996.

[12] Maletic J.I., Munson E.V., Marcus A., Nguyen T.N., "Using a Hypertext Model for Traceability Link Conformance Analysis", Proceedings of the 2nd International Workshop on

Traceability for Emerging Forms of Software Engineering (TEFSE '03), Canada, Oct., 2003.

[13] Pfleeger S.L., Bohner S.A., "A Framework for Software Maintenance Metrics", Proceedings of 6th Conference on Software Maintenance, pp.320-327, 1990.

[14] Antoniol G., Canfora G., Casazza G., De Lucia A., "Maintaining Traceability Links during Object-Oriented Software Evolution", Software Practice and Experience, 31(4), pp.331-355, 2001.

[15] Buckner J., Buchta J., Petrenko M., Rajlich V., "JRipples: A tool for program comprehension during incremental change", Proceedings of 13th International Workshop on Program Comprehension", pp.149-152, 2005.

[16] Gallagher K., Lyle J., "Using Program Slicing in Software Maintenance", IEEE TSE, 17(8), pp.751-761, 1991.

[17] Ren X., Shah F., Tip F., Ryder B.G., Chesley O., "Chianti: a tool for change impact analysis of java programs", ACM SIGPLAN Notices, 39(10), pp.432-448, 2004.

[18] Antoniol G., Canfora G., Casazza G., De Lucia A., Merlo E., "Recovering Traceability Links between Code and Documentation", IEEE Transactions on Software Engineering, 2003.

[19] Hayes J.H., Dekhtyar A., Osborne J., "Improving Requirements Tracing via Information Retrieval", Proceedings of the 11th IEEE International Requirements Engineering Conference, Monterey Bay, 2003.

[20] Cleland-Huang J., Settimi R., Duan C., Zou X., "Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability", 13th IEEE International Conference on Requirements Engineering, Paris, pp.135-144, 29 Aug.-2 Sept., 2005.

[21] Yeong-Jae, Yoo, "Development of a Traceability Analysis Method based on Case Grammar for NPP Requirement Documents written in Korean Language", M.S. Thesis, Department of Nuclear and Quantum Engineering, KAIST, 2003.

[22] Canfora G., Cerulo L., "Impact Analysis by Mining Software and Change Request Repositories", Proceedings of 11th International Symposium on Software Metrics, pp.20-29, 2005.

[23] Gall H., Hajek K., Jazayeri M., "Detection of Logical Coupling based on Product Release History", Proceedings of 14th ICSM, pp.190-198, 1998.

[24] Gall H., Jazayeri M., Krajewski J., "CVS Release History Data for Detecting Logical Coupling", Proceedings of 6th International Workshop on Principles of Software Evolution, pp.13-23, 2003.

[25] Ying A.T., Murphy G.C., Ng R., Chu-Carroll M.C., "Predicting Source Code Change by Mining Change History", IEEE TSE, 31(6), pp.429-445, 2005.

[26] Zimmermann T., Weisserber P., Diehl S., Zeller A., "Mining Version Histories to Guide Software Changes", IEEE TSE, 31(6), pp.429-445, 2005.

[27] Kagdi H., Maletic J.I., Sharif B., "Mining Software Repositories for Traceability Links", 15th IEEE International Conference on Program Comprehension (ICPC'07), pp.145-154, 2007.

[28] Gotel O., Finkelstein A., "Contribution Structures", Proceedings of 2nd International Symposium on Requirements Engineering, pp.100-107, 1995.

[29] Sherba S.A., Anderson K.M., Faisal M., "A Framework for Mapping Traceability Relationships", Proceedings of the 2nd International Workshop on Traceability for Emerging Forms of Software Engineering (TEFSE 2003), Montreal, Canada, Sept., 2003.

[30] Cleland-Huang J., Chang C., Wise J., "Supporting Event Based Traceability through High-Level Recognition of Change Events", Proceedings of IEEE COMPSAC Conference, Oxford, England, Aug., 2002.

[31] 김대엽, 윤청, "소프트웨어 형상관리와 작업공간의 통합을 통한 산출물의 추적성 향상 기법", 정보처리학회 논문지, 제16-D권, 제6호, 2009.

[32] Kim D., Youn C., "Traceability Enhancement Technique through the Integration of Software Configuration Management and Individual Working Environment", Proceedings of IEEE International Conference on Secure Software Integration and Reliability Improvement, pp.163-172, Jun., 2010.

[33] Berczuk S.P., Appleton B., "Software Configuration Management Patterns, Effective Teamwork, Practical Integration", Addison Wesley, 2002.

[34] Marcus A., Xie X., Poshyvanyk D., "When and How to Visualize Traceability Links?", Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering, pp.56-61, 2005.

김 대 엽

e-mail : kdymn2@cnu.ac.kr

2005년 충남대학교 정보통신공학부(학사)

2005년~현 재 충남대학교 컴퓨터공학과 석·박사 통합과정

관심분야 : 소프트웨어 형상관리, 추적성관리, 컴포넌트 기반 개발 방법론 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