소프트웨어 프로덕트 라인에서 체계적인 요구사항 변경 관리 방법

(An Approach to Managing Requirements Change Systematically in Software Product Lines)

박 지 현 † 문 미 경 †† 염 근 혁 †††
(Jihyeon Park) (Mikyeong Moon) (Keunhyuk Yeom)

요 약 현재 소프트웨어 개발 공정이 복잡해짐에 따라 개발하는 소프트웨어의 요구사항 역시 복잡해지고 있으며 요구사항 관리에 많은 노력이 소요되고 있다. 그러나 소프트웨어의 요구사항을 처음부터 모두 정의하는 것은 사실상 불가능하며, 개발이 진행되면서 환경은 변하기 마련이다. 또한 요구사항 변경은 개발보다 많은 비용이 소요되므로 체계적인 변경 관리를 통해 변경에 민첩하게 대응하고 관리되어야 한다.

본 논문에서는 소프트웨어 프로덕트 라인에서 요구사항 변경 관리 프로세스를 기반으로 체계적인 요구사항 변경 관리 방법을 제시한다. 소프트웨어 프로덕트 라인(software product lines)에서 어플리케이션의 요구사항은 도메인 요구사항과 밀접하게 관련되어 있으므로, 어플리케이션 공학(application engineering) 단계에서 요구사항 변경이 발생했을 경우 도메인 공학(domain engineering) 단계를 이용하여 변경을 분석하고 관리하는 방법을 제시한다. 이러한 일관된 변경 관리를 통해 변경 범위를 분석하고, 변경 대처방안을 제시함으로써 변경 처리 결정을 내리는데 도움을 주며 한번 요청된 변경이 완벽하게 처리되도록 도와주어 같은 변경의 반복 요청을 막을 수 있기 때문에 잠재된 변경 비용을 절약할 수 있다.

키워드 : 요구사항 변경 관리, 요구사항 변경 관리 프로세스, 소프트웨어 프로덕트 라인

Abstract As the software development process becomes complicated, software requirements become complicated, too. Many efforts are needed in requirements management. It is impossible to define all requirements of software at first, and the development environment changes as project is gone. As the cost of requirements change management is much more than development cost, the changes should be controled immediately through systematic change management.

In this paper, I suggest a method to manage requirements change systematically based on the change management process in software product lines. The requirements change at the application engineering process is analyzed and managed using the domain engineering process because the application requirements are customized from the domain requirements in software product lines. Such the consistent change management helps to make decisions about changes by change impact analysis and alternative solution design. Through this method, the potential change costs can be saved because same change requests are not repeated by controlling the change requests completely.

Key words : Requirements Change Management, Requirements Change Management Process, Software Product Lines

1. 서 론

소프트웨어의 요구사항을 처음부터 모두 정의하는 것은 사실상 불가능하며, 개발이 진행되면서 환경은 변하기 마련이다[1]. 또한 유지보수에 대한 필요성이 높아지면서 개발비용에 비해 많은 요구사항 변경 비용이 수반되게 되었다[2]. 이는 좀더 나은 기능의 소프트웨어와 새로운 소프트웨어를 요구하는 시장의 요구가 점차 짧아지고 있기 때문이다. 즉, 소프트웨어의 지속적인 진화

· 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업의 연구결과로 수행되었음
† 정 회 원 : 국방과학연구소 연구원
witch03@hanmail.net
†† 정 회 원 : 부산대학교 컴퓨터및정보통신연구소 교수
mkmoon@pusan.ac.kr
††† 종신회원 : 부산대학교 컴퓨터공학과 교수
yeom@pusan.ac.kr
논문접수 : 2005년 1월 31일
심사완료 : 2005년 9월 14일

로 인해 다양한 요구사항 변경이 발생하게 되었고, 이를 관리하는 체계적인 절차의 부족으로 예상치 못한 시스템 동작의 문제가 발생하게 되었다.

기존의 변경 관리 방법은 요구사항 변경 발생 시 변경을 추적할 수 있도록 산출물들 간의 추적성 연구에 초점을 두고 있어 실제 변경 처리에 대한 연구는 부족하다[3]. 일관성 있는 변경 관리는 변경에 민첩하게 대응하고 같은 변경이 되풀이 되지 않도록 관리해 시간 낭비를 막을 수 있다. 이때 변경을 처리함에 있어 변경이 영향을 미치는 최상위 추상적 수준부터 시작해서 관련된 시스템 컴포넌트로 변경의 영향을 확장해 나가야 한다[1]. 그래서 추상화된 변경 요청은 개발자가 다룰 수 있도록 세부 수준으로 재정의 되어야 한다. 또한 하나의 요구사항에 대한 변경은 다른 요구사항에 영향을 미치므로 변경 관리는 변경 범위 분석을 통해 변경의 영향을 파악하고, 적절한 시기에 변경 결정을 내리도록 도와주어야 한다.

이러한 일관성 있는 변경 관리를 위해 기존에 여러 변경 관리 프로세스[1,3,4]가 제시되었다. 그러나 기존의 변경 관리 프로세스는 변경 범위 분석에 초점을 두고 있어 추상화된 변경 요청을 세부 수준으로 재정의 하고, 변경 처리를 위한 컴포넌트 수준에서의 구체적인 대처 방안 제시에는 미흡하다.

'프로덕트 라인(product line)'이란 공통의 서비스를 가지는 프로덕트의 집합을 말하는 것이다[5]. 소프트웨어 프로덕트 라인에서 어플리케이션 공학 단계는 도메인 공학에서 개발된 핵심 자산을 재사용해서 소프트웨어를 개발한다. 즉, 어플리케이션의 요구사항은 도메인 요구사항과 밀접하게 관련되므로, 어플리케이션 요구사항 변경이 발생했을 경우 도메인 공학 단계의 산출물들을 이용하여 변경을 분석하고, 구체적인 대처방안을 제시할 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 컴포넌트 기반 소프트웨어 프로덕트 라인과 기존의 요구사항 변경 관리 프로세스를 소개하고, 3장에서는 소프트웨어 프로덕트 라인에서 일관성 있는 요구사항 변경 관리를 위해 요구사항 변경 관리 프로세스를 제시한다. 4장에서는 본 연구에서 제시한 방법론을 적용하여 요구사항 변경을 처리하는 사례 연구를 소개한다. 마지막으로 5장에서는 결론 및 향후 연구 과제를 제시한다.

2. 관련 연구

2.1 요구사항 변경 관리 프로세스

변경 관리는 요구사항 변경을 관리하기 위해 사용되어지는 절차(procedures)와 프로세스(process)에 관련이 있다[4]. 즉 변경 관리는 요구사항 변경을 처리하기 위해 사용되어지는 프로세스를 통해 수행되어 진다. 그래서 기존의 요구사항 관리에 대한 여러 연구는 일관성 있는 요구사항 변경 관리를 위해서 변경 관리 프로세스를 제시하고 있다.

[1]은 템플릿 형태로 변경 관리 프로세스를 제시한다. 변경 관리 프로세스 템플릿은 프로세스를 실행하기 전에 충족되어야 하는 조건인 시작 기준(entry criteria), 프로세스에 포함된 다양한 작업들(tasks), 작업이 제대로 완료되었는지 검증(verify)하는 단계들, 프로세스가 성공적으로 완료된 것을 알려주는 종료 조건(exit criteria)으로 구성되어 진다.

[1]에서는 변경의 영향 범위를 분석할 때 경험에 의한 체크 리스트와 템플릿을 이용해 가이드를 제공하고 있다. 그렇지만 변경 요청이 구체적으로 다른 요구사항에 영향을 미치는 범위를 제시하고 있지 못하고, 변경에 대한 컴포넌트 수준의 대처방안 제시에는 미흡하다. 또한 변경 요청은 요청하는 사람에 따라 그 크기가 다양하기 때문에 개발자가 다룰 수 있는 크기로 세분화 되어져야 하는데 이런 과정이 존재하지 않는다.

[4]에서 제시하는 변경 관리 프로세스는 그림 1과 같은 단계들로 구성되어 있다.

[4]에서 제시하는 변경 관리 프로세스는 변경 분석에 초점을 두고 있다. 변경 분석 시 추적 정보를 이용해 영향을 미치는 요구사항들을 분석하는 절차만 제시하고, 구체적으로 어떤 방법을 이용해서 영향 분석할 지는 제시하고 있지 못하다. 그리고 프로세스의 최종 산출물은 변경으로 인해 영향을 받는 요구사항으로, 실제 변경 처리를 지원해 주지 못하고 있다. 즉, 요구사항 변경은 이후 설계와 구현 변경을 수반하기 때문에 변경 관리를 수행하기 위해서는 요구사항 변경이 설계 변경과 구현 변경에 어떻게 추적되어 지는지 연결을 지원해 주어야 한다.

[3,6,7]에서 제시하는 변경 관리 프로세스는 그림 2의 단계들로 구성되어 있다.

[3,6,7]의 변경 관리 프로세스는 변경의 영향 분석을 통해 하나의 기능적 요구사항의 변경이 다른 기능적 요구사항에 미치는 영향 범위를 분석하고 이를 그래프를 이용해 도식화하고 있다. 그러나 하나의 기능적 요구사항 변경에 대한 영향 분석만을 고려하고 있으므로 여러 개의 기능적 요구사항으로 구성될 수 있는 변경 요청을 모두 포함하기에는 부족하다. 또한 변경 분석에 초점을 두고 있어 변경 처리 시에 활용할 수 있는 변경 대처방안을 제시하는 부분이 부족하다.

앞서 살펴본 관련 연구에 의하면 변경 요청을 받아들이는 단계에서 추상화된 변경 요청을 개발자가 다루는 상세화된 수준으로 재정의가 필요하며, 하나의 기능적 요구사항이 미치는 영향의 범위가 아니라 여러 개의 기능적 요구사항으로 구성된 변경 요청이 다른 요구사항

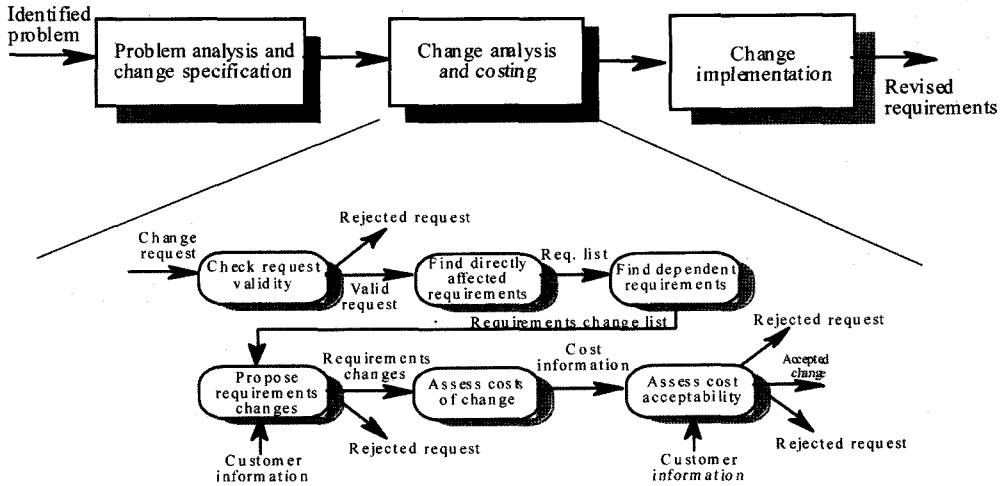


그림 1 변경 관리 프로세스의 단계들

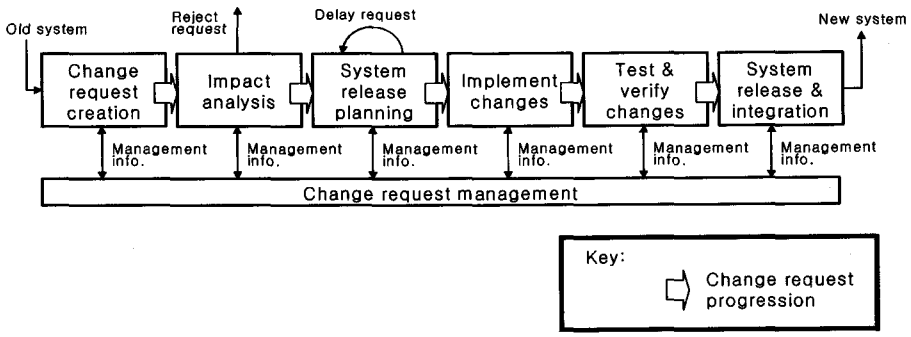


그림 2 구조화된 변경 관리 프로세스

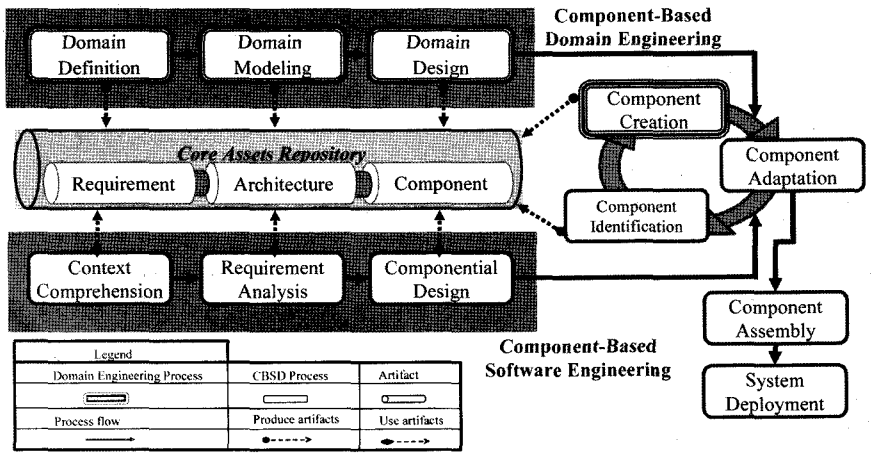


그림 3 컴포넌트 기반 소프트웨어 프로덕트 라인 프로세스[9]

에 미치는 영향의 범위가 분석되어야 한다. 그리고 변경 처리 시에 지침을 제공해 줄 수 있는 컴포넌트 수준에서의 구체적인 변경 대처방안이 제시될 필요가 있다.

2.2 소프트웨어 프로덕트 라인

컴포넌트 기반 소프트웨어 프로덕트 라인 프로세스 그림 3과 같다.

상위에 위치한 프로세스는 소프트웨어 개발자들이 새로운 시스템을 개발하는 데 재사용 할 수 있는 핵심 자산들을 식별, 개발, 보급하는 도메인 공학 프로세스이다[8]. 하위에 위치한 프로세스는 고객의 요구사항을 분석하고 도메인에서 이미 개발된 컴포넌트들을 조립하여 어플리케이션을 개발하는 활동인 어플리케이션 공학 프로세스이다. 이 과정에서는 도메인 공학 과정에서 산출된 핵심 자산들을 이용하여 프로덕트 라인 내에서 새로운 프로덕트를 만들어 내고, 프로덕트 라인의 자산에 의해 충족되지 않는 요구사항들은 도메인 공학 프로세스에 피드백(feedback)하여 프로덕트 라인에 결합되도록 한다.

컴포넌트 기반 도메인 공학 프로세스와 어플리케이션 공학 프로세스는 독립된 생명주기를 가지고 있지만, 컴포넌트 생성, 추출, 적응과 같은 마지막 단계에 이르게 되면 서로 자연스럽게 결합되어 하나의 프로세스로 융합되면서 컴포넌트 기반 소프트웨어 프로덕트 라인 프로세스를 완성하게 된다.

소프트웨어 시스템 세계에서 '도메인(domain)'이란 특정 분야에서 공통의 기능을 가지는 일련의 시스템의 집합을 나타낸다[10]. 도메인 요구사항(domain requirement)[11]은 프로덕트 라인 상에서 개발되는 시스템들의 공통적인 골격에 대한 요구사항들과 함께 시스템마다 상이하게 나타나는 부분들을 일정수준 추상화시킨 형태를 취한다. 그러므로 어플리케이션의 요구사항은 도메인 요구사항과 밀접하게 연관이 되어 있고, 요구사항 변경 역시 도메인 공학 과정에서 산출된 핵심 자산을 이용하여 처리 가능하다.

3. 요구사항 변경 관리 방법

3.1 요구사항 변경 관리

소프트웨어 프로덕트 라인에서 요구사항 변경은 도메인 요구사항 변경과 어플리케이션 요구사항 변경으로 구분되어 진다. 먼저, 도메인 요구사항 변경은 도메인 재분석 시에 자연스럽게 피드백 되어 도메인 공학 프로세스에 들어간다. 즉, 여러 번의 도메인 재분석을 통해 도메인 요구사항의 크기가 변하게 되며, 도메인 요구사항의 속성이 변화될 수 있다.

본 논문에서 다루고 있는 것은 컴포넌트 기반의 소프트웨어 프로덕트 라인에서 어플리케이션 요구사항의 변경 관리이다. 그림 4는 어플리케이션 요구사항 변경이 발생할 때 도메인 공학 프로세스를 이용해서 변경 처리를 하는 과정을 보여준다. 도메인 공학 프로세스를 진행하는 동안 다양한 산출물들 간의 추적성이 연결되어 개발되며, 이를 변경 관리에서 이용 가능하다. 어플리케이션 개발 시 발생할 수 있는 요구사항 변경들이 도메인 요구사항 분석 시 이미 예상되어 계획되어 있을 확률이 높다. 왜냐하면, 도메인 내에서 개발되는 시스템이 가질 수 있는 변경 사항들은 또한 도메인 내에 있기 때문이다[11].

따라서 어플리케이션 요구사항에서 발생하는 변경 요청은 어플리케이션에 곧바로 적용시키는 것이 아니라, 다시 역방향(backward)으로 추적하여 도메인 요구사항을 보게 된다. 도메인 요구사항에서는 이미 계획하고 있었던 변경 항목들 중에 변경 요청이 들어있게 된다. 그러면 도메인 요구사항으로부터 개발된 도메인 산출물들을 전방향으로(forward) 추적하여 미리 계획된 변경 구

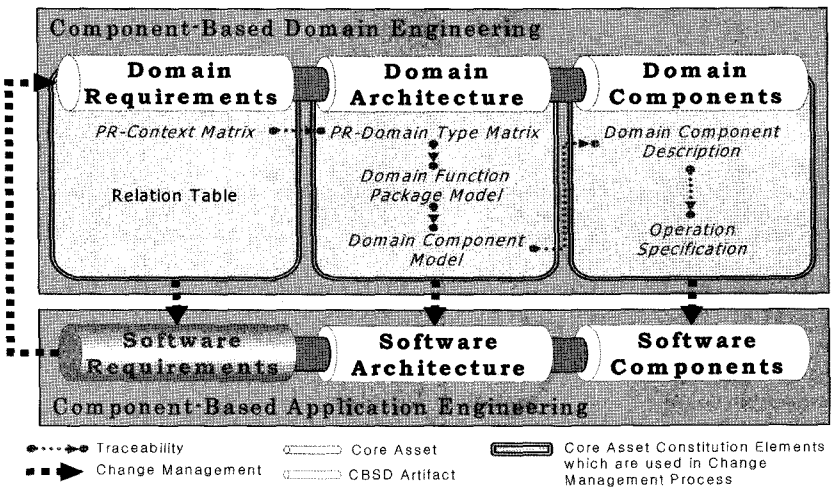


그림 4 소프트웨어 프로덕트 라인에서 요구사항 변경 관리

조를 확인할 수 있게 된다. 그래서 소프트웨어 프로덕트 라인 문맥에서 개발되는 어플리케이션은 요구사항 추적 가능성을 보장받을 수 있으며, 그로 인해 변경 관리의 수행을 용이하게 해 준다.

그림 4에서 어플리케이션 요구사항 변경 요청은 도메인 공학 산출물인 PR-Context Matrix에서 도메인 요구사항으로 추적되고, Relation Table을 이용해 변경 범위를 분석한다. 그리고 도메인 요구사항은 도메인 기능 패키지로, 도메인 기능 패키지는 도메인 컴포넌트로 추적된다. 따라서 변경 요청에 대한 도메인 컴포넌트 기술서(domain component description)와 오퍼레이션 명세서(operation specification)를 변경 처리 팀에게 전달하게 되므로, 어플리케이션에서 변경 요청을 수용하기 위해 분석, 설계, 구현 단계를 모두 거치는 시간과 비용을 줄일 수 있다.

3.2 요구사항 변경 관리 프로세스

본 논문에서는 일관된 변경 관리를 위해 다음 그림 5의 요구사항 변경 관리 프로세스를 제시한다.

프로세스는 변경 요청 처리, 변경 분석, 변경 대처방안 설계, 변경 처리 단계의 네 단계로 구성되어 있고, 각 단계는 내부에 세부 활동(activity)을 가지고 있다. 그리고 각 단계는 변경 요청이 모두 처리 되거나 거절될 때까지 반복 수행되어 진다.

3.2.1 변경 요청 처리

변경 요청 처리 단계는 시스템과 관련된 여러 사람들(stakeholder)로부터 변경 요청을 입력 받아서 문제를 기술하고 변경 관리 정보 템플릿을 작성하며, 변경 분석을 위한 정보로 활용하기 위해 변경 요청을 재정의 하는 단계이다. 이 단계의 가장 중요한 목표는 입력으로 들어온 추상화된 변경 요청을 기능적 요구사항 단위로 상세화해서 재정의 하는 것이다.

(1) 변경 관리 정보 작성

처음으로 변경 요청을 입력 받아 변경 관리 정보 템플릿을 작성한다. 변경 관리 정보 템플릿에는 변경 관리 정보, 스케줄링(scheduling), 우선순위 정보가 포함되며, 변경 요청에 대한 문제 정보가 문제 기술서에 작성되어 지고, 변경 요청 수용 여부가 결정된다. 이때 작성된 변경 관리 정보 템플릿은 이후 단계의 중요한 입력으로 들어가며, 이 단계에서 기록되지 않은 나머지 정보는 이후 프로세스를 진행하면서 업데이트 되어 진다.

(2) 변경 재정의

변경 요청은 누가 변경 요청을 하는 가에 따라 크기(granularity)가 다르므로 이를 일정한 수준으로 상세화 시킬 필요가 있다. 즉, 추상화된 변경 요청은 개발자가 다루는 크기로 재정의가 필요하다. 일반적으로 변경 요청은 여러 개의 기능적인 요구사항으로 세분화 될 수 있다. 이를 위해 본 논문에서는 그림 6의 변경 트리를 제시한다.

먼저, 첫 번째 변경 결정(Change Decision 1)에서 변경 요청은 하드웨어나 다른 소프트웨어를 구입해서 해결될 수 있고, 그렇지 않은 경우에는 현재 소프트웨어에서 변경을 처리해야 한다. 하드웨어나 다른 소프트웨어를 사용해 변경 요청을 처리할 경우에는 이후 변경 관리 프로세스를 수행하지 않고 종료하게 된다.

첫 번째 변경 결정에서 현재 소프트웨어를 변경하기로 결정되어지면, 변경 요청이 기능적 변경 요청인지 비기능적 변경 요청인지를 나누어 변경 트리를 작성한다. 기능적인 변경 요청은 다양한 기능적 요구사항들로 처리될 수 있고, 비기능적인 변경 요청은 다시 내부에 비기능적인 요구사항으로 구성될 수 있다[12]. 이때 두 번째 변경 결정(Change Decision 2)으로, 변경 요청을 처리하기 위한 다양한 방법 중에서 어떤 기능적 요구사항, 비기능적 요구사항으로 변경 요청을 처리할 지가 선택되어 진다.

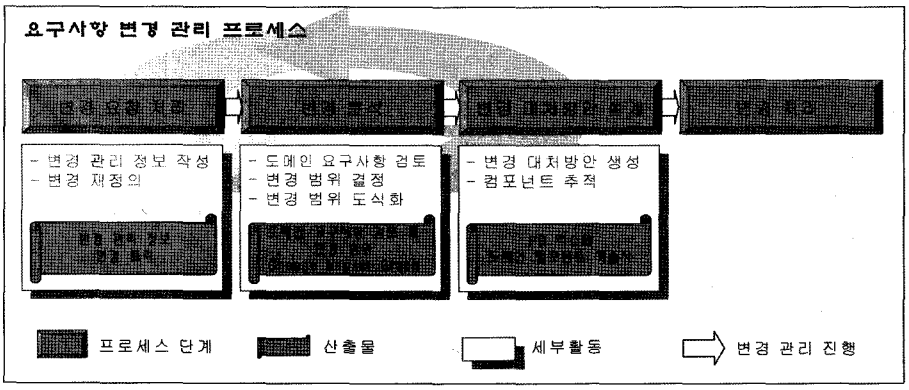


그림 5 요구사항 변경 관리 프로세스

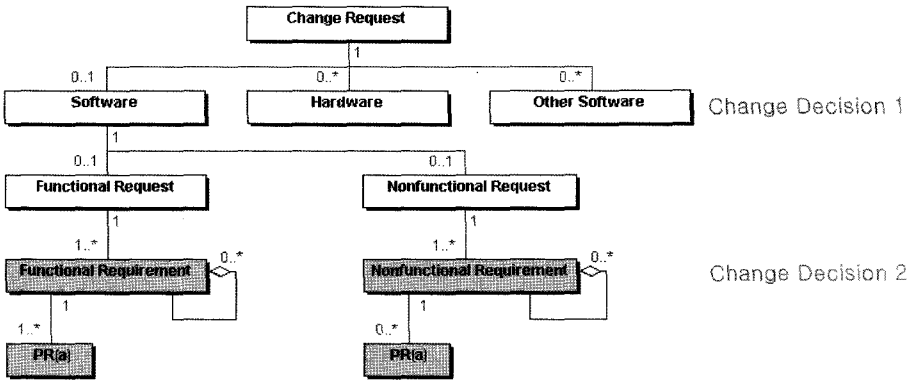


그림 6 변경 트리(Change Tree) 메타 모델

하나의 기능적, 비기능적 요구사항이 선택되어지면 이를 PR(Primitive Requirement)[11,9,13] 수준으로 상세화한다. PR은 도메인 요구사항에서 공통성과 가변성을 분석하기 적당한 수준으로 기능적 요구사항을 나눈 단위이다. 본 논문에서는 도메인 요구사항과 어플리케이션 요구사항을 분리하기 위해서 도메인 요구사항은 PR로 표현하고 어플리케이션 요구사항은 PR(a)로 표현한다.

기능적인 요구사항은 내부에 다시 기능적인 요구사항들로 구성될 수 있고, 최종적으로 PR(a) 단위로 쪼개어진다. 그리고 비기능적인 요구사항은 하나 이상의 기능적 요구사항과 관련되어 있으므로[14] 최종적으로 PR(a) 단위로 재정의 된다.

3.2.2 변경 분석

변경 영향 분석(change impact analysis)은 변경 요청을 구성하는 각 PR이 영향을 미치는 범위를 결정하는데 도움을 준다. 그리고 변경을 요청한 사용자나 고객에게 제안된 변경 요청의 복잡도를 설명해 주는 근거가 될 수 있다. 마지막으로, 변경을 요청한 각 PR뿐만 아니라 영향을 미치는 요구사항들을 함께 고려해서 같은 변경 요청이 반복해서 되풀이 되지 않도록 지원하므로 전체 유지보수 비용을 줄일 수 있다[15].

(1) 도메인 요구사항 검토

도메인 공학 산출물들을 이용해서 변경 관리를 하기 위해서는 먼저 앞의 변경 요청 처리 단계의 변경 트리를 이용해 찾아진 PR(a)들이 도메인 요구사항에 존재하는지 역추적해서 검토해야 한다. 도메인 요구사항이 다른 도메인 공학 산출물들과의 추적성을 연결하는 시작점이기 때문이다.

변경 요청을 구성하는 각 PR(a)에 대해서 일치하는 도메인 요구사항이 있는지 검토한다. 이때 각 PR을 추가, 삭제, 수정할 것인지 변경 타입을 결정한다. 삭제와 수정의 경우는 이미 어플리케이션 요구사항에 PR(a)가 존재할 때 가능하므로 당연히 도메인 요구사항에 존재한다. 그렇지만 추가의 경우는 현재 어플리케이션 요구사항에 PR(a)가 존재하지 않으므로 도메인 요구사항에 추가하고자 하는 PR이 존재할 수도 있고 존재하지 않을 수도 있다. 도메인 요구사항에 존재하지 않을 경우는 같은 도메인에 속하는 다른 시스템에는 추가하고자 하는 PR(a)가 존재하지 않는다는 의미이므로 아주 선택적(optional)인 PR이 되며, 도메인 재분석을 통해 도메인 공학 프로세스에 피드백 되어 입력된다. 이 경우에는 변경 요청이 도메인 공학 프로세스에서 발생하므로 본 논문에서는 다루지 않고, 추가하고자 하는 PR(a)가 도메인 요구사항에 존재하는 경우만을 고려한다. 도메인 요구사항 검토 활동 이후에는 변경 요청을 구성하는 기능적 요구사항 PR(a)는 일치하는 도메인 요구사항 PR로 대체되고, 둘 사이의 추적성이 유지된다.

다음으로 변경 요청을 구성하는 PR 사이에 순서를 고려해야 하는 경우를 찾아낸다. 어떤 PR을 먼저 구현하는 가에 따라 원래 원하는 결과와 일치하지 않을 수 있기 때문이다. 순서를 고려해야 하는 경우에는 '→'를 이용해서 표현하고, 고려하지 않아도 상관없는 경우에는 ','로 관계를 표현한다.

① *Change Request = {PR1, PR2→PR3}*

② *Change Request = {(PR1, PR2)→PR3}*

위의 예제 ①, ②에서 Change Request는 둘 다 PR1, PR2, PR3으로 구성되어 있지만 ①에서는 PR3을 수행하기 전에 반드시 PR2가 수행되어야 하지만 PR1은 순서를 고려하지 않아도 된다. ②에서는 PR1과 PR2 사이에는 순서를 고려하지 않아도 되지만 PR3을 수행하기 전에 반드시 PR1과 PR2가 먼저 수행되어야 함을 의미한다.

(2) 변경 범위 결정

하나의 요구사항 변경은 다른 요구사항들에 영향을 미치므로, 변경 요청을 구성하는 각 PR이 다른 요구사항에 미치는 변경 영향의 범위를 결정한다. 이때 도메인 공학 산출물인 Relation Table[16]을 이용해서 각 PR과 연관관계가 존재하는 PR들을 분석하게 된다. Relation Table에 나타나는 PR 사이에 존재하는 연관관계 중 본 논문에서 다루는 기능적 요구사항 사이의 관계는 표 1과 같다.

Relation Table을 이용해 그림 8의 알고리즘을 이용해 변경 범위를 분석할 경우 아래와 같이 두 번의 과정을 거치게 된다.

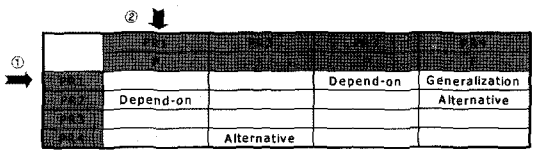


그림 7 Relation Table

```
n_CR = # of PRs in Change_Request
n_PR = # of PRs in Relation_Table

change_impact_analysis(){
   for(int i=0; i<n_CR; i++){
      for(int j=0; j<n_PR; j++){
         if(Relation_Table(Change_Request(i), j)=="Depend-on"){
            Change_Scope.add(Change_Request(i), j, 'd');
            depend_on(j);
         }
         if(Relation_Table(Change_Request(i), j)=="Generalization"){
            Change_Scope.add(Change_Request(i), j, 'g');
         }
         if(Relation_Table(Change_Request(i), j)=="Alternative"){
            Change_Scope.add(Change_Request(i), j, 'a');
         }
      }
      for(int k=0; k<n_PR; k++){
         if(Relation_Table(k, Change_Request(i))=="Depend-on"){
            Change_Scope.add(k, Change_Request(i), 'd');
         }
      }
   }
}

depend_on(int PRnum){
   for(int j=0; j<n_PR; j++){
      if(Relation_Table(Prnum, j)=="Depend-on"){
         Change_Scope.add(PRnum, j, 'd');
         depend_on(j);
      }
   }
}

(Note)
Change_Request(i) : number of Ist PR in Change_Request
Relation_Table(m, n) : relationship between PRm and PRj
relation = {a, d, g}
Change_Scope.add(p, q, relation) : change impact between PRp and PRq
```

그림 8 변경 분석 알고리즘

① Relation Table의 열(row)을 분석한다. 각 PR과 연관관계가 있는 모든 PR들을 분석한다. Depend-on 관계에 있는 PR에 대해서 다시 Relation Table의 열을 따라 Depend-on 관계에 있는 PR들을 찾아낸다. Generalization과 Alternative 관계는 기존의 PR을 대체하기 때문에 Depend-on 관계와 달리 변경되는 각 PR과 직접적으로 연결된 PR만 분석한다.

② Relation Table의 행(column)을 분석한다. 각 PR에 Depend-on 관계가 있는 PR들을 찾아낸다.

Change Request = {PR2→PR1}일 경우에 그림 7의 Relation Table을 참조하여

1. 먼저 PR2에 대해 ①을 수행하면 *PR2→PR1, PR2|PR4* 가 분석되고,
2. Depend-on 관계가 있는 PR1을 기준으로 다시 열을 따라 Depend-on 관계를 분석하면 *PR1→PR3*이 분석된다.
3. PR2에 대해서는 ②의 경우 Depend-on 관계가 있는 PR이 존재하지 않으므로 종료한다.
4. 두 번째 PR1에 대해서 ①을 수행하면 *PR1→PR3, PR1 ↳PR4*가 분석되고,
5. Depend-on 관계가 있는 PR3을 기준으로 다시 열을 분석하면서 Depend-on 관계를 찾게 되는데 PR3과 관련된 PR이 존재하지 않으므로,
6. ②를 수행한다. 행을 분석해 PR1에 Depend-on 관계가 있는 PR을 찾아내면 *PR2→PR1*의 관계를 분석해 낼 수 있다.

(3) 변경 범위 도식화

여러 PR로 구성된 변경 요청은 영향을 미치는 다른 PR과 많은 연관관계가 존재하므로, 이를 효율적으로 제시하기 위해 그래프로 도식화할 필요가 있다. 그래서 그림 9의 CIG: Change Impact Graph를 작성한다. CIG=(V,E,I)의 V는 변경 범위에 포함되는 PR의 집합, E는 변경 범위에 포함되는 PR 사이의 연결, I는 변경 범위에 포함되는 PR 사이의 연관관계로 구성된다. 그리고 CIG에서 변경 요청을 구성하는 V는 구분하기 위해 음영 처리 되어 있으며, 각 E에는 어떤 연관관계가 존재하는지 표시된다. d는 Depend-on 관계, a는 Alternative 관계, g는 Generalization 관계를 의미한다.

① 변경 요청을 구성하는 각 PR에 대해 Change Impact Graph를 작성한다.

표 1 도메인 요구사항들 간의 관계[13]

기능적 요구사항 vs. 기능적 요구사항	
Depend-on (PR_1 → PR_2)	하나의 PR이 다른 PR을 '요구(require)'하거나 '필요(need)'로 하는 경우이다.
Generalization (PR_1 ↳ PR_2)	PR이 하나 이상의 방법으로 인스턴스 될 수 있는 경우이다.
Alternative (PR_1 | PR_2)	하나의 PR이 다른 PR들과 대체될 수 있는 경우이다.

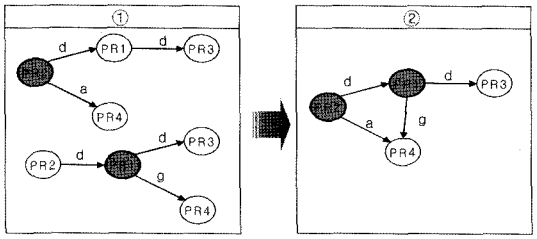


그림 9 Change Impact Graph

② 변경 순서를 고려해서 변경 요청을 구성하는 각 PR들의 CIG를 결합하고 중복을 제거한다.

3.2.3 변경 대처방안 설계

요구사항 추적가능성이란 요구사항 일생(life)을 앞(forward), 뒤(backward) 방향으로 기술하고 따라갈 수 있는 능력을 말한다[17]. 요구사항 추적가능성은 변화의 영향과 결과를 분석하기 위한 기초 자료라는 점에서 요구사항 관리의 핵심이 된다. 도메인 공학 프로세스에서 도메인 요구사항은 기능 패키지(package), 도메인 컴포넌트, 도메인 컴포넌트의 인터페이스가 가지는 오퍼레이션으로 추적 관계가 연결되어 있다. 이러한 추적 정보를 바탕으로 변경 대처방안이 설계되어 진다.

(1) 변경 대처방안 생성

변경 분석 단계에서 분석된 변경 범위에 포함되는 PR들에 대해서 표 2의 PR의 목록을 작성한다.

먼저 변경 범위에 포함된 각 PR이 현재 시스템에 존재하는지 여부를 검토하고, 도메인에 속한 전체 시스템에서 나타나는 빈도수(ratio) 정보를 함께 제공해 준다. 여기서 세 번째 변경 결정(Change Decision 3)이 내려지게 된다. 변경 범위에 속하는 PR들을 모두 변경에 포함시킬 것인지 PR 범위의 결정이 필요하다.

이때, 현재 시스템에 존재하지 않을 경우 PR을 추가할 것인지 결정은 빈도수 정보를 이용해서 결정하도록 도와준다. 현재 시스템에 존재하더라도 변경 요청의 변경 타입과 Change Impact Graph의 바인딩 정보에 따라서 수정할 것인지, 삭제할 것인지, 변경하지 않을 것인지를 결정하도록 도와준다.

변경 결정을 내리고 나면 필수적으로 고려해야 하는 PR들을 선택해서 표 2의 PR 목록의 마지막 항목인 PR 범위를 작성한다. 필수적으로 고려해야 하는 PR이 Change Impact Graph에서 alternative 관계에 있을 경우 현재 시스템에 존재하는 PR을 대체할 것인지 아니면 덧붙여서 추가할 것인지가 같이 결정되어야 한다. 그리고 PR 범위에 속한 모든 PR들이 모두 변경 처리가 될 때까지 변경 요청 처리 단계부터 반복적으로(iteratively) 수행한다.

(2) 컴포넌트 추적

도메인 요구사항은 도메인에 속하는 어플리케이션들의 요구사항의 집합이므로 어플리케이션 요구사항 변경 요청은 일치하는 도메인 요구사항으로 추적 가능하고, 따라서 기능 패키지, 도메인 컴포넌트, 도메인 컴포넌트의 인터페이스에서 오퍼레이션으로 추적되어 진다.

다음은 변경을 대처하기 위해 컴포넌트를 추적하는 과정이다.

① PR 범위에 속하는 각 PR에 연결되는 모든 기능 패키지들을 추출한다.

② 기능 패키지들 중에서 독립 컴포넌트로 추적되는 경우를 찾는다. 각 기능 패키지에 대해서, 기능 패키지를 구성하는 모든 PR들이 PR 범위에 속하고, PR들이 같은 변경 타입인 경우 독립 컴포넌트로 추적되어질 수 있다.

③ 변경 순서에 따라서 PR 범위에 속하는 각 PR에 대해 기능 패키지에서 도메인 컴포넌트를 추적한다. ②에서 독립 컴포넌트로 추적되어 지는 경우 도메인 컴포넌트 기술서 형태로 독립 컴포넌트를 추가하거나 삭제하도록 변경 대처방안이 제공된다.

④ 독립 컴포넌트로 추적되지 않는 경우 도메인 컴포넌트의 인터페이스를 추적한다. PR이 인터페이스가 가지는 모든 오퍼레이션과 연결되는 경우 인터페이스 명세서 형태로 인터페이스를 추가하거나 삭제하도록 변경 대처방안이 제공된다.

⑤ 인터페이스로 추적되지 않는 경우 인터페이스가 가지는 오퍼레이션을 추적한다. 연결되는 오퍼레이션 명세서 형태로 오퍼레이션을 추가, 삭제, 수정하도록 변경 대처방안이 제공된다.

표 2 PR 목록

변경 요청 (Change Request)		CRn {PR2 → PR1 }			
변경 범위 (Change Scope)		변경 타입 (Change Type)	시스템 존재유무	빈도수 (Ratio)	PR 범위 (PR Scope)
PR1					
PR2					
PR3					
PRn					

결과적으로 추적되어지는 도메인 컴포넌트 기술서와 인터페이스 명세서, 오퍼레이션 명세서가 변경 처리 팀에게 전달되어 진다.

3.2.4 변경 처리

마지막 변경 처리 단계는 앞의 변경 요청 처리, 변경 분석, 변경 대처방안 설계 단계를 통해 생성된 산출물들을 변경 처리 팀에게 전달하고, 변경 처리 팀에서 실제 변경되는 부분을 구현한다.

변경 처리 단계에서는 도메인 공학 산출물 중에서 도메인 컴포넌트 기술서와 오퍼레이션 명세서를 중요한 입력으로 받아들인다. 변경처리 팀은 문서형태로 제공되는 컴포넌트를 실제 구현하고, 기존의 컴포넌트를 오퍼레이션 명세서를 이용해 수정 가능하다. 실제 변경이 구현되어진 소프트웨어는 시험 및 검증 단계를 거쳐 고객에게 배포되어진다. 마지막으로 변경 관리 정보 템플릿을 완성하고 프로세스를 종료한다.

4. 사례 연구

본 논문에서 제안한 요구사항 변경 관리 프로세스를 e-TS(e-Travel Systems) 도메인의 여행 포털 시스템에 적용하였다. 이를 위해서 선행적으로 e-TS 도메인에 대한 컴포넌트 기반 도메인 공학 산출물들이 생성되어야 하며, 이 산출물들을 이용해서 여행 포털 시스템이 개발되어야 한다. 이후 여행 포털 시스템에서 요구사항 변경 요청이 발생 시 본 연구의 요구사항 변경 관리 프로세스를 따라 변경 관리가 수행되어짐을 사례 연구를 통해 밝힌다.

4.1 도메인 공학 프로세스

e-TS는 전자 여행 예약 서비스 시스템으로 웹 기반으로 고객에게 여행 예약 및 자신의 여행 예약 정보를 조회, 수정, 삭제할 수 있는 기본 기능을 제공한다. 선택 기능으로 맞춤 여행 상품을 예약할 수 있다. 그리고 여행 예약을 위해 외부에 항공사, 호텔 등과 연계하여 서비스를 제공한다.

e-TS 도메인에서 그림 10의 왼쪽의 PR-Context Matrix를 통해 31개의 도메인 요구사항 PR이 수집되었고, e-TS 도메인에 속하는 여러 시스템들에서 PR의 존재여부가 'O', 'X'로 표현되었으며 각 PR이 시스템들에 나타나는 빈도수인 Ratio가 계산되었다. Ratio를 바탕으로 PR의 공통성과 선택성의 속성이 결정되어진다. 도메인 분석 결과 도메인 요구사항 사이의 연관관계가 오른쪽 그림인 *Relation Table*을 이용해 추출되어 졌다.

그림 11의 왼쪽 도메인 기능 패키지 모델은 도메인의 구조를 모델링하기 위해 PR과 도메인 타입의 관계로부터 도메인 기능 패키지를 추출하고 그들 사이의 의존관계를 추출하였다. 그림 11의 오른쪽 그림은 e-TS 도메인에 대해 도메인 컴포넌트 모델을 개발한 모습을 보여준다. 도메인 컴포넌트는 기능 패키지 하나에 하나씩 대응된다. 이 모델에서는 도메인 컴포넌트가 제공하는 인터페이스를 보여주고, 그 도메인 컴포넌트가 사용하는 인터페이스의 관계도 보여준다. 도메인 아키텍처 상에서 도메인 컴포넌트의 변경은 해당 도메인의 모델링 요소와 그와 연관된 바인딩과, 내부적으로 변경되어야 하는 오퍼레이션을 고려해야 한다.

각 도메인 컴포넌트에서 구현될 타입들이나 구성하는 인터페이스 정보가 표 3 도메인 컴포넌트 기술서에 작

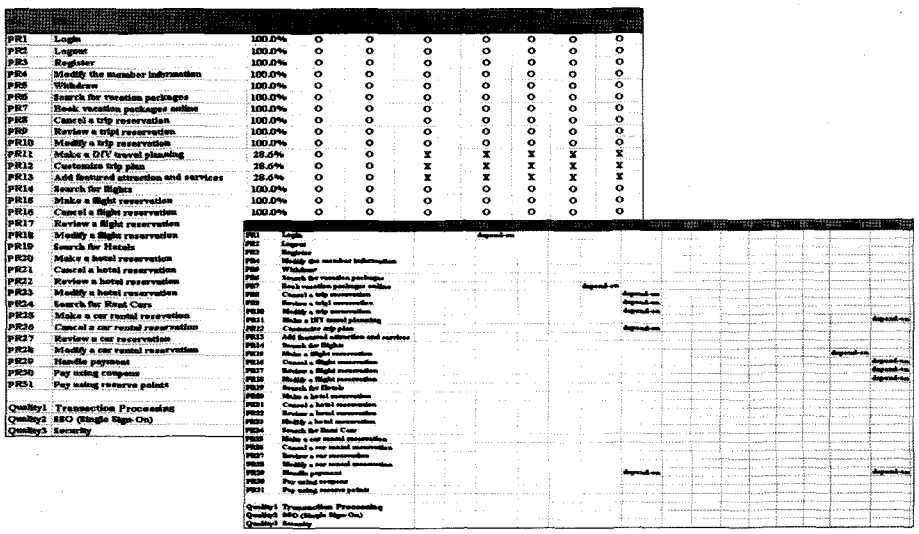


그림 10 e-TS PR-Context Matrix와 Relation Table

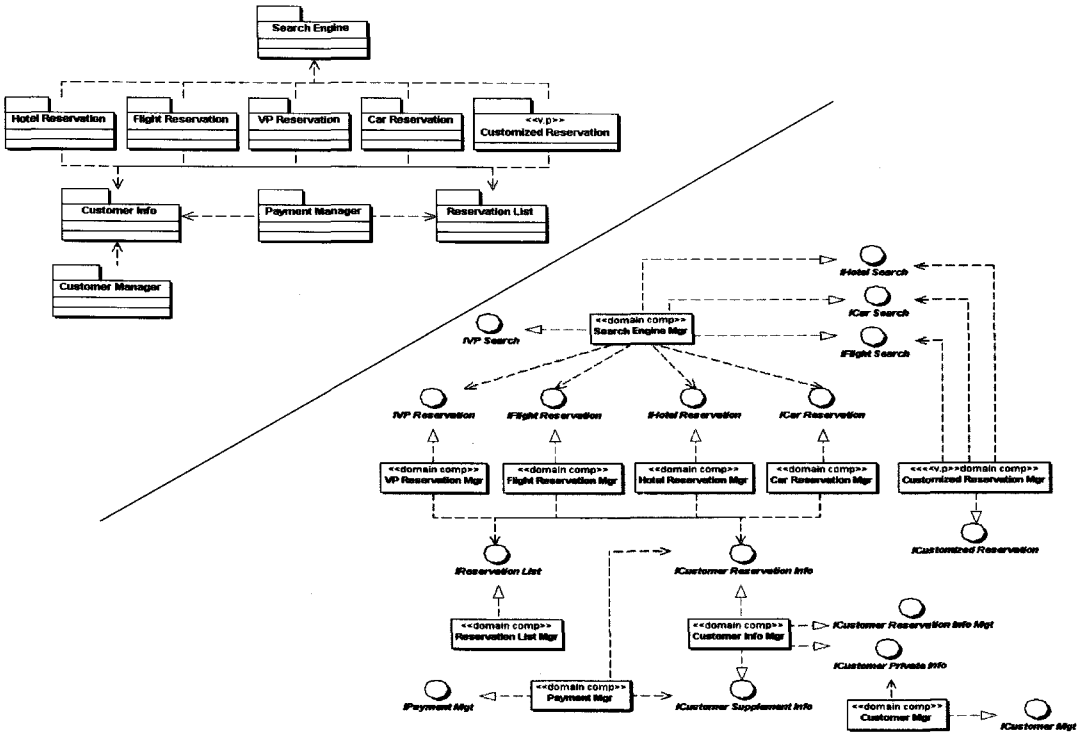
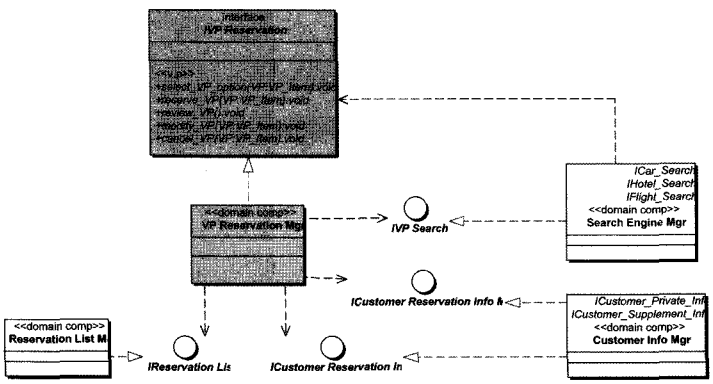


그림 11 e-TS 도메인 기능 패키지 모델과 도메인 컴포넌트 모델

표 3 VP_Reservation 도메인 컴포넌트 기술서

name	***VP_Reservation_DomainComponent***
Description	This component manages the reservation of Vacation Packages i.e. VP. This component provides reservation, review, modification, and cancellation of VP

Provided interface	<<interface>>IVP_Reservation
	<<v.p>> void select_VP_option(VP_Item VP);
	void reserve_VP(VP_Item VP);
	void review_VP();
	void modify_VP(VP_Item VP);
	void cancel_VP(VP_Item VP);

Required interface	<<interface>>IVP_Search:Search_Engine_Mgr <<interface>>ICustomer_Reservation_Info_Mgt:Customer_Info_Mgr <<interface>>ICustomer_Reservation_Info:Customer_Info_Mgr <<interface>>IReservation_List:Reservation_List_Mgr
Domain component specification model	

성되었다. 표 3의 도메인 컴포넌트 기술서를 통해서 VP_Reservation 도메인 컴포넌트에 대한 설명과 구체적인 주요 기능들을 알 수 있다. 그리고 도메인 컴포넌트가 제공하는 인터페이스의 기능을 기술하기 위해서 오퍼레이션 명세서와 각 오퍼레이션들이 어떻게 동작하는지는 도메인 오브젝트 상호작용 모델을 작성했다.

4.2 어플리케이션 공학 프로세스

소프트웨어 프로덕트 라인에서 도메인 공학 프로세스를 통해 개발된 산출물들은 어플리케이션 공학 프로세스에서 재사용되어진다. 4.1에서 e-TS 도메인 산출물들을 참조하여 웹 기반의 여행 포털 시스템을 개발하였다[7].

표 4는 개발되어진 여행 7포털 시스템의 요구사항으로 사용자와 은행, 예약과 관련해서 15개의 요구사항이 선택되어졌다. 그림 12는 표 4의 요구사항을 바탕으로

표 4 여행 포털 시스템 요구사항

구분	R_ID	요구 사항 명	수용 여부	우선 순위
사용자 관련	PR01	로그인	O	상
	PR02	로그아웃	O	상
	PR03	회원가입	O	상
	PR04	회원정보 수정	O	중
은행 관련	PR05	신용카드 정보 조회	O	중
	PR06	신용카드 이용 내역 조회	O	중
	PR07	신용카드 사용 등록	O	중
	PR08	신용카드 사용 해지	O	중
	PR09	신용카드 결제	O	상
	PR10	신용카드 결제 취소	O	중
예약 관련	PR11	여행 패키지 조회	O	중
	PR12	여행 예약	O	상
	PR13	여행 예약 수정	O	중
	PR14	여행 예약 취소	O	중
	PR15	여행 예약 조회	O	중

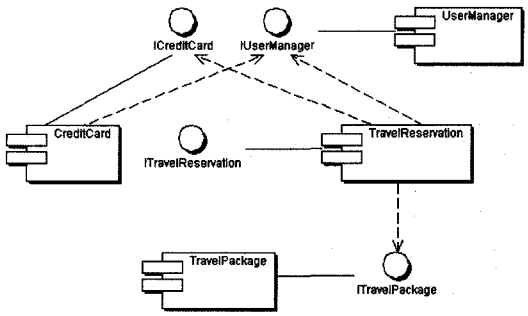


그림 12 여행 포털 시스템 컴포넌트 모델

생성된 컴포넌트 모델을 보여준다.

4.3 요구사항 변경 관리

여행 포털 시스템에 존재하지 않는 호텔 예약기능을 추가해달라는 변경 요청에 대해 본 논문에서 제시하는 프로세스를 적용한 결과이다.

4.3.1 변경 요청 처리

먼저 그림 13의 호텔 예약 기능을 추가해달라는 변경 요청에 대해 작성된 변경 관리 정보 템플릿과 변경트리를 이용해 PR(a) 단위의 "Search for hotels", "Make a hotel reservation", "Cancel a hotel reservation", "Review a hotel reservation", "Modify a hotel reservation"으로 재정의 했다.

4.3.2 변경 분석

변경 트리에서 재정의 된 변경 요청에 대해서 도메인 공학 산출물인 PR-Context Matrix를 참조해서 일치하는 PR19, PR20, PR21, PR22, PR23을 찾아 표 5의 도메인 요구사항 검토 표를 작성했다. 호텔 예약 기능은 여행 포털 시스템에 존재하지 않는 요구사항이므로 변경 타입은 '추가'가 되었다.

변경 요청 CR5는 5개의 PR로 구성되었으므로 이들의 변경 순서를 고려해야 한다. 호텔 예약을 하기 위해서는 호텔 조회가 선행되어야 하며, 호텔 예약이 되어야

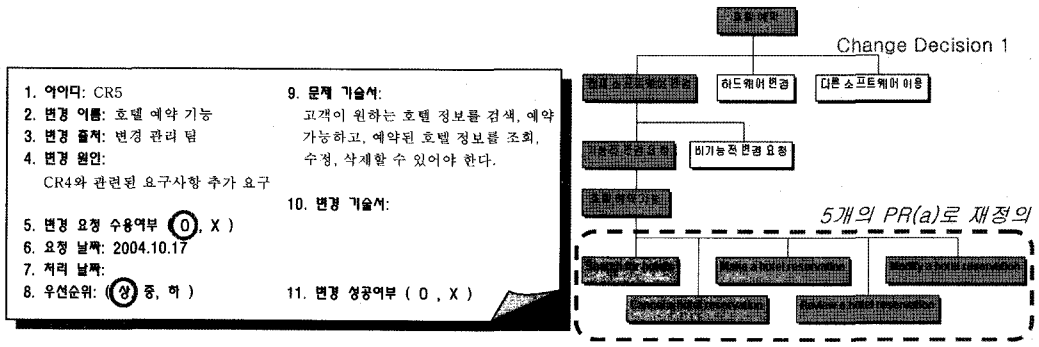


그림 13 변경 관리 정보 템플릿과 변경 트리

표 5 도메인 요구사항 검토 표

변경 요청 (Change Request)	PR(a) (Application Requirement)	변경 타입 (Change Type)	PR-도메인 요구사항 (Domain Requirement)
CR5		추가	PR19 Search for hotels
		추가	PR20 Make a hotel reservation
		추가	PR21 Cancel a hotel reservation
		추가	PR22 Review a hotel reservation
		추가	PR23 Modify a hotel reservation

지 호텔 예약 정보를 조회, 수정, 취소가 가능하므로 다음과 같은 변경 순서를 찾았다.

Change Request 5 = {PR19 → PR20 → (PR21, PR22, PR23)}

변경 순서를 따라서 각 PR에 대해 Relation Table에서 연관관계가 있는 PR들을 두 단계에 거쳐 분석한 결과인 변경 영향의 범위를 그래프를 이용해 표현하면 그림 14의 왼쪽 ①과 같다. 그림 14의 오른쪽 ②는 변경 요청이 여러 개의 PR들로 구성되어 있으므로 그래프를 결합하고, 중복을 제거한 모습이다.

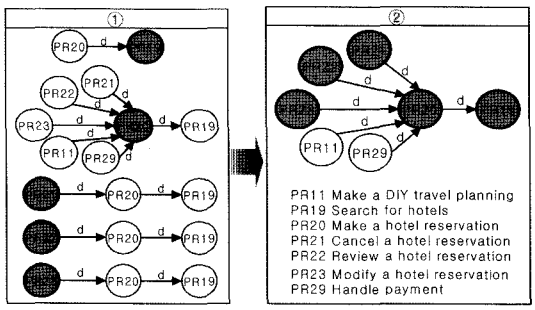


그림 14 Change Impact Graph

4.3.3 변경 대처방안 설계

변경 범위에 포함된 PR11, PR19, PR20, PR21, PR22, PR23, PR29에 대해서 여행 포털 시스템에 존재 여부와 PR-Context Matrix로부터 각 PR이 도메인에 포함된 시스템에 나타나는 빈도수인 Ratio 정보를 표 6의 PR 목록에 기록했다. PR11은 빈도수가 낮으므로 PR 범위에서 제외되며, PR29는 PR20에 Depend-on 관계가 있으므로 수정될 수 있다.

PR 범위에 속하는 PR에 대해 도메인 공학 산출물들의 추적 관계를 쫓아간 결과는 표 7과 같다. 최종적으로 변경 처리 팀에게 도메인 컴포넌트 기술서, 인터페이스 명세서, 오퍼레이션 명세서 제공되었다.

4.4 사례 연구 평가

본 논문의 방법론을 적용하여 사례 연구했던 e-TS 도메인의 여행 포털 시스템에서 요구사항 변경 요청에 대해 요구사항 변경 관리 프로세스를 적용했을 때, 기존의 변경 관리 프로세스를 적용했을 때와 비교하여 가지는 장점은 다음 표 8과 같다.

먼저, 추상화된 변경 요청을 변경 트리를 이용하여 PR 수준으로 상세화해서 재정의 했다. 따라서 여러 개의 기능적 요구사항으로 구성된 변경 요청이 변경 관리를 수행하는 단위로 사용되었다. 그리고 도메인 공학 산출물을 이용해 변경 요청이 다른 요구사항에 미치는 범위가 분석되어 그래프로 도식화 되었다. 변경 범위에 포함된 요구사항들은 추적 관계를 따라 다양한 도메인 공학 산출물로 추적되어 컴포넌트 수준의 변경 대처방안이 제시되었다.

5. 결론 및 향후 연구 과제

본 논문에서는 일관성 있는 변경 관리를 지원하기 위

표 6 PR 목록

변경 요청 (Change Request)		CR5 {PR19→PR20→(PR21, PR22, PR23)}			
변경 범위 (Change Scope)		변경 타입 (Change Type)	시스템 존재유무	빈도수 (Ratio)	PR 범위 (PR Scope)
PR11	Make a DIY travel planning		X	28.6%	
PR19	Search for hotels	추가	X	100%	O
PR20	Make a hotel reservation	추가	X	100%	O
PR21	Cancel a hotel reservation	추가	X	100%	O
PR22	Review a hotel reservation	추가	X	100%	O
PR23	Modify a hotel reservation	추가	X	100%	O
PR29	Handle payment	수정	O	100%	O

표 7 컴포넌트 추적

PR 범위	기능 패키지	컴포넌트	인터페이스	오퍼레이션
PR19	Search Engine	Search Engine Mgr	IHotel_Search	
PR20	Hotel Reservation	Hotel Reservation Mgr		
	Customer Info	Customer Info Mgr	ICustomer Reservation Info	insert_hotel()
	Reservation List	Reservation List Mgr	IReservation List	append_hotel()
PR21	Hotel Reservation	Hotel Reservation Mgr		
	Customer Info	Customer Info Mgr	ICustomer Reservation Info	delete_hotel()
	Reservation List	Reservation List Mgr	IReservation List	delete_hotel()
PR22	Hotel Reservation	Hotel Reservation Mgr		
	Customer Info	Customer Info Mgr	ICustomer Reservation Info	take_hotel_info()
PR23	Hotel Reservation	Hotel Reservation Mgr		
	Customer Info	Customer Info Mgr	ICustomer Reservation Info	update_hotel()
	Reservation List	Reservation List Mgr	IReservation List	update_hotel()
PR29	Payment manager	Payment Mgr	IPayment_Mgt	pay_by_credit()

표 8 다른 방법론과의 비교 검토

비교 내용	Karl Wiegers [1]	Gerald Kotonya & Ian Sommerville [4]	Simon Lock & Gerald Kotonya [3,6,7]	본 논문에서 제안된 방법
변경 관리 프로세스 제시	○	○	○	○
변경 요청 상세화 과정	×	△	×	○
변경 영향 범위 분석 절차 제공	○	○	○	○
변경 범위 결과 표현	×	×	○	○
변경 대처방안 제시	×	×	×	○
적용 가능성	○	○	○	○

○: 명확하게 제시함 △: 제시하고 있으나 미흡함 ×: 존재하지 않음

해 요구사항 변경 관리 프로세스를 제시하였다. 기존의 요구사항 변경 관리 프로세스는 일반적인 시스템의 요구사항 변경을 다루고 있어 컴포넌트 수준의 구체적인 대처방안을 제시해 주지 못했다. 이에 본 연구에서는 상세 수준의 변경 처리가 가능하도록 소프트웨어 프로덕트 라인에서 어플리케이션 요구사항 변경 관리로 그 범위를 제한했다. 요구사항 변경 관리 프로세스는 변경 요청 처리 단계에서 추상화된 변경 요청을 PR 단위로 재정의 했으며, 변경 분석 단계에서 하나의 변경 요청이 다른 요구사항에 미치는 영향의 범위를 분석하는 방법을 제공했다. 또한 변경 대처방안 설계 단계에서 도메인 공학 산출물들의 추적성을 이용하여 컴포넌트 수준의 변경 대처방안을 제시하였다.

사례 연구를 통해 살펴본 결과 요구사항 변경 관리 프로세스를 따라서 다양한 변경 요청을 처리할 수 있었고, 변경 처리 시간을 단축시킬 수 있었다.

향후 연구 과제로는 변경 범위를 분석할 때 기능적 요구사항들 사이의 관계뿐만 아니라 기능적 요구사항과 비기능적 요구사항 사이의 관계, 또 비기능적 요구사항들 사이의 관계를 함께 고려해서 연구를 확장할 필요가 있다.

참 고 문 헌

[1] Wiegers, K., *Software Requirements*, Microsoft Press, 2003.

[2] Andriole, S., *Managing Systems: Requirements, Methods, Tools and Cases*, McGraw-Hill, 1996.

[3] Lock, S., and Kotonya, G., "Requirement Level Change Management and Impact Analysis," Cooperative Systems Engineering Group, Technical Report Ref: CSEG/21/1998, October 1998.

[4] Kotonya, G., and Sommerville, I., *Requirements Engineering Process and Techniques*, John Wiley & Sons, 1998.

[5] Bosch, J., *Design and use of software architectures*, Addison Wesley, 2000.

[6] Lock, S., and Kotonya, G., "An Integrated Framework for Requirement Change Impact Analysis," Proceedings of the 4th Australian Conference on Requirements Engineering, Sydney, Australia, September 1999, pp.29-42.

[7] Lock, S., and Kotonya, G., "Abstract: An Integrated Framework for Requirement Change Impact Analysis," Requirenautics Quarterly: The Newsletter of the Requirements Engineering

Specialist Group of the British Computer Society, Issue 18, January 2000.
[8] Frank, S., "The Three "R's" of Mature System Development: Reuse, Reengineering, and Architecture," In The Fifth Systems Reengineering Technology Workshop, 1995.
[9] Moon, M., and Yeom, K., "Domain Design Method to Support Effective Reuse in Component-Based Software Development," Proceedings of the 1st ACIS International Conference on Software Engineering Research & Applications, San Francisco, USA, 2003, pp.149-154.
[10] Creps, D., Klinger, C., Levine, L., and Allemang, D., "Organization Domain Modeling(ODM) Guidebook Version 2.0," Software Technology for Adaptable, Reliable System(STARS), 1996.
[11] 문미경, 염근혁, "소프트웨어 프로덕트 라인에서 핵심 자산으로서 요구사항을 관리하는 방법", 한국정보과학회 논문지 : 소프트웨어 및 응용, vol.31, no.8, 2004년 8월, pp.1010-1026.
[12] Losavio, F., "Quality Models to Design Software Architecture," In Journal of Object Technology, vol.1, no.4, September-October 2002, pp.165-178, http://www.jot.fm/issues/issue_2002_09/article4.
[13] Moon, M., and Yeom, K., "An Approach to Develop Requirement as a Core Asset in Product Line," Bosch, J., and Krurger, C. (Eds.): ICSR 2004, LNCS 3107, July 2004, pp.23-34.
[14] Kotonya, G., and Sommerville, I., "Requirements Engineering with Viewpoints," BCS/IEE Software Engineering Journal, vol.11, no.1, January 1996, pp.5-18.
[15] Moreton, R., "A Process Model for Software maintenance," Software Change Impact Analysis, Bohner, S.; Arnold, R., Los Alamitos, IEEE Computer Society, 1996, pp.29-33.
[16] Park, J., Moon, M., and Yeom, K., "DREAM: Domain REquirement Asset Manager in Product Lines," International Symposium on Future Software Technology (ISFST2004), Xian, China, October 20-22, 2004.
[17] Gotel, O., and Finkelstein, A., "An Analysis of the Requirements Traceability Problem," 1st IEEE International Conference on Requirements Engineering (ICRE'94), Colorado Springs, April, 1994, pp.94-101.

박 지 현

1993년 2월 부산대학교 컴퓨터공학과(공학사). 2005년 2월 부산대학교 컴퓨터공학과(공학석사). 2005년 3월~현재 국방과학연구소 연구원. 관심분야는 소프트웨어 프로덕트 라인 공학, 지휘통제체계 개발 등임

문 미 경

1990년 2월 이화여자대학교 전자계산학과(학사). 1992년 2월 이화여자대학교 전자계산학과(석사). 2005년 2월 부산대학교 컴퓨터공학과(박사). 2005년 3월~2005년 8월 부산대학교 컴퓨터공학과(박사후 연구원). 2005년 9월~현재 부산대학교 컴퓨터 및 정보통신 연구소 교수. 관심분야는 소프트웨어 프로덕트 라인 공학, 적응형 소프트웨어 개발, FRID 기반 미들웨어 및 FRID 비즈니스 이벤트 프레임워크 개발 등임

염 근 혁

1985년 2월 서울대학교 계산통계학과(학사). 1992년 8월 Univ. of Florida 컴퓨터공학과(석사). 1995년 8월 Univ. of Florida 컴퓨터공학과(박사). 1985년 1월~1988년 2월 금성반도체 컴퓨터연구실 연구원. 1988년 3월~1990년 6월 금성사 정보기기연구소 주임연구원. 1995년 9월~1996년 8월 삼성SDS 정보기술연구소 책임연구원. 1996년 8월~현재 부산대학교 컴퓨터공학과 부교수. 부산대학교 컴퓨터 및 정보통신 연구소 연구원. 관심분야는 소프트웨어 재사용, 프로덕트라인 공학, 소프트웨어 아키텍처, 컴포넌트 기반 소프트웨어 개발, 적응형 소프트웨어 개발, RFID기반 미들웨어 등임