http://dx.doi.org/10.3745/KTSDE.2013.2.7.461

Traceability Management Technique for Software Artifacts which Comprise Software Release

Dae Yeob Kim† · Cheong Youn††

ABSTRACT

The capacity for tracing relationships among various artifacts which are created at each phase of software system development is essential for software quality management. Software release refers to delivering a set of newly created or changed artifacts to customers. The relationships among artifacts which comprise software release must be traced so that the work for customer's requirement of change and functional enhancement is effectively established. And release management can be effectively realized through the integration of configuration management and change management. This paper proposes the technique for supporting change management of artifacts and for tracing relationships of artifacts which comprise software release through the integrated environment of personal workspace and configuration management system. In the proposed environment, the visualized version graph and automated tagging function are used for tracing relationships of artifacts.

Keywords : Release, Traceability, Baselined Document, Configuration Management, Version Control

소프트웨어 릴리스를 구성하는 산출물들의 추적성 관리 기법

김 대 엽† · 윤 청††

요 약

소프트웨어 시스템 각 개발단계에서 만들어지는 다양한 산출물들의 연관 관계를 추적하는 능력은 소프트웨어의 품질 관리에 필수적인 요소이다. 소프트웨어 릴리스는 신규 또는 변경된 산출물들의 집합(실행 프로그램 포함)을 고객에게 인도하는 것이다. 소프트웨어 릴리스를 구성하는 산출물들의 연관 관계를 정확하게 추적할 수 있어야 고객의 변경 요구나 기능 개선을 위한 효율적인 작업을 수행할 수 있으며, 릴리스 관리는 형상 관리와 변경 관리를 통합할 때 효율적으로 이루어질 수 있다. 본 논문은 개인 작업 공간과 형상 관리 시스템을 통합한 환경을 통해 산출물들의 변경 관리를 지원하고 릴리스를 구성하는 산출물들의 연관 관계를 효율적으로 추적할 수 있는 방법을 제시한다. 제시된 환경에서 산출물의 연관 관계를 추적하기 위해 시각화된 버전 그래프와 자동화된 태깅(tagging) 기능을 사용한다.

키워드 : 릴리스, 추적성, 기준선 문서, 형상 관리, 버전 관리

1. 서 론

소프트웨어 시스템은 각 개발 단계에서 만들어지는 산출물들의 조합으로 이루어지며, 이런 산출물들 사이의 연관 관계를 추적할 수 있는 능력, 즉 추적성(traceability)은 개발 과정뿐만 아니라 향후 유지보수 과정에 있어서도 매우 중요하다. 예를 들어 소프트웨어를 고객에게 인도한 이후 새로운 변경 요구사항이 발생한다면, 이전 릴리스를 구성하는 산출물들의 연관 관계를 정확하게 추적해야 효율적인 작업을 수행할 수 있다.

시스템은 하드웨어 시스템과 소프트웨어 시스템을 포괄하는 개념이며, 본 논문은 소프트웨어 시스템을 구성하는 요소들에 대한 추적성 관리 방법을 제시한다. 자동차나 가전제품과 같은 시스템은 하드웨어와 소프트웨어가 결합된 형태로 존재하는데, 전체 시스템 관점에서 보면 소프트웨어 시스템도 하나의 형상 항목이 될 수 있다. 본 논문은 전체 시스템과 함께 릴리스 되는 소프트웨어 자체를 하나의 형상 항목으로 간주하지 않고, 소프트웨어 시스템을 구성하는 요소들에 대해서만 고려한다.

소프트웨어 산출물들의 연관 관계 추적에 대한 연구는 다양한 관점에서 진행되어왔다. 추적성과 관련된 기존 연구들을 종합해보면 산출물들의 연관 관계에 대한 추적성은 크게

※ 이 연구는 2012년도 충남대학교 학술연구비에 의해 지원되었음.
† 준 회 원 : 충남대학교 컴퓨터공학과 박사과정
†† 정 회 원 : 충남대학교 컴퓨터공학과 교수
논문접수 : 2013년 2월 15일
수 정 일 : 1차 2013년 4월 2일
심사완료 : 2013년 4월 3일
* Corresponding Author : Cheong Youn(cyoun@cnu.ac.kr)

요구사항 기반의 추적성과 변경 이력 기반의 추적성으로 구분할 수 있다.

요구사항 기반의 추적성은 사용자 요구사항으로부터 그것과 관련된 다른 산출물들의 연관 관계를 추적하는 능력으로 정의할 수 있다. 요구사항 기반의 추적성은 요구사항 분석 단계 이전에 대한 추적성과 요구사항 분석 단계 이후에 대한 추적성으로 구분할 수 있다.

요구사항 분석 이전에 대한 추적성은 이해 당사자(stakeholder)들의 요구와 식별된 요구사항 사이의 관계를 나타냄으로써 요구사항 검증(validation)과 같은 활동을 지원한다. 요구사항 분석 이후의 추적성은 요구사항과 그것을 기반으로 생성되는 다른 산출물들(논리 모델, 설계 모델, 소스 코드, 시험 사례 등) 간의 관계를 나타냄으로써 확인(verification)이나 시험과 같은 활동을 지원한다[1, 2-8]. 요구사항 기반의 추적성은 산출물들의 내용을 토대로 의미적인 연관성을 규명하는데 초점을 맞춘다. 이러한 형태의 추적성은 산출물들의 변경 이력을 고려하지 않는다는 것이 한계로 지적된다[9, 10].

변경 이력 기반의 추적성은 과거 변경 이력을 기반으로 각 산출물의 진화 과정 및 여러 산출물들 사이의 연관 관계를 추적하는 것이다. 어떠한 산출물의 변경이 다른 산출물들의 변경을 야기한 경우, 변경 이력 정보를 통해서 함께 변경된 산출물들 사이의 연관 관계를 밝혀낸다. 산출물들의 연관 관계는 일반적으로 버전들 사이의 링크를 구성함으로써 나타낸다[9-14]. 변경 이력 정보를 기반으로 산출물들 사이의 연관 관계를 식별하면, 변경에 대한 영향 분석이나 기존 소프트웨어에 대한 유지보수 등의 작업을 보다 효율적으로 수행할 수 있다. 단, 기존의 연구들은 주로 버전 관리 도구에서 제공하는 이력 정보만을 활용하여 산출물들의 연관 관계를 추적한다는 점에서 한계를 갖고 있다. 변경 이력을 이용하여 추적성을 관리하고자 할 때, 단순 버전 관리 도구와 형상 관리 도구를 통합하여 함께 활용하면 더욱 효과적인 작업이 가능할 것이다.

본 논문은 위의 두 가지 추적성 관리 방법 가운데 변경 이력에 기반한 추적성 관리 방법을 제시한다. 개인의 작업 공간과 형상 관리 시스템을 통합함으로써 기존의 연구들(변경 이력 기반의 추적성 관리에 대한 연구들)이 제공하지 못했던 확장된 추적 정보들을 제공한다. 본 연구의 통합 환경에서는 다음과 같이 세 가지 유형의 추적성을 지원한다.

1) 단일 산출물의 진화 과정에 대한 추적성
2) 단일 기준선 문서[1](baselined document)에 포함된 여러 산출물들의 연관 관계에 대한 추적성
3) 소프트웨어 시스템의 특정 릴리스를 구성하는 기준선 문서 및 전체 산출물들의 연관 관계 추적

첫 번째 추적성은 한 산출물의 진화 과정을 추적하는 능력이다. 산출물은 개인 작업 공간에서 변경되고, 그 결과는 형상 관리 시스템에서 공식화된다. 개인 작업 공간에서 발생한 변경의 이력과 형상 관리 시스템에 등록된 공식화된 변경을 통합 관리할 수 있도록 지원한다[15].

두 번째 추적성은 여러 산출물들이 하나의 형상 항목에 포함되는 경우, 이들 산출물들의 버전 링크를 식별함으로써 산출물들 사이의 연관 관계를 추적하는 능력을 의미한다[16].

위의 두 가지 추적성에 대한 지원 기법은 이미 선행 연구를 통해 소개한 바 있다. 본 논문에서 중점적으로 다루고자 하는 내용은 세 번째 추적성에 대한 지원 기법으로 특정 릴리스를 구성하는 기준선 문서들 및 전체 산출물들 사이의 연관 관계를 추적하는 방법이다.

본 연구에서 제시하는 통합 환경은 위의 세 가지 추적성을 지원하기 위해 시각화된 버전 그래프와 자동화된 태깅 기능 등을 제공한다. 버전 그래프는 산출물의 버전들을 트리 형태로 표현한 것으로서, 산출물의 진화 과정을 쉽게 파악할 수 있도록 하는 대표적인 버전 모델이다. 버전 그래프는 버전들의 집합을 구조화된 형태로 나타내며, 과거 버전을 획득하거나 새로운 버전을 생성하는데 있어 효과적인 방법을 제공한다[17].

태깅 기능이란 산출물의 특정 버전에 의미 있는 정보를 태그(tag)로 연결하는 기능이며, 본 연구의 통합 환경은 이 기능을 자동화하여 효율적인 추적성 관리가 가능하도록 한다. 태그는 단일 산출물의 진화 과정 중에서 개인에 의한 임의의 변경과 형상 관리 시스템에 공식화된 변경을 구별할 수 있게 해주며, 산출물들의 연관 관계를 추적할 수 있는 단서를 제공해주기도 한다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 소프트웨어 릴리스 관리의 개념을 소개하고 소프트웨어 추적성과 관련된 몇몇 연구들을 소개한다. 3장에서는 소프트웨어 릴리스와 관련된 추적성 관리 개념에 대해서 설명하고, 4장에서는 추적성 관리 대상의 개념과 본 논문에서 제시하는 추적성 관리 기법에 대해서 소개한다. 5장에서는 4장에서 소개한 추적성 관리 기법이 어떻게 구체화되었는지 그 구현 결과를 소개한다. 6장에서는 다른 상용 형상 관리 시스템들과 본 논문에서 제시한 통합 환경을 정성적, 정량적으로 비교 평가한다. 마지막으로 7장에서는 본 논문의 결론과 향후 연구에 대해 기술한다.

2. 관련 연구

릴리스란 공식적인 변경 승인 과정과 테스트를 거친 후 소프트웨어, 하드웨어 및 관련된 문서 등의 변경사항을 사용자에게 전달하는 것을 의미한다. ITIL(IT Infrastructure Library)에서는 릴리스를 IT 서비스에 대한 인가된 변경의 집합이며, 변경 요청(RFC, Request for Change)에 의해 유발되는 것으로 정의하고 있다[18]. ISO/IEC 20000-1에서는 릴리스는 시험되고 운용 환경에 도입된 신규 또는 변경된 형상 항목들의 집합이라 정의한다[19].

1) 형상 항목으로 식별된 산출물(들)이 형상 관리 시스템을 통해 공식화된 것으로, 소프트웨어 릴리스의 구성요소이다.

릴리스 관리는 릴리스와 관련된 모든 변경 사항을 점검하고 확인하는 과정이다. 릴리스 관리의 주목적은 공식적인 절차와 확인을 거쳐 IT 운영환경과 서비스를 안전하게 제공하기 위한 것이며[20], 릴리스에 포함된 하나 또는 그 이상의 변경을 운용 환경에 제공하고 배포하며 추적하고자 하는 것이다. ISO/IEC 20000-1 표준에서는 릴리스 관리가 변경 관리, 형상 관리 활동과 연계되어 이루어져야 한다고 강조한다[19].

변경 관리는 모든 변경들이 통제된 방식으로 심사, 승인, 구현, 검토됨을 보장하기 위한 목적으로 수행되며, 형상 관리는 형상 항목을 정의 및 통제하고 정확한 형상 정보를 유지하기 위한 목적으로 수행된다. 변경 관리와 형상 관리 역시 릴리스 관리와 같이 하나의 통합된 접근 방법을 적용해야 한다. 특히 형상 관리는 형상 항목의 버전을 추적할 수 있는 메커니즘을 제공해야 한다. 다시 말해 형상 항목에 대한 변경을 추적 가능하도록 관련 정보를 기록해야 한다.

ITIL은 릴리스 관리 프로세스, 변경 관리 프로세스, 형상 관리 프로세스가 어떻게 통합되어야 하는지 관련 프로세스를 제시하고 있다. Fig. 1은 ITIL에서 제시하는 서비스 관리 프로세스를 개략적으로 나타내고 있다. ITIL에서 제시하고 있는 서비스 관리 프로세스는 여러 가지 하위 프로세스들을 포함하고 있으며, 그 중 변경 관리 프로세스, 릴리스 관리 프로세스, 형상 관리 프로세스는 하나의 프로세스처럼 유기적인 관계를 맺고 있어야 한다. 통합된 프로세스에서는 변경의 식별, 평가 및 승인, 변경의 구현, 시험 및 릴리스, 상태 보고, 감사와 같은 활동이 유기적으로 이루어져야 하며, 시스템을 구성하는 모든 산출물들이 추적될 수 있어야 한다.

Mohan은 일반적으로 소프트웨어에 대한 형상 관리와 추적성 관리가 독립적으로 이루어지는 점을 문제로 지적하고, 위의 두 관리 프로세스를 통합하기 위한 프로세스 프레임워크를 제시하고 있다. 저자가 제시한 프레임워크에는 형상 관리 프로세스와 추적성 관리 프로세스가 통합되기 위해 어떠한 상호작용이 이루어져야 하는지 보여주고 있다. 그는 형상 관리와 추적성 관리가 통합되어 이루어지기 위해서는 계획, 통합 환경 관리, 변경의 요청 및 배포 관리, 기준선 및 릴리스 관리, 형상 상태에 대한 감시 및 보고 등에 대한 업무가 유기적으로 연계되어야 한다고 주장한다[21].

그의 주장에 따르면, 두 업무 프로세스의 계획 단계에서는 어떠한 정보가 어느 정도의 상세화 수준에서 관리되어야 하는지 결정되어야 하며, 형상 관리와 추적성 관리의 업무 프로세스에서 산출물들의 특정 버전들에 대한 연관 관계 및 형상 항목들 간의 연관 관계 정보가 잘 정제되어 관리되어야 한다. 또한 변경의 요청 및 배포 관리에 있어서 버전 정보에 기반한 산출물들의 연관 관계를 적절히 식별할 수 있어야 한다. 기준선 및 릴리스 관리에 있어서는 시스템을 구성하는 모든 산출물들이 완전성과 일관성을 유지해야 하며, 이를 위해서는 모든 산출물의 연관 관계가 정확하게 수립되어야 한다. 마지막으로 형상 상태의 감시 및 보고에서는 형상 항목들의 생성 배경뿐만 아니라 산출물들의 버전 정보를 포함한 연관 관계의 기록이 반드시 이루어져야 한다.

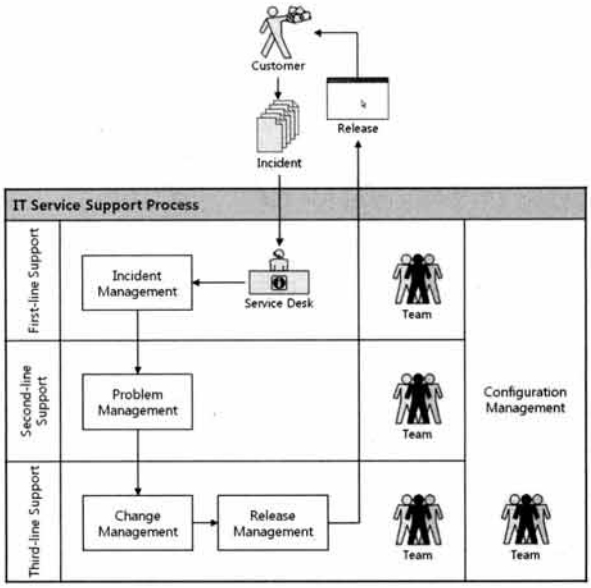

Fig. 1. Service Management Process of ITIL

Mohan이 주장하는 형상 관리와 추적성 관리의 통합은 ITIL이나 ISO/IEC 20000-1 표준에서 강조하는 통합 관리 프로세스의 실현과 밀접한 관련을 맺고 있다. 다만 현실적으로 적용 가능한 관리 기법이나 도구들에 대한 소개가 이루어지지 않아 추가적인 연구가 필요하다.

이 외에도 소프트웨어 산출물들의 추적성을 관리하기 위한 다양한 연구가 진행되어 왔다. 요구사항을 기반으로 다양한 종류의 산출물들에 대해 연관 관계를 식별하는 기법들이 연구되었으며[22-25], 변경 이력 정보를 기반으로 산출물들의 연관 관계를 식별하는 연구들도 진행되었다[26-31]. 기술적인 관점에서 산출물들의 연관 관계를 식별하고 추적하는 방법도 중요하지만, 관리적인 관점에서 산출물들의 추적성 관리가 어떻게 수행되어야 하는지 연구될 필요가 있다. 즉, 변경 요청으로부터 새로운 소프트웨어 시스템을 구축하기 위해 릴리스 관리와 형상 관리, 변경 관리가 어떻게 통합되어야 하며, 통합된 프로세스 안에서 산출물들의 연관 관계를 어떻게 나타내야 할지에 대한 연구가 수반되어야 할 것이다.

3. 소프트웨어 릴리스에 대한 추적성 관리의 개념

릴리스 관리, 변경 관리 그리고 형상 관리의 통합은 Fig. 2와 같은 프로세스를 포함한다[32]. 통합 프로세스는 변경의 식별(Identification)에서부터 시작한다. 식별된 변경에 대한 평가와 승인(Assessment & Approval), 변경의 구현(Implementation), 시험(Test) 및 릴리스(Release)가 이루어지면, 시스템에 대한 상태 보고(Status Accounting)가 이루어진다. 이 때 산출물들의 변경과 관련된 모든 정보들이 데이터베이스에 기록된다. 기록된 정보는 향후 릴리스와 관련된 내용을 추적하고 검증하는데 사용된다.

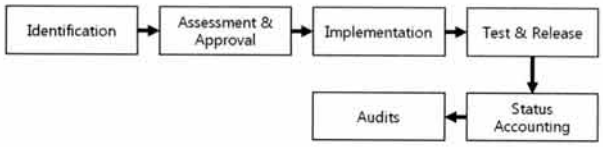

Fig. 2. Integrated Process of Configuration Management, Change Management, and Release Management

본 논문의 핵심인 릴리스에 대한 추적은 릴리스를 구성하는 기준선 문서들 및 전체 산출물들의 연관 관계를 추적하는 것을 의미한다. 각 기준선 문서는 형상 관리 시스템에 체크인[2](check-in)될 때마다 새로운 버전을 갖게 된다. 기준선 문서가 갖는 버전을 본 연구의 통합 환경에서는 수정 버전(revision)이라 부른다. 수정 버전은 기준선 문서가 형상 관리 시스템을 통해 공식적으로 승인되었을 때 생성되는 버전 정보로서 일반적인 버전 관리 도구에서 산출물에 부여하는 버전과는 다른 개념이다.

릴리스를 구성하는 기준선 문서들의 연관 관계를 나타내기 위해 본 연구는 각 기준선 문서들의 수정 버전을 링크시키는 방법을 사용한다. 릴리스를 구성하는 기준선 문서들의 수정 버전을 링크시키기 위해서 각 기준선 문서의 수정 버전에 시스템의 릴리스 번호[3]를 연결시킨다. 동일한 릴리스 번호와 연결된 수정 버전을 식별하게 되면 여러 기준선 문서들의 연관 관계를 추적할 수 있다.

본 연구의 제시한 환경에서 기준선 문서의 수정 버전과 시스템의 릴리스 번호를 연결시키는 일은 Fig. 2의 통합 프로세스 가운데 상태 보고 프로세스에서 이루어진다. 즉, 기준선 문서의 변경에 대한 시험이 완료되고 시스템이 릴리스되었을 때, 해당 릴리스를 구성하는 기준선 문서들에 릴리스 번호를 기록한다. 모든 기준선 문서에 대해서 릴리스 번호를 기록하면, 기준선 문서들의 수정 버전은 릴리스 번호에 의해 연결되고, 그 결과를 통해 기준선 문서들의 연관 관계를 추적할 수 있게 된다.

전체 산출물들의 연관 관계를 나타낼 때도 역시 릴리스 번호를 사용할 수 있다. 산출물들의 변경 이력을 나타내는 버전 정보와 시스템의 릴리스 번호를 연결하면 산출물들의 연관 관계를 나타낼 수 있다. 더 자세한 내용은 4장과 5장에서 설명하도록 한다.

4. 추적성 관리 대상의 구성과 관련 기법의 소개

기준선 문서 및 전체 산출물들의 연관 관계가 어떻게 관리될 수 있는지 보다 정확하게 이해하기 위해서는 먼저 기준선 문서와 산출물의 실질적인 형태와 그 구조를 이해해야 한다.

일반적으로 기준선 문서로 식별되는 것에는 요구사항 명세서나, 설계 명세서, 소스 코드, 실행 파일 등이 있을 수 있는데, 이러한 산출물들은 물리적인 파일 형태로 존재한다. 그 중 소스 코드는 일반적으로 여러 개의 파일들이 모여 하나의 단위 컴포넌트를 구성한다. 산출물들이 형상 관리 시스템 상에서 관리되기 위해서는 형상 항목과 연결되어야 한다. 형상 항목은 공식적인 승인 과정을 거쳐 비로소 하나의 기준선 문서가 되며, 기준선 문서는 특정 릴리스를 구성하는 요소가 된다. 지금까지 설명한 요소들을 계층적인 구조로 도식화하면 Fig. 3과 같다.

서론에서 설명한 바와 같이 선행 연구에서는 Fig. 3의 각각 맨 아래와 중간 계층에 존재하는 산출물과 기준선 문서에 대한 추적성 관리 기법을 소개하였다[15, 16].

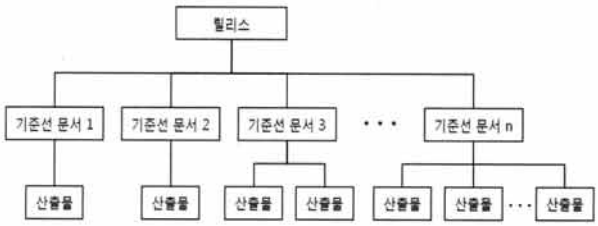

Fig. 3. Hierarchy of Release Component

본 논문에서 제시하고자 하는 내용은 릴리스를 구성하는 기준선 문서들의 연관 관계와 나아가 전체 산출물들의 연관 관계를 추적하는 방법이다. 소프트웨어 시스템이 릴리스 될 때 부여되는 릴리스 번호를 이용하여 그것과 관계된 기준선 문서들의 연관 관계를 나타낼 수 있으며, 나아가 전체 산출물들의 연관 관계까지도 나타낼 수도 있다.

릴리스를 구성하는 기준선 문서들의 연관 관계는 각 기준선 문서에 시스템의 릴리스 번호를 기록함으로써 나타낼 수 있다. 보다 정확히 말하면, 각 기준선 문서는 형상 관리 시스템에 등록될 때마다 새로운 수정 버전을 갖게 되며, 릴리스 번호를 기준선 문서에 기록한다는 것은 릴리스 번호가 각 기준선 문서의 어떤 수정 버전과 관계를 맺는지 나타낸다는 것을 의미한다.

릴리스 번호를 이용해 전체 산출물들(Fig. 3의 맨 아래 계층에 위치)의 연관 관계를 나타내기 위해서는 서론에서 언급한 바 있는 시각화된 버전 그래프와 자동화된 태깅 기법을 사용한다. 산출물들은 실질적인 변경의 대상으로서 개발자들의 작업 공간에서 변경이 이루어지며, 저마다 다른 흐름의 변경 이력을 갖고 있다. 각 산출물들의 변경 이력을 기록하기 위해 버전 번호가 사용되며, 본 연구에서는 각 산출물들의 변경 이력을 보다 명확하게 이해할 수 있도록 시각화된 버전 그래프를 제공한다.

자동화된 태깅 기법은 각 산출물의 버전 번호 가운데 공식화된 기준선 문서와 연결된 버전 번호를 추적할 수 있도록 하기 위해 사용되었다[15, 16]. 기준선 문서에 포함된 산출물들의 변경이 완료되고 그 결과를 다시 형상 관리 시스템에 반영하고자 할 때, 기준선 문서를 체크인하게 되는데, 이 때 생성된 기준선 문서의 수정 버전이 각 산출물들의 버전 번호(최신 버전)에 태그로 연결된다. 산출물의 버전 번호

2) 형상 관리 시스템으로부터 다운로드 받은 기준선 문서를 개인 작업 공간에서 작업한 뒤, 다시 형상 관리 시스템에 공식적으로 반영하는 과정.
3) 시스템이 릴리스 될 때 부여되는 일종의 릴리스 버전 정보.

와 기준선 문서의 수정 버전의 연결은 버전 그래프로 쉽게 확인할 수 있다. 기준선 문서에 여러 개의 산출물들이 포함된 경우, 태그를 통해 기준선 문서의 수정 버전과 연결된 각 산출물들의 버전 번호를 쉽게 획득할 수 있다.

통합 환경의 자동화된 태깅 기법을 사용하면 소프트웨어 시스템의 특정 릴리스 번호와 관련된 전체 산출물들의 연관 관계 역시 쉽게 나타낼 수 있다. 기준선 문서에 릴리스 번호를 기록할 때, 기준선 문서에 포함된 산출물들의 버전에 릴리스 번호를 태그로 연결하면 전체 산출물들의 연관 관계를 추적할 수 있게 된다.

5. 추적성 관리 기법의 구현 결과

이번 장에서는 4장에서 설명한 추적성 관리의 개념을 어떻게 구체화하였는지 소개하고자 한다.

기준선 문서의 수정 버전과 시스템의 릴리스 번호를 연결하는 것은 상태 보고 프로세스에서 이루어진다고 앞서 설명한 바 있다. 기준선 문서들의 연관 관계를 추적하려면 동일 릴리스 번호와 각 기준선 문서의 어떤 수정 버전이 연결되었는지 확인하면 된다.

전체 산출물들의 연관 관계를 나타내기 위해서는 자동화된 태깅 기법이 사용된다. 형상 관리의 상태 보고 프로세스에서 형상 관리 담당자는 릴리스를 구성하는 모든 기준선 문서들에 대해서 릴리스 번호를 기록하게 되며, 이 때 기록된 릴리스 번호가 기준선 문서를 구성하는 산출물들의 버전에 태그로 연결된다. 연결된 태그 정보는 각 산출물의 버전 그래프에서 확인할 수 있으며, 특정 릴리스와 관련된 산출물들의 버전은 릴리스 번호를 키워드로 하여 쉽게 획득할 수 있다.

Fig. 4는 특정 산출물의 버전 그래프를 보이고 있다. 버전 그래프는 산출물의 이름, 버전 번호, 태그, 분기로 구성된다. 버전 번호는 해당 산출물의 변경 이력을 나타내는 정보로서 버전 그래프에서는 버전 번호를 통해서 산출물의 전체 변경 이력을 확인할 수 있다. 태그는 산출물의 각 버전에 어떠한 설명을 기록하고자 할 때 사용된다. Conradi의 논문에 의하면 특정 버전과 관련된 작업(task)이나 의미 있는 기호(일반적으로 형상 항목의 수정버전이나 변형 정보)가 태그가 될 수 있다[17].

태그는 산출물의 변경 이력에서 이정표(milestone)와 같은 의미를 지닌다. 태그는 사용자가 수동으로 생성할 수도 있으며(Fig. 4의 'payment_way_add' 태그의 경우) 시스템에 의해 자동으로 생성시킬 수도 있다(Fig. 4의 'REV_00', 'REV_01' 태그의 경우). 시스템에 의해 자동으로 생성되는 경우는, 산출물과 연결된 기준선 문서를 형상 관리 시스템에 공식적으로 반영(check-in)하거나(기준선 문서의 수정버전이 태그로 붙음), 형상 관리 담당자가 기준선 문서에 릴리스 번호를 기록하고자 할 때(시스템의 릴리스 번호가 태그로 붙음)이다.

분기(brach)는 기존의 버전에 대해서 독립적인 흐름으로 새로운 변형(variants)을 개발하는 것이다. 예를 들어, 특정 버전에 대해 버그를 수정한다거나 다른 응용분야에 적용하기 위해 새로운 흐름의 개발을 진행하고자 할 때 분기를 생성할 수 있다. 분기를 이용하면 병렬적인 개발이 가능하기 때문에 기존의 개발 흐름에 영향을 주지 않고 효율적인 작업을 수행할 수 있다. Fig. 4의 경우, 'SEShop'이라는 분기는 기존의 요구사항 명세서를 변형하여 새로운 요구사항 명세서를 작성하기 위해 생성한 것이다.

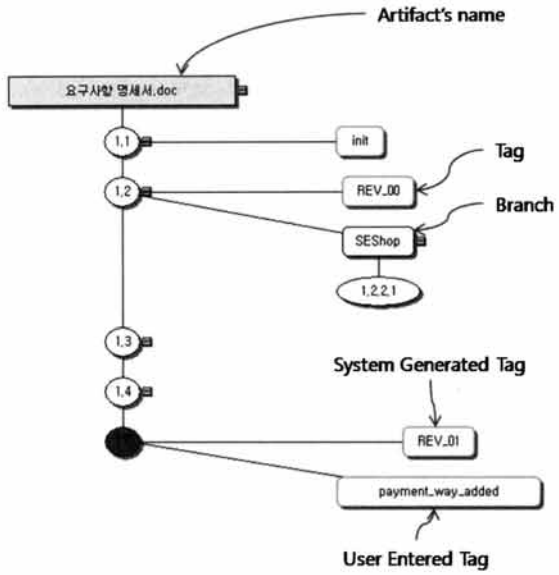

Fig. 4. Version Graph of Artifact

Fig. 5는 기준선 문서에 포함된 여러 산출물들이 기준선 문서의 수정 버전에 의해서 서로 연관 관계를 맺게 되는 모습을 보여준다. 기준선 문서에 포함된 산출물들은 각각 서로 다른 흐름의 변경 이력을 갖지만, 기준선 문서가 형상 관리 시스템에 반영될 때, 기준선 문서의 수정버전이 자동으로 산출물들의 최신 버전에 태그로 연결됨으로써 산출물들의 연관 관계를 나타낼 수 있게 된다.

Fig. 5에서 4개의 서로 다른 산출물들은 동일한 기준선 문서에 포함된 것이며, 산출물들의 연관관계가 기준선 문서의 수정버전에 의해서 식별될 수 있음을 버전 그래프를 통해 보여주고 있다. 본 연구에서 개발한 관리 도구는 한 기준선 문서에 여러 산출물들이 포함된 경우, 기준선 문서의 특정 수정버전과 연결된 산출물들의 버전을 손쉽게 획득할 수 있도록 지원한다. 이 기능은 기준선 문서의 과거 변경 내용을 추적하고 분석하는데 있어 효율성을 제공한다. 결과적으로 향후에 발생할 변경에 대한 영향을 분석하는데 있어서도 이러한 기능이 도움이 될 수 있다. Table 1은 Fig. 5의 경우에서 산출물들의 버전이 기준선 문서의 수정 버전에 따라 어떻게 연결되었는지 간단하게 정리한 것이다. 기준선 문서의 특정 수정 버전에 의해서 산출물들의 버전 링크를 식별할 수 있으며, 결과적으로 한 기준선 문서에 포함된 여러 산출물들의 연관 관계를 추적할 수 있다.

Table 1. Relationships among Artifacts through Revision of Baselined Document

Name of Artifacts \ Revision	REV_00	REV_01
PaymentList.jsp	1.2	1.3
OrderPaymentSave.jsp	1.1	1.3
OrderPayment.jsp	1.2	1.3
PaymentConfirm.jsp	1.1	1.2

마지막으로 Fig. 6은 시스템의 릴리스 번호에 의해 산출물들이 연결된 모습을 나타내고 있다. 그림을 통해서 시스템의 릴리스 번호에 따라 전체 산출물들이 관계를 맺을 수 있다는 것을 알 수 있다. 시스템이 릴리스 된 이후에 형상 관리 담당자는 릴리스를 구성하고 있는 기준선 문서들에 대해서 상태 보고를 수행한다. 상태 보고 시에 기준선 문서가 특정 릴리스에 포함되었다는 것을 나타내기 위해 기준선 문서에 릴리스 번호를 기록하게 된다. 이 때 기준선 문서에 기록된 릴리스 번호는 기준선 문서를 구성하는 모든 산출물들의 버전에 태그로 연결된다. 형상 관리 담당자가 모든 기준선 문서에 대해서 이러한 작업을 수행하면 결과적으로 릴리스를 구성하는 모든 산출물들이 릴리스 번호를 통해 연관 관계를 맺게 된다.

참고로 Fig. 6에 나타난 기준선 문서와 기준선 문서에 포함된 산출물들은 연관 관계 식별의 개념을 설명하기 위해 일부만을 나타낸 것이다. Fig. 6에서 나타난 기준선 문서와 산출물들이 릴리스 번호에 의해서 어떻게 연관 관계를 맺고 있는지 정리해보면 Table 2와 같다. Table 2에 나타난 것과 같이 시스템의 특정 릴리스 번호를 통해 기준선 문서 및 전체 산출물들이 어떠한 연관 관계를 맺고 있는지 추적할 수

Table 2. Relationships among Baselined Document and Artifacts through Release Number

	Baselined Document	Requirement Specification	Design Specification	Source Code(Payment Component)	
	Artifact	Requirement Specification.doc	Design Specification.doc	OrderPayment.jsp	PaymentList.jsp
Release No.	release_1_0_0	REV_00	REV_01	REV_00	
		1.2	1.3	1.2	1.2
	release_1_1_0	REV_01	REV_02	REV_01	
		1.5	1.4	1.3	1.3

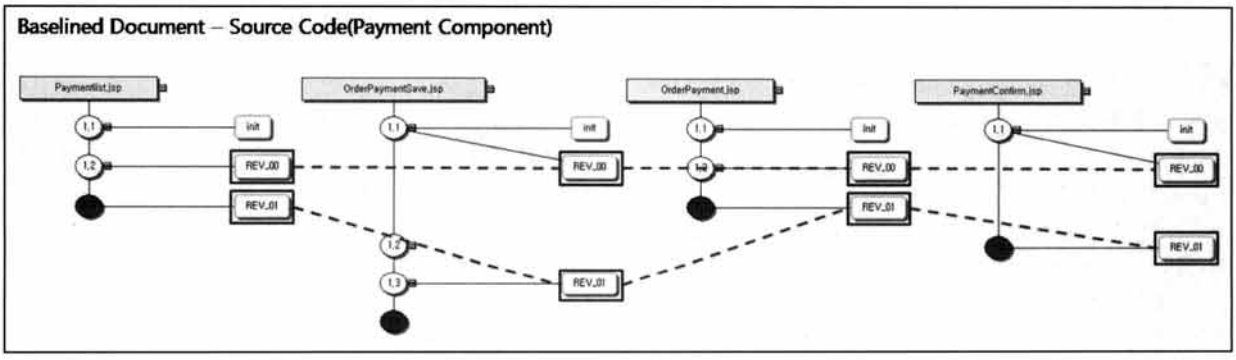

Fig. 5. Relationships among Artifacts included Baselined Document

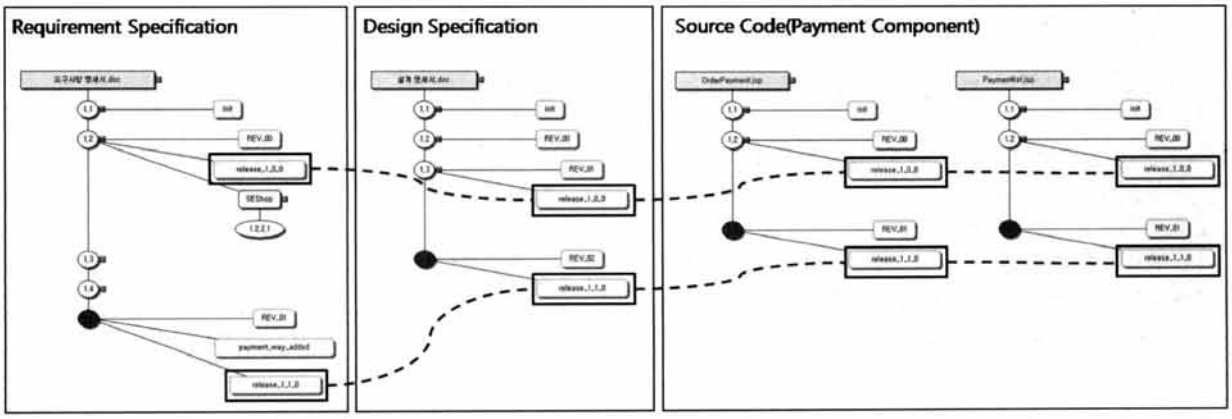

Fig. 6. Relationships among Baselined Document and Artifacts through Release Number

있다. 특정 릴리스 번호에 대한 기준선 문서의 연관 관계 및 전체 산출물들의 연관 관계를 정확하게 추적함으로써 이전 릴리스에 대한 변경 및 기능 개선 등의 작업을 훨씬 효율적으로 수행할 수 있다.

지금까지 설명한 세 가지 유형의 추적성 관리 기법은 Fig. 3에서 설명한 각 계층별 추적성 정보를 나타내는데 사용된다. 산출물의 버전에 기준선 문서의 수정 버전이나 시스템의 릴리스 번호를 태그로 연결시키는 과정은 본 연구에서 개발한 시스템에 의해 자동으로 이루어진다.

6. 연구 평가

본 장에서는 형상 관리와 추적성 관리를 위한 핵심적인 기능 요구사항들을 기준으로 본 논문에서 제시한 시스템을 평가하고자 한다. 먼저 Marcus가 제시한 추적성 관리 지원 도구의 요구사항들과 그 외 형상 관리 도구가 갖춰야 할 핵심적인 기능 요구사항들을 기준으로 정성적인 기능 평가를 수행한다. 이후 주로 사용되는 형상 관리 도구들과 기능적 관점에서 비교하고 그 결과를 정량적인 결과로 나타낸다 [33, 34].

6.1 추적성 관리 및 형상 관리 요구사항에 대한 정성적 평가

Table 3은 Marcus가 제안한 추적성 관리 요구사항과 그 외 형상 관리 도구들이 갖춰야 할 핵심적인 기능 요구사항들을 목록화한 것이다. 또한 본 논문에서 제시한 시스템이 각 요구사항들을 어떻게 만족시키는지 설명하고 있다. Table 4는 Table 3에서 나열한 요구사항들을 기존 상용 도구들과 본 논문의 시스템이 어느 정도 만족하고 있는지 나타내고 있다.

Table 4에서 비교 대상인 시스템들 가운데 CVS와 Subversion은 공개 소프트웨어로서 기본적인 버전 관리 기능들을 제공하고 있으나, 형상 관리 시스템이 갖춰야 할 변경 통제 기능이나 프로젝트 관리 기능 등은 제공하고 있지 않다. 그 외 형상 관리 도구들은 CVS나 Subversion보다는

Table 3. Requirements of Traceability Management and Configuration Management and Evaluation of Proposed System

Req. no	Requirements of Traceability and Configuration Management	Evaluation of Proposed System
1	Visualize and store traceability information among various artifacts, regardless of the extraction methodology used.	Provide traceability information of artifacts, regardless of the extraction tools used. Traceability information of each artifact is provided by visualized version graph.
2	Interface or integrate with traceability link recovery tools.	Proposed system support itself identification of traceability link based on change history. But, this requirement is partially satisfied because the system can't provide traceability information based on requirements.
3	Allow the user to browse the traceability links.	Users can confirm the change history of single artifact and the relationships of various artifacts through workspace system.
4	Interoperate with other software engineering tools (e.g., analysis tools, document management tools, etc.)	Interoperability is satisfied by integration of version management system and configuration management system. But, interoperation with other CASE tools such as design tools or development tools is limited.
5	Provide comprehensive configuration management and change tracking facilities.	Provide the tracking facility for configuration management process and change through integration of configuration management system and personal workspace.
6	Support user querying and filtering of the traceability links.	Support version graph viewing function of single artifact and version link identifying function for tracing relationships of various artifacts.
7	Analyze and summarize the data on the traceability process and links.	Proposed system can extract and visualize the traceability information. But the analyze and summarize function are not provided in the system.
8	Integrate local traceability and global traceability.	Support tracing not the change in personal workspace but also the change in configuration management system.
9	Support branching function for manage multiple versions for various customer.	Provide branching function to develop same artifact with multiple flow.
10	Support tagging or labeling for release.	Support tagging function to indicate what version of artifact is linked with release number of system.
11	Support traceability for release.	Support tracing what version of artifact is linked with release through attach the release number as tag. Also tracing relationships of artifacts related some release number is supported.
12	Provide the environment in which parallel development is available.	Developers can work independently in own workspace by using branching function.

Table 4. Comparison of Configuration Management Tools for the Requirements of Traceability Management and Configuration Management (○ - satisfied / △ - partially satisfied / × - not satisfied)

SCM tools / Req. No.	CVS	SubVersion	Perforce SCM	Rational ClearCase	AccuRev	Borland StarTeam	Proposed System
1	△	△	○	○	○	○	○
2	×	×	△	△	△	△	△
3	×	×	△	△	△	△	○
4	△	△	△	○	△	△	△
5	×	×	×	○	×	×	○
6	×	×	×	×	×	×	○
7	×	×	×	×	×	×	×
8	×	×	△	○	△	×	○
9	○	○	○	○	○	×	○
10	○	○	○	○	×	×	○
11	×	×	○	○	×	×	○
12	○	○	△	△	△	×	○

향상된 기능과 편의성을 제공하고 있으나, 변경 이력이나 산출물의 연관 관계와 같은 정보를 효과적으로 추적하는데 있어 다소 불편함이 있다. 특히 개인 작업 공간에서 발생한 변경, 즉 형상 관리 시스템을 통해 공식화되지 않은 개인적인 변경에 대해서는 제대로 추적할 수 없은 경우들이 많다. 또한 형상 관리 시스템과 개인 작업 공간이 엄격하게 구분되어 효율적인 병렬 개발을 방해하는 경우도 발견할 수 있다. 또한 본 논문의 핵심인 릴리스에 대한 추적 기능들(태깅 혹은 라벨링)이 일부 형상 관리 시스템에서는 제공되지 않고 있다. Tabel 4의 평가 결과에 대한 근거는 참고문헌 16번과 34번에서 확인할 수 있다.

6.2 추적성 관리 및 형상 관리 요구사항에 대한 정량적 평가

본 장에서는 추적성 관리 및 형상 관리 요구사항에 대한 정성적 평가 결과를 토대로 각 형상 관리 도구들을 정량적으로 평가하고자 한다. Table 3의 12가지 요구사항들을 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 0, 5, 10 세 가지 점수 부여

위의 평가 모델을 토대로 비교 대상인 형상 관리 시스템들과 본 논문에서 제안한 시스템을 정량적으로 평가한 결과가 Fig. 7에 나타나 있다.

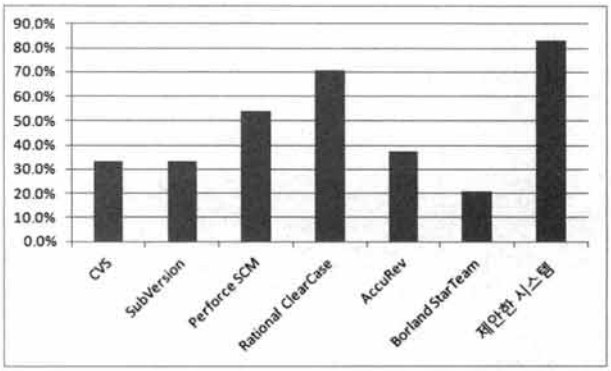

Fig. 7. Evaluation Result of Each System for 12 Requirements

7. 결론 및 향후 연구

소프트웨어 산출물에 대한 추적은 고품질의 소프트웨어를 개발하는데 있어 매우 중요하다. 개발자들은 추적을 통해 자신들이 올바른 소프트웨어를 만들고 있다는 확신을 갖게 되며, 고객은 정확한 추적을 통해 개발된 소프트웨어에 대해 높은 신뢰감을 갖게 된다. 특히 소프트웨어 시스템의 릴리스에 대한 전체 산출물들의 연관 관계를 추적하는 것은 기존 릴리스의 문제를 해결하거나 새로운 기능 개선에 대한 요구가 있을 때, 변경 작업의 효율성을 높이기 위해 필요하다.

소프트웨어에 대한 추적은 다양한 방법으로 연구되어 왔다. 추적의 단위와 표현 방법, 추적을 지원하는 방법, 추적의 자동화 수준 등에 따라서 많은 연구가 진행되어 왔다. 소프트웨어 산출물의 변경 이력과 그로 인한 서로 다른 산출물들 사이의 연관 관계를 정확하게 추적하는 능력이 추적성으로 정의될 수 있는데, 본 논문뿐만 아니라 기존의 많은 연구들이 이러한 추적성을 얼마나 충분히 지원할 수 있는가에 초점을 두었다.

본 논문은 변경 이력을 기반으로 소프트웨어 산출물의 진화 과정과 산출물들 사이의 연관 관계를 추적할 수 있도록 지원하는 방법을 제시하였다. 단일 산출물의 진화에 대한 추적, 한 기준선 문서에 포함된 산출물들의 연관 관계 추적

은 선행 연구를 통해 소개한 바 있다. 본 논문은 선행 연구를 확장하여 시스템의 릴리스를 구성하는 기준선 문서 및 전체 산출물들의 연관 관계를 추적하는 방법을 제시하였다.

릴리스에 대한 추적을 위해 본 연구는 릴리스 번호를 기준선 문서에 기록하고, 산출물들의 버전에 태그로 부착시키는 방법을 제안하였다. 릴리스 번호는 기준선의 어떤 수정 버전이 혹은 산출물의 어떤 버전이 해당 릴리스에 포함되었는지 확인하는데 사용되며, 이를 통해 기준선 문서 및 전체 산출물들의 연관 관계를 추적할 수 있다. 릴리스 번호를 산출물들의 버전에 태그로 부착시키는 과정은 개인 작업 공간과 형상 관리 시스템이 통합된 환경에서 자동으로 처리된다.

관련 연구에서 소프트웨어 릴리스 관리는 변경 관리 프로세스와 형상 관리 프로세스를 통합할 때 효율적으로 이루어질 수 있다고 설명한 바 있다. 본 연구의 통합 환경은 ITIL과 ISO/IEC 20000-1에서 강조하는 세 가지 프로세스를 통합하고, 릴리스를 구성하는 요소들(기준선 문서 및 전체 산출물)의 연관 관계를 추적할 수 있도록 지원한다. 통합 환경에서 한 릴리스에 속한 전체 산출물들의 연관 관계를 효율적으로 추적할 수 있도록 하기 위해 시각화된 버전 그래프와 자동화된 태깅 기법을 사용하였다. 향후 특정 릴리스에 포함된 기준선 문서들의 연관 관계를 보다 쉽게 추적할 수 있도록 형상 관리 시스템의 기능을 개선하기 위한 추가적인 연구를 수행할 예정이다.

참 고 문 헌

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

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

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

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

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

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

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

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

[9] H. Kagdi, I. M. Jonathan, S. Bonita, "Mining Software Repositories for Traceability Links", 15th IEEE International Conference on Program Comprehension, 2007.

[10] H. Kagdi, S. Yusuf, J. I. Maletic, "Mining Sequences of Changed-files from Version Histories", in Proceedings of 3rd International Workshop on Mining Software Repositories, pp. 47-53, Shanghai, China, May, 2006.

[11] T. Zimmermann, A. Zeller, Weibgerber P., Diehl S., "Mining Version Histories to Guide Software Changes", IEEE Transactions on Software Engineering, Vol.31, No.6, pp. 429-445, 2005.

[12] S. Sundaram, J. H. Hayes, A. Dekhtyar, "Baselines in Requirements Tracing", in Proceedings of Workshop on Predictive Models of Software Engineering, pp.12-17, St. Louis, May, 2005.

[13] S. Harvey, C. Parvathi, J. R. Daniel, S. Mahadevan, "Discovering Dynamic Developer Relationships from Software Version Histories by Time Series Segmentation", 23rd IEEE International Conference on Software Maintenance, pp.415-424, Paris, Oct., 2007.

[14] G. Harald, J. Mehdi, K. Jacek, "CVS Release History Data for Detecting Logical Coupling", in Proceedings of the 6th IEEE International Workshop on Principles of Software Evolution", 2002.

[15] D. Y. Kim, C. Youn, "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.

[16] D. Y. Kim, C. Youn, "Traceability Enhancement Technique for Dependency Relations of Software Artifacts based on the Integration of Software Configuration Management System and Personal Workspace", The KIPS Transactions, Vol. 18-D, No.6, 2011.

[17] R. Conradi, B. Westfechtel, "Version Models for Software Configuration Management", ACM Computing Surveys, Vol. 30, No.2, June, 1998.

[18] Office of Government Commerce, ITIL Service Operation, The Stationary Office, UK, 2007.

[19] ISO/IEC 20000-1, Information technology - Service management-Part 1: Specification

[20] TTA.KO-10.0256, "Guideline for Configuration and Change Management of Information Systems", 2007.

[21] K. Mohan., P. Xu and B. Ramesh, "Improving the Change

Management Process", Communications of the ACM, Vol.51, No.5, pp.59-64, May, 2008.

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

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

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

[25] Y. J. 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.

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

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

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

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

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

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

[32] http://www.processdox.com/, "Configuration, Change and Release Management Policies and Procedures Guide"

[33] A. Marcus, X. Xie, D. Poshyvanyk, "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.

[34] T. Fatma, "Evaluating Software Configuration Management Tools for Opticon Sensors Europe B.V.", Masters Thesis Software Engineering, 25 June, 2004.

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 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