J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

소프트웨어 요구사항 추적 및 변경 관리를 위한 시각화 모델☆

Visualization models for tracking software requirements and managing their changes

송 유 리[1] 김 현 수[1*]
YooRi Song Hyeon Soo Kim

요 약

이 연구에서는 소프트웨어 개발 과정에서 소프트웨어 시스템의 품질을 높이기 위해 요구사항에 대한 추적성 관리 및 변경 관리를 체계적으로 수행하는 요구사항 모니터링 방법을 제안한다. 이를 위해 산출물 간 연관 관계를 정의하기 위한 추적 매트릭스와 개별 산출물들에 대한 변경 관리를 체계적으로 수행하기 위한 산출물 변경 이력 관리 모델을 제시한다. 또한 특정 산출물이 변경될 때 연관된 산출물에 변경이 파급되는 상황을 쉽게 파악할 수 있도록 하는 시각적 대시보드도 제안한다.

☞ 주제어 : 소프트웨어 요구사항, 추적성, 소프트웨어 변경관리, 시각화 모델

ABSTRACT

This study proposes a requirements monitoring method that systematically performs traceability management and change management for requirements in order to improve the quality of software systems in the software development process. To this end, we present the artifact change history management models to systematically perform change management for individual artifacts and the traceability matrixes to define the relationship between artifacts. It also proposes a visual dashboard that makes it easy to grasp the situation in which changes are propagated to related artifacts when specific artifacts change.

☞ keyword : Software Requirement, Traceability, Software Change Management, Visualization Model

1. 서 론

통상적인 소프트웨어 개발 프로세스에서 요구사항이 추출되고 명세화가 되고나면 이 요구사항은 분석, 설계, 구현 단계에서 소프트웨어 개발을 위한 기준으로 사용된다. 그러나 일단 설계 요소로 변환된 요구사항은 더 이상 요구사항 자체로 관리되지 않는다. 한편, 하나의 소프트웨어가 생산될 때까지는 많은 변경이 발생하며, 변경이 거듭될 때마다 진전, 변형, 합병 등이 일어난다. 이러한 변경이 빈번하게 발생하면 소프트웨어 개발자들은 개정판(revision) 및 버전에 대한 지속적인 관리에 상당한 부담을 갖게 되며 혹여 이러한 관리를 지속하지 못할 경우에는, 잦은 구축 오류의 발생으로 인해 소프트웨어의 품질 저하, 납기의 지연 및 개발 비용의 낭비를 초래하게 된다[1].

1 Department of Computer Science & Engineering, Chungnam National University, Daejeon, 34134, Korea.
* Corresponding author (hskim401@cnu.ac.kr)
[Received 19 March 2021, Reviewed 29 March 2021, Accepted 13 April 2021]
☆ 이 연구는 충남대학교 학술연구비에 의해 지원되었음

본 연구에서는 소프트웨어 개발 프로세스 동안 소프트웨어 시스템의 일관성을 유지하고 품질을 확보하기 위해 요구사항에 대한 추적성 관리 및 변경 관리를 효과적으로 지원하는 요구사항 모니터링 모델을 제안한다. 이를 위해 우선 산출물 간 추적 매트릭스와 산출물의 변경 이력 관리 모델을 바탕으로 모니터링 시스템의 기본이 되는 추적 및 변경 관리를 위한 방법을 수립한다. 그 후 이를 결합하여 소프트웨어 개발 단계별 산출물간의 연관 관계를 그래프 형태로 표현한 시각화 모델을 제시한다.

2. 관련 연구

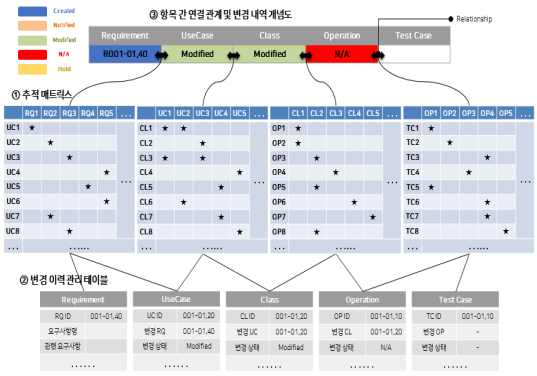
요구사항 추적성은 고객의 요구사항으로부터 소프트웨어 개발 과정에서 생산되는 산출물들 사이의 관계를 기술하고 추적하는 능력을 말한다[2]. 이러한 요구사항 추적을 통해 변경된 요구사항이 설계, 개발, 검증 단계에서 제대로 적용되었는지를 확인할 수 있으며, 이 과정에서 각 산출물 간의 관계를 이용한다.

먼저 추적성을 위해 산출물 간의 매핑과 이를 지원하

기 위한 모형을 제시한 방법들을 살펴본다. 연구 [3]에서는 소프트웨어 각 개발 단계 산출물 간의 추적 매트릭스(Traceability Matrix)를 제시하였다. 소프트웨어 개발 각 단계에 대한 추적 매트릭스를 적용함으로써 직관적이고, 1대 다 관계일 때 추적을 쉽게 수행할 수 있게 되었다. 추적 매트릭스는 정확한 추적 링크를 나타낼 수 있지만, 완전히 수동적인 방법으로 관리되기 때문에 복잡하고 큰 시스템의 경우 빈번하게 발생하는 변경으로 인해 요구사항과 산출물들의 추적 링크가 일치하지 않는 가능성이 존재한다. 연구 [4]에서는 기존 추적 테이블(Traceability Table) 기반 요구사항 추적의 단점을 해결하고자 추적 테이블의 장점은 그대로 이용하면서 가독성을 높이기 위해 변경 요구사항 별 버전을 확인할 수 있는 ID를 부여하는 방식을 택하였다. 추적 테이블은 단순 링크 방식에 속하며 산출물을 일렬로 순차적으로 연결하면서 추적하는 방식을 사용함으로써 사용이 편리하고 추적의 정확성이 높다. 전 개발 단계에서 추적 정보가 체계적으로 관리되지 않고 개발 산출물들만 존재하는 경우에 추적 정보를 식별하기 위한 기법에 대한 연구도 진행되었다. 이벤트 기반 추적 기법은 요구사항과 산출물 간의 연결을 설정하는 실행 모델과 시뮬레이션을 사용해 동적으로 추적하는 방법으로 산출물의 변경된 항목을 개발자에게 알리기 위한 변경 알림 패턴을 기반으로 동작한다[5]. 연구 [6]에서는 요구사항과 소스코드 간의 추적성을 유지하기 위하여 추적성 매트릭스나 추적성 테이블에 기반을 둔 방법을 사용하는 대신 자동화된 방법을 제시한다. 이 연구에서는 소프트웨어 시스템이 진화할수록 추적 링크가 저하되는 다양한 상황을 제시하고 그런 상황에 대해 시나리오를 정의한다. 그런 다음 소프트웨어 시스템의 연속 버전에서 발생하는 사전 정의된 변경 시나리오를 감지하기 위한 방법을 제시한다. 결과적으로 소스 코드 클래스 또는 메소드와 요구사항 사이의 양방향 추적 링크의 진화를 자동화하기 위해 TLE(Trace Link Evolver)를 제시한다. 연구 [7]에서는 기존 소프트웨어 형상관리 시스템이 소프트웨어 개발 문서나 코드 아티팩트에 대한 형상관리에 치중하는 반면에, 소프트웨어 연구 문서 아티팩트를 관리하지 않는 이유를 파악하고 이를 추적하고 관리하기 위한 콘텐츠 기반 구성 관리 시스템을 제시하고 있다. 요구사항 변경의 추적 과정을 자동화한 도구들은 추적 기능, 변경 관리 기능 등을 제공한다. 상용 제품으로 DOORS, Connect 등이 있다. DOORS는 요구사항 관리 도구로 강력한 추적성이 장점인 도구이다. 이 도구는 요구사항부터 서브요구사항 그리고 아키텍처, 설계 및 테스트 단계까지 모든 추적을 가능하게 한다[8]. Connect는 추적 관계를 기반으로 산출물들 간의 연관 관계뿐만 아니라 이해당사자들에 대해서도 연결 정보를 유지하는 추적성 관리 도구로써 감사 기능을 위해 변경 사항을 문서화하여 유지한다[9]. 이러한 도구들은 추적 테이블 형태의 입력을 바탕으로 링크 및 관계 매트릭스를 자동으로 생성하며 가시성이 있는 도표 및 그래프를 제공하여 높은 사용편의성을 제공한다.

3. 요구사항 추적 및 변경 관리를 위한 시각화 모델

이 절에서는 요구사항 추적 매트릭스와 변경 이력 관리 테이블을 기반으로 하는 소프트웨어 요구사항 추적 및 변경 관리 시각화 모델(Visualization Models for Requirement Tracking and Change Management: VM4RTCM)'을 제안한다. VM4RTCM은 기본적으로 이벤트 기반 추적 기법을 사용해 각 단계의 산출물이 변경될 때 연결된 하위 산출물과 해당 산출물을 생산한 담당자들에게 변경 알림을 보낸다.



(그림 1) VM4RTCM의 구성도
(Figure 1) Configuration diagram of VM4RTCM

그림 1은 이 논문에서 제안하는 VM4RTCM의 구성도이다. 먼저 소프트웨어 개발 단계에서 개발자들이 생산하는 산출물을 요구사항, 유스케이스 모델, 클래스 모델, 오퍼레이션 모델 및 구현, 테스트 케이스와 같이 5가지로 구분한다. VM4RTCM은 크게 3개의 요소들로 이루어져 있다. 첫 번째 구성 요소는 산출물들 간의 관계를 정의하고 추적하기 위한 추적 매트릭스이다. 이를 통해 요구사

항 변경 발생 시 변경 사항을 반영할 관련 요소들을 파악할 수 있다. 두 번째 구성 요소는 변경 이력 관리 테이블이다. 각 개발 단계에서 개발자들은 산출물의 변경 이력을 관리하기 위해 이와 같은 테이블을 생산한다. 세 번째 구성 요소는 변경 이력, 변경 사항 반영 여부, 변경 알림 등을 모니터링 할 수 있는 대시보드이다. 대시보드에는 변경된 요구사항의 반영 상황이 구별되어 표현되므로 사용자는 변경 사항의 반영 여부를 쉽게 파악할 수 있다.

3.1 추적 매트릭스

요구사항이 소프트웨어에 적절히 반영되었는지 여부를 확인하는 작업은 분석, 설계, 구현, 테스트 단계까지 요구사항 추적성 관리를 통해 이루어진다. 추적을 용이하게 수행하기 위해서는 요구사항 항목에서 분석/설계 항목으로, 분석/설계 항목에서 코드 항목으로, 코드 항목에서 테스트 케이스로 매핑 하는 것이다. 이러한 추적성 관리를 위해서는 일반적으로 추적 매트릭스를 사용한다[3].

(표 1) 요구사항과 유스케이스 간의 추적 매트릭스
(Table 1) Traceability matrix between requirements and use cases

	RQ001	RQ002	RQ003	RQ004	…
Manag.	Kim	Kim	Song	Song	…
UC01	✓				
UC02	✓				
UC03		✓			
UC04			✓		
UC05				✓	
UC06				✓	
UC07		✓			
…					

이 논문에서는 요구사항과 유스케이스 사이의 관계, 유스케이스와 클래스 사이의 관계, 클래스와 오퍼레이션 사이의 관계, 오퍼레이션과 테스트 케이스 사이의 관계로 총 4개의 추적 매트릭스를 구성하여 요구사항과 하위 산출물 사이의 추적성을 보장한다. 매트릭스에서 연관되는 두 요소의 집합은 행과 열로 표시되며, 요소 간의 직접적인 연관성이 있음은 ✓ 기호로 표현한다. 아울러 각 산출물에 대한 담당자도 함께 표현함으로써 추후 변경이 필요할 경우 해당 담당자에게 통보될 수 있도록 구성된다. 표 1은 요구사항과 유스케이스 간의 추적 매트릭스를 표현한 예이다. 또한, 추적 매트릭스는 요구사항이 생성, 삭제, 변경될 때 연관성을 쉽게 추가, 삭제, 변경할 수 있도록 데이터베이스에 저장된다. 데이터베이스에 저장할 때 표 2와 같이 연관성을 갖는 요소의 쌍으로 저장하며, 양방향 추적성을 위해 두 요소가 모두 키(Key)가 될 수 있다.

(표 2) 요구사항과 유스케이스 간의 관계 정보의 예
(Table 2) Examples of relation information between requirements and use cases

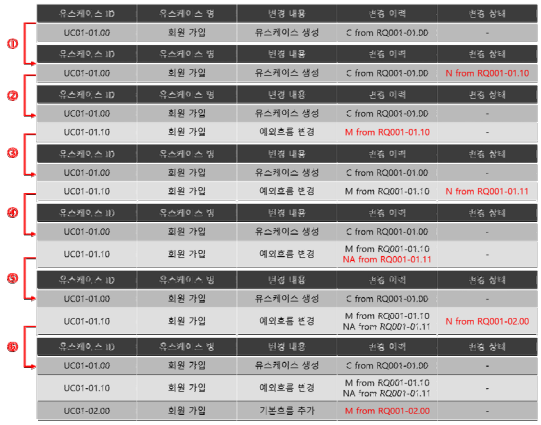
Requirement (Key 1)	UseCase (Key 2)
RQ001	UC01
RQ001	UC02
RQ002	UC03
RQ002	UC07
RQ003	UC04

3.2 변경 이력 관리 테이블

이 논문에서는 소프트웨어 산출물의 변경 내역을 유지하고 확인하기 위한 수단으로 변경 이력 관리 테이블을 제안한다. 변경 이력 관리 테이블은 요구사항, 유스케이스, 클래스, 오퍼레이션, 테스트 케이스 별로 별도로 유지되며, 각 테이블은 식별자와 변경 내용, 연관된 상위 단계 산출물의 변경으로 인해 발생한 변경 이력, 변경 상태 등에 대한 정보를 포함한다. 식별자는 산출물의 종류를 나타내는 구분 기호(RQ(Requirement), UC(UseCase), CL(Class), OP(Operation), TC(Test Case))와 일련 번호, 버전 번호로 구성된다. 버전 번호는 xx.yy로 표현되는데, 낮은 수준의 변경은 소수점 이하에 반영하고 높은 수준의 변경은 소수점 이상에 반영한다. 변경 내용은 해당 산출물에 가해진 변경 내용을 나타낸다. 변경 상태와 변경 이력은 생성됨(Created), 변경됨(Modified), 알림(Notified), 보류(Hold), 적용 불가(Not Applicable)로 구분한다. 표 3은 각 변경 상태에 대한 설명이다.

(표 3) 변경 상태의 정의
(Table 3) Definition of change states

변경 상태	설명
Created (C)	초기 생성됐거나 새로운 산출물과 연관된 상태
Modified (M)	산출물이 변경된 상태 (하위 산출물로 변경 알림)
Notified (N)	이해당사자에게 변경이 공지된 상태 (하위 산출물로 전파 보류 상태)
Hold (H)	변경 사항 적용을 보류한 상태 (하위 산출물로 전파 보류 상태)
Not Applicable (NA)	변경 사항 적용 불가 또는 불필요한 상태 (변경 전파 프로세스 종료)



(그림 2) 유스케이스 변경 이력 관리 테이블의 업데이트 시나리오
(Figure 2) Update scenario of use case change history management table

이를 이용하여 변경 이력 및 변경 상태를 작성하는 규칙은 다음과 같이 간략히 기술할 수 있다.

변경 이력 : 종료된 상태 (C, M, 또는 NA) + 'from' + 상위 산출물의 식별자

변경 상태 : 현재 진행 중인 상태 (N 또는 H) + 'from' + 상위 산출물의 식별자

여기서, C, M, NA, N, H는 변경 상태의 종류를 나타낸다.

그림 2는 유스케이스 1(UC01)과 연관되어 있는 요구사항 1(RQ001)의 변경에 따라 유스케이스 변경 이력 관리 테이블이 업데이트되는 시나리오를 보여주는 예시이다. 그림에서 보듯이 테이블은 유스케이스 식별자와 유스케이스 이름, 변경 내용, 변경 이력, 변경 상태를 포함하며, 유스케이스 식별자는 UC에 일련번호를 추가하고 '-'을 연결한 다음, 버전 번호를 붙여 생성한다. 유스케이스의 변경 이력과 변경 상태는 이전 산출물인 요구사항을 기준으로 작성한다. 그림 2의 업데이트 시나리오의 흐름 및 그에 따른 테이블의 변경 내용은 다음과 같다.

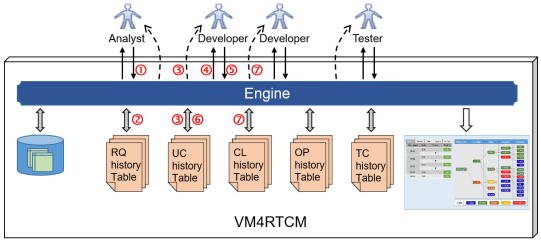
1. [RQ001-01.10] 요구사항 변경에 따른 변경 공지
   → 변경 상태에 N from RQ001-01.10 기록
2. [RQ001-01.10] 변경 내용 반영
   → 변경 내용 반영에 따라 UC01-01.10으로 변경
   → 변경 이력에 M from RQ001-01.10 기록
3. [RQ001-01.11] 요구사항 변경에 따른 변경 공지
   → 변경 상태에 N from RQ001-01.11 기록
4. [RQ001-01.11] 변경 내용 반영 불가
   → 변경 이력에 NA from RQ001-01.11 기록
5. [RQ001-02.00] 요구사항 변경에 따른 변경 공지
   → 변경 상태에 N from RQ001-02.00 기록
6. [RQ001-02.00] 변경 내용 반영
   → 변경 내용 반영에 따라 UC01-02.00으로 변경
   → 변경 이력에 M from RQ001-02.00 기록

위의 1과 2의 과정에서 각 산출물 담당자와 시스템 간의 상호작용은 그림 3과 같으며 세부적인 동작은 다음과 같은 방식으로 일어난다.

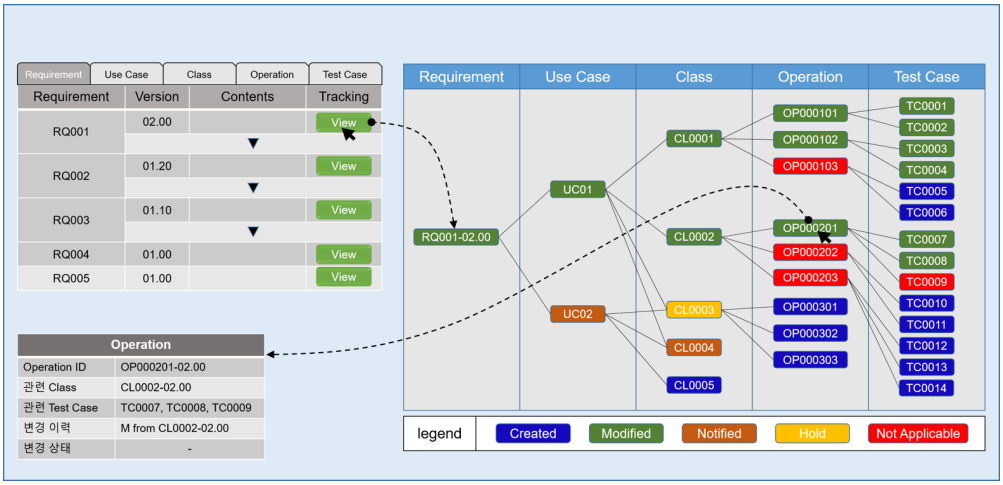
① [요구사항 담당자] RQ001-01.00 변경 후 시스템에 check-in
② [시스템] RQ 변경이력관리 테이블 업데이트
③ [시스템] UC 담당자에게 변경 공지 및 UC 변경이력 관리 테이블에 변경 공지 반영
④ [유스케이스 담당자] 변경 공지 내용 검토 및 변경을 위해 해당 UC check-out
⑤ [유스케이스 담당자] UC01-01.00 변경 후 시스템에 check-in
⑥ [시스템] UC 변경이력관리 테이블 업데이트
⑦ [시스템] CL 담당자에게 변경 공지 및 CL 변경이력관리 테이블에 변경 공지 반영

이와 같은 일련의 과정은 다른 산출물에서도 동일한 방식으로 각 산출물 담당자와 시스템 간의 상호작용이 이루어진다.

그림 3에서 엔진 부분은 VM4RTCM의 핵심 기능을 담당하는 부분으로 추적 매트릭스, 변경 이력 관리 테이블, 여러 종류의 산출물을 저장하고, 검색하며, 시스템과 여러 담당자 및 산출물 간에 발생하는 이벤트를 처리하고, 변경을 통보하며, 최종적으로 다양한 처리 사항을 대시보드에 시각화하는 기능을 수행한다.



(그림 3) 시스템과 이해당사자 간의 상호작용
(Figure 3) Interactions between system and stakeholders



(그림 4) 요구사항 추적 및 변경 관리를 위한 시각화 대시보드
(Figure 4) Visualization dashboard for requirements tracking and change management

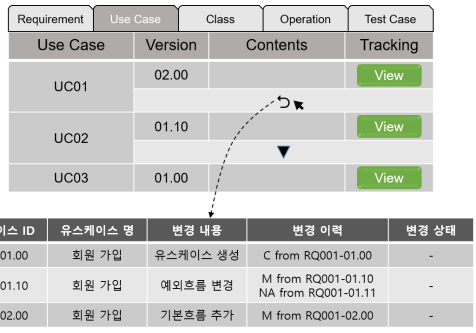
3.3 요구사항 추적 및 변경 관리 시각화 대시보드

이 절에서는 소프트웨어 요구사항에 대한 추적 매트릭스와 변경 이력 관리 테이블을 기반으로 요구사항 변경 내용의 전파 여부와 변경 이력을 추적하고 모니터링 할 수 있는 시각화 대시보드에 대하여 기술한다.

시각화 대시보드는 크게 변경 이력 추적 시각화 모델과 산출물 연결 시각화 모델로 구성된다. 여기서 변경 이력 추적 시각화 모델은 앞 절에서 제시한 변경 이력 관리 테이블의 내용을 시각화하기 위한 모델로써 산출물 리스트업 기능, 변경 이력 확인 기능 등을 포함한다. 산출물 연결 시각화 모델은 추적 매트릭스와 변경 이력 관리 테이블의 내용을 결합하여 산출물들 간의 연관 관계와 산출물의 변경 사항을 시각화한 모델이다. 그림 4는 이러한 내용이 반영된 시각화 대시보드의 모습이다.

사용자는 화면 좌측의 변경 이력 추적 시각화 모델을 통해 요구사항, 유스케이스 등의 ID 별 최신 버전을 확인할 수 있으며, 초기 버전 이후의 변경 이력을 확인할 수 있다. 만약 설계자나 개발자 등의 사용자가 자신이 담당하는 산출물의 상위 산출물 및 하위 산출물들의 변경 이력 정보를 확인하고 싶다면, 대시보드 우측의 산출물 연결 그래프를 통해 추적 한 다음 원하는 산출물을 선택하면 변경 이력 정보를 확인할 수 있다.

그림 5는 변경 이력 추적 시각화 모델에서 유스케이스 목록을 시각화 한 내용을 보여주는 것으로서 유스케이스 식별자, 최종 버전, 유스케이스에 대한 간략한 내용 설명과 산출물 연결 추적 버튼을 포함하고 있다. 각 유스케이스를 처음 생성할 때에는 버전 번호가 01.00으로 자동 부여 되며, 기존 유스케이스에 변경 내용이 반영될 경우에는 버전이 증가하게 된다. 목록상의 유스케이스별 버전은 특정 유스케이스의 최신 버전이 기본적으로 출력되며, 초기 생성 버전에서 변경이 발생한 유스케이스의 경우 그림과 같이 유스케이스 내용 하단에 ▼ 버튼이 생성된다. 이 버튼을 클릭할 경우, 그림 5의 하단에 보이는 것처럼 변경 이력과 변경 내용 등이 포함된 변경 이력 관리 테이블이 나타나게 된다. 이런 사항은 목록 상단의 탭을 통해 알 수 있듯이 유스케이스뿐만 아니라 요구사항, 클래스, 오퍼레이션, 테스트 케이스 목록에도 동일하게 적



(그림 5) 유스케이스 목록의 시각화와 변경 이력의 추적
(Figure 5) Visualization of use case lists and tracking of change history

용된다.

산출물 연결 시각화 모델은 추적 매트릭스와 변경 이력 관리 테이블의 정보를 바탕으로 특정 요구사항, 유스케이스, 클래스, 오퍼레이션, 테스트 케이스 산출물들 간의 관계와 변경 내용 적용 여부를 보여준다. 그림 4에서 오른쪽의 그래프 형태의 그림은 산출물 연결 시각화 모델을 구현한 산출물 연결 그래프의 모습이다. 산출물 연결 그래프는 산출물들 간의 추적 매트릭스로부터 추출한 정보를 이용하여 산출물들 간의 연관 관계를 그래프 형태로 표현하고, 특정 산출물에서 변경 내용의 반영 여부는 해당 산출물의 변경 이력 관리 테이블로부터 추출한 정보를 이용하여 색상으로 시각화한다.

그림 4의 산출물 연결 그래프는 요구사항 RQ001이 02.00 버전으로 변경된 이후에 하위 산출물들에서 해당 요구사항에 대한 변경 내용의 반영 여부를 시각화한 사례를 보여준다. UC01을 예로 들면 UC01의 변경 이력 관리 테이블에서 가장 최근의 변경 이력 상황이 'M from RQ001-02.00'이므로 UC01은 변경된 상태로 표시된다. 계속해서 UC01의 변경 사항은 CL0001, CL0002에는 반영된 상황이며, CL0003에서는 변경 사항 반영을 보류 중인 상황이고, CL0004에는 통보된 상황임을 알 수 있다. 클래스와 연결된 오퍼레이션 OP000101은 변경을 반영한 상태이고, OP000103은 요구사항 변경을 적용할 필요가 없는 상황임을 알 수 있다. OP000301에서 OP000303은 현재 상위 노드인 CL0003이 보류 중인 상태이므로 아직 변경 내용에 대한 알림을 받지 못한 상태이다. 계속해서 TC0005, TC0006은 상위 노드 OP000103에 요구사항 변경 내용을 적용할 필요가 없음으로 인해 변경 사항이 통보될 필요가 없는 상태이다. 이렇게 각 산출물의 상태를 여러 색상으로 표현함으로써 사용자는 요구사항 RQ001-02.00의 변경이 어떤 산출물까지 반영되었는지를 한 눈에 확인할 수 있다.

산출물 연결 그래프에서 원하는 노드를 선택하면 해당 노드와 관련된 정보를 보여주는 테이블이 생성된다. 이 테이블에서는 산출물 연결 그래프에서 연결된 상위 산출물과 하위 산출물, 그리고 변경 이력과 변경 상태 등을 볼 수 있다. 그림 4에서 왼쪽 하단의 오퍼레이션 테이블은 OP000201 노드를 선택했을 때 나타나는 테이블의 모습이다.

이와 같이 요구사항 변경에 따른 변경 이력 추적을 가능하게 함으로써 개발자들은 변경 내용의 반영 여부 파악 및 이전 버전으로의 롤백을 수월하게 수행하는데 도움이 된다. 또한, 변경 과정에서 관련된 이해당사자들에게 변경 내용을 반영하도록 통보함으로써 업무의 효율을 높일 수 있다.

4. 제안 모델 비교 검증

이 절에서는 우리의 연구 결과인 VM4RTCM과 기존 연구들을 표 4의 기준을 바탕으로 비교 분석한다.

표 4는 요구사항 추적 및 변경 관리를 위한 소프트웨어의 주요 기능들을 바탕으로 작성된 비교 항목이다. 이 연구에서는 요구사항 추적성과 변경 관리가 가중 중요한 목표이고 이를 효과적으로 지원하며 사용하기 쉽도록 사용자의 편의성을 높이는 것 또한 중요한 목표이다. 이런 측면에서 비교 항목을 추적성, 변경 관리, 사용성으로 정하였고, 이 항목들을 세분화하여 전체 9개의 세부항목으로 비교 항목을 설정하였다. 이 비교 항목 중 추적성과 변경 관리 관점에서는 기능을 제공하는 경우 ○, 기능 제공은 가능하나 기능을 활용하기 위하여 사용자의 추가적인 노력이 요구되는 경우△, 기능 제공이 불분명한 경우 X로 표기하였다. 사용성 관점은 시각적으로 정보가 제공되며 직관적으로 사용하기 편리한지 여부를 비교하기 위한 항목으로서 시각화 수단(예, 그래픽)의 사용 여부, 정제된 텍스트 형태(예, 테이블)의 사용 여부를 판단하여 모두 가능하면 ○, 하나만 가능하면△, 모두 불가능하면 X로 표기하였다.

(표 4) 연구 비교 항목
(Table 4) Comparison items for the researches

항목	세부항목	설명
추적성	연관성 추적	산출물 간에 올바른 연관 관계 추적이 가능한가?
	양방향 추적	양방향 (전방향 및 역방향) 추적이 가능한가?
	개발 전 단계 추적	개발 과정 전 단계에 걸쳐 추적이 가능한가?
변경 관리	변경 추적	변경 사항의 진원지 추적이 가능 한가?
	버전별 변경 추적	특정 버전별 변경 추적 및 변경 반영 여부 확인이 가능한가?
	변경 알림	산출물의 변경 발생 시 연관되는 산출물 및 이해당사자에게 변경 사항이 통보되는가?
	변경 이력 관리	산출물의 변경 이력 관리가 가능한가?
사용성	가시성	산출물 간의 연관 관계 및 변경 사항 반영 여부를 쉽게 파악할 수 있는가?
	편의성	직관적으로 사용 가능한가?

(표 5) 연구 비교 결과
(Table 5) Comparison results for the researches

세부항목	비교 대상 연구					
	연구 [3]	연구 [4]	연구 [6]	Connect	Doors	VM4RT CM
연관성 추적	○	○	○	○	○	○
양방향 추적	○	○	○	○	○	○
개발 전 단계 추적	○	○	X	○	○	○
변경 추적	X	○	○	○	○	○
버전별 변경 추적	X	△	○	X	X	○
변경 알림	X	X	△	○	○	○
변경 이력 관리	X	○	X	△	○	○
가시성	X	△	X	○	△	○
편의성	○	X	X	△	△	○

표 5는 표 4의 비교 항목을 통해 우리 연구와 다른 연구의 소프트웨어를 비교한 결과표이다. 연구 [3]은 소프트웨어 개발 프로세스의 단계를 5단계로 나누고 산출물을 7가지로 정의하였다. 인접한 단계의 산출물들 간의 추적성을 하나의 매트릭스로 표현하였고, 산출물의 종류는 구현 단계까지의 산출물만을 제시해 총 4개의 매트릭스를 구성하였다. 그러나 어떤 단계에서 수정이 발생하면 추적을 통해 유지보수를 할 수 있지만 구체적인 변경 관리를 위한 방법은 제시하지 않는다. 이에 비해 우리 연구에서는 변경 이력 관리 테이블을 제시해 구체적인 변경 내용과 상태를 관리하고 추적 매트릭스와 변경 이력 관리 테이블을 결합해 그래프 형태로 시각화함으로써 변경 추적뿐만 아니라 변경 내역도 쉽게 파악할 수 있다. 연구 [4]는 요구사항 적용의 일관성 및 완전성 검증을 위해 프로젝트 시작부터 최종 단계까지 30여개의 산출물별 세부 단계로 분할된 추적 테이블을 제안하였다. 정규화된 표현 방법을 사용하므로 요구사항 적용의 완전성 및 추적 정확성이 향상될 수 있다. 우리 연구에서는 이 연구에서 부족한 가시성을 높여 산출물 간의 연관 관계와 변경 사항에 대한 반영 여부 등을 쉽게 파악할 수 있다. 연구 [6]은 추적 링크의 진화를 효과적으로 수행할 수 있으나 가시성은 전혀 고려하고 있지 않다. 요구사항과 코드 간의 링크의 변화를 관리하기 위해 만들어졌기 때문에 설계 단계의 산출물과의 연관 관계 및 변경 근거 추적이 불가능하다는 단점이 있다. 이에 비해 우리 연구는 소프트웨어 개발 과정 전 단계에 걸쳐 연관 산출물 추적이 가능하다는 장점이 있다. Connect는 기본적으로 추적성 관리를 위한 도구로써 산출물들뿐만 아니라 이해당사자들에 대해서도 연결을 통한 추적 관계를 유지한다. 그렇지만 단지 연관된 산출물들 간의 연결 관계에만 집중하므로 각 개발 단계의 진행 상황이나 개발 단계별 산출물들을 일목요연하게 파악하기 어렵다. 이에 반해 우리의 방법은 각 단계의 산출물들을 대시보드에서 단계 별로 나열하므로 단계의 진행 상황뿐만 아니라 산출물들 간의 관계를 전체적으로 쉽게 파악할 수 있다. DOORS의 경우 각각의 요구사항을 상세히 제공하고 그들 간의 연관 관계를 생성, 삭제, 추적하는 것이 가능하나 시스템 전체 관점에서 형상을 구성하는 아티팩트를 조망하기 어려운 측면이 있다. 또한 산출물 내의 변경에 대해서도 간단한 텍스트 수정도 유지하는 등 너무 많은 정보를 유지하고 관리함으로 인해 오히려 변경 추적을 어렵게 한다. 이에 반해 우리의 방법은 시각화 대시보드를 통해 산출물을 각각의 노드로 표현함으로써 소프트웨어 시스템의 하나의 형상을 구성하는 아티팩트들을 쉽게 파악할 수 있으며, 산출물 내의 변경에 대해 추상화된 변경 이력 정보를 유지함으로써 변경 추적을 쉽게 수행할 수 있다.

5. 결 론

소프트웨어 개발 과정에서 소프트웨어 시스템의 품질을 높이기 위한 방안 중에 하나가 요구사항의 변경에 대한 체계적인 관리이다. 이 연구에서는 소프트웨어 개발 전 단계에 걸쳐 요구사항 추적 및 변경 관리를 체계적으로 수행할 수 있도록 소프트웨어 요구사항 추적 및 변경 관리를 위한 시각화 모델을 제안하였다.

이 모델은 소프트웨어 개발 각 단계에서 제작되는 산출물 간의 연관 관계를 정의하기 위해 추적 매트릭스를 제공하며, 이를 통해 산출물의 변경 발생 시 연관 산출물을 파악하기 위한 양방향 추적이 가능하게 되었다. 아울러 각 산출물들에 대해 변경 이력 정보를 관리함으로써 개별 산출물들에 대한 변경 관리를 체계적으로 수행할 뿐만 아니라 변경 이력으로부터 변경의 근원 정보를 추출하는 것도 가능하게 되었다. 또한 특정 산출물이 변경될 때 연관된 산출물에 변경이 파급되는 상황을 한 눈에 확인할 수 있도록 하는 시각적인 대시보드도 제공하고 있다.

참고문헌(Reference)

[1] K. Mohan, P. Xu, L. Cao, and B. Ramesh “Improving change management in software development: Integrating traceability and software configuration management,” Decision Support Systems, Vol. 45, Issue 4, pp. 922-936, 2008. https://doi.org/10.1016/j.dss.2008.03.003

[2] P. Mader and A. Egyed, “Do developers benefit from requirements traceability when evolving and maintaining a software system?” Empirical Software Engineering, Vol. 20, pp. 413-441, 2015. https://doi.org/10.1007/s10664-014-9314-z

[3] E. Y. Byun, et. al., “Requirement traceability matrix based on closed architecture mechanism,” Proc. of KIPS 2017 Fall Conference, Vol. 24, No. 2, pp. 631-634, 2017. https://doi.org/10.3745/PKIPS.y2017m11a.631

[4] J. Y. Kim, et. al., “A study of requirement change management and traceability effect using traceability table,” The KIPS Transactions: Part D, Vol. 17, No. 4, pp.271-282, 2010. https://doi.org/10.3745/KIPSTD.2010.17D.4.271

[5] J. C. Huang, C. K. Chang, and M. Christensen, “Event-based traceability for managing evolutionary change,” IEEE Transactions on Software Engineering, Vol. 29, No. 9, pp. 796-810, 2003. http://dx.doi.org/10.1109/TSE.2003.1232285

[6] M. Rahimi, and J. C. Huang, “Evolving software trace links between requirements and source code,” Empirical Software Engineering, Vol. 23, pp. 2198-2231, 2018. https://doi.org/10.1007/s10664-017-9561-x

[7] D. Baek, B. Lee, and J. W. Lee, “Content-based configuration management system for software research and development document artifacts,” KSII Transactions on Internet and Information Systems, Vol. 10, No. 3, pp. 1404-1415, 2016. http://dx.doi.org/10.3837/tiis.2016.03.027

[8] IBM Rational DOORS V9.6.0 Document, https://www.ibm.com/support/knowledgecenter/ko/SSYQBZ_9.6.0/com.ibm.doors.requirements.doc/topics/c_welcome.html

[9] Jama Software White Paper, “Better Product Development: Five Tips for Traceability,” https://resources.jamasoftware.com/whitepaper/better-product-development-five-tips-for-traceability

◐ 저 자 소 개 ◑

송 유 리(YooRi Song)
2016년 충남대학교 컴퓨터공학과(공학사)
2019년 충남대학교 대학원 컴퓨터공학과(공학석사)
관심분야 : 소프트웨어 공학, 소프트웨어 유지보수
E-mail : yrsong92@gmail.com

김 현 수(Hyeon Soo Kim)
1988년 서울대학교 계산통계학과(이학사)
1991년 한국과학기술원 전산학과(공학석사)
1995년 한국과학기술원 전산학과(공학박사)
1995년~1995년 한국전자통신연구원 Post Doc.
1996년~2001년 금오공과대학교 컴퓨터공학과 조교수
2001년~현재 충남대학교 컴퓨터융합학부 교수
관심분야 : 소프트웨어 공학, 소프트웨어 아키텍처, 소프트웨어 테스팅, 소프트웨어 유지보수
E-mail : hskim401@cnu.ac.kr