실무적 요구공학 공정

Practical Approach to Requirements Engineering Process

박수용, 황만수*, 박수진, 서성숙, 나호영
Soo-Yong Park, Soo-Jin Park, Sung-Sook Seo, Ho-Young Na, Man-Soo Hwang
서강대학교, 신흥대학*

ABSTRACT

시스템 개발환경이 대형화, 복잡화됨에 따라 개발 생명주기의 각 단계에서 발생하는 요건은 기술적, 관리적 측면에서 많은 영향을 받게 되었다. 즉, 개발 생명주기의 초기단계에서 요건에 대한 잘못된 이해나 분석, 개발 영역에 대한 충분한 이해와 관리의 부재 및 계속적인 변경 요구는 부정확성하고 불완전한 요건을 발생시키고, 다른 요건과의 충돌 및 일관성 결여 등을 발생시킬 수 있다. 이러한 결과는 전체적인 시스템의 완전성과 성능 등에 커다란 영향을 주고 유지보수에 많은 비용과 노력을 요구한다.

본 연구에서는 이러한 문제점을 해결하기 위하여 실무 프로젝트 개발에 적용할 수 있도록 요건관리와 관련된 프로세스와 활동을 요구공학을 기반으로 완전성과 일치성을 가진 요구사항의 생성 및 관리 등을 포함하는 총체적인 활동과 원칙에 대한 공학적 접근을 제시하여 전체 소프트웨어 개발비용과 위험부담을 경감시키며 품질향상을 이룰 수 있도록 한다. 또한, 각 업무 도메인과 개발 환경에 따라 적절하게 적용할 수 있도록 프로세스와 활동을 커스터마이징 및 컴포넌트화하고 요구사항 관리 도구의 프로토타입을 제시한다.

1. 서 론

소프트웨어 개발환경이 대형화, 복잡화됨에 따라 개발 생명주기의 각 단계에서 발생하는 요구사항은 기술적, 관리적 측면에서 많은 영향을 받게 되었다.

소프트웨어 개발 생명주기의 초기단계에서 요구사항에 대한 잘못된 이해나 분석은 부정확하고 불완전한 요구사항을 발생시키고, 다른 경험과 의미를 가지는 다양한 그룹의 영역에 대한 충분한 이해와 관리의 부재는 다른 요구사항과의 충돌 및 일관성 결여 등을 발생시킬 수 있다. 특히 사용자와 개발자에 의한 계속적인 변경요구는 소프트웨어 프로젝트의 성공에 대해 중요한 요소로 나타나고 있다.

이러한 결과는 전체적인 소프트웨어의 완전성과 성능 등에 커다란 영향을 주고 생명주기의 다른 단계에서 발생하는 에러에 비하여 유지보수에 많은 비용과 노력을 요구한다.

본 연구에서는 이러한 문제점을 해결하기 위하여 요구사항 관리와 관련된 프로세스와 활동을 요구공학과 CMM 을 기반으로 개발 참여자들 사이에서 완전하고 일치성을 가진 요구사항의 생성, 관리 및 유지 등을 포함하는 모든 총체적인 활동과 원칙에 대한 공학적 접근을 제시하여 전체 소프트웨어 개발비용과 위험부담을 경감시키며 품질향상을 이루고 소프트웨어 개발 프로세스의 이후 단계에 충실한 기초를 제공할 수 있도록 일관된 요구사항 관리를 위한 단계 및 활동, 관리 기법 그리고 관리 도구들이 통합된 요구공학 프레임워크를 구축하는데 목표를 둔다.

또한, 이렇게 구축된 프레임워크가 실제적인 업무에 사용되어지기 위해서는 다양한 프로젝트 환경에 맞도록 커스터마이징 하는 메커니즘까지가 함께 제공되어져야 하는 데 본 논문의 후반부에서는 이 부분에 대한 아이디어가 기술되어져 있다.

2. 관련연구

2.1 요구공학

요구공학은 그림 1 과 같이 요구사항 관리에 관한 모든 활동과 원칙에 대한 체계적이고 총괄적인 공학적 접근 방법이다. 즉, 참여자들이 제시하는 다양한 형태의 추상적인 관련 정보로부터 정확하고 의미있는 요구사항을 획득하고, 사용자 요구를 만족시키기 위하여 획득된 요구사항에 대한 완전성과 일관성에 대한 분석을 수행하며, 제안된 시스템 기능의 신뢰성 확보를 위하여 서비스와 기능 그리고 제약사항을 명세한 후, 명세 문서에 대한 일치성, 완전성, 현실성 등의 검증을 수행한다. 또한 요구사항 진화에 따른 변경관리와 추적관리 등을 위하여 수행되는 모든 생명주기 프로

세스와 이를 지원하는 행위를 포함한다[1][2].

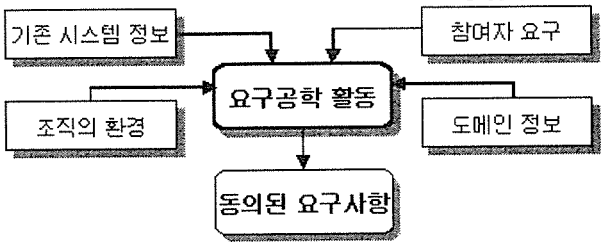

그림 1 요구공학 활동

2.2 CMM 의 요구사항 관리

CMM 의 성숙도 레벨 2 에서는 첫 번째 단계로 그림 2 와 같이 요구사항 관리에 대한 평가를 수행한다. CMM 에서 제안하는 요구사항 관리를 위한 활동은 다음과 같다[3][6].

1) 소프트웨어 요구사항은 문서화되어야 한다.
2) 소프트웨어 요구사항의 명세는 프로젝트의 다른 요소들과의 관계를 정의한다.
3) 기능적 요구사항, 비기능적 요구사항 및 인증기준을 정의하고 문서화한다.
4) 요구사항 관리를 위하여 적절한 자원과 비용이 요구된다.
5) 소프트웨어 요구사항은 개발과 관리를 위한 기준선 설정과 제어가 필요하다.
6) 요구사항에 대한 변경은 검토되고 프로젝트 계획에 포함되어야 한다.
7) 각각의 요구사항에 대한 상태를 포함하여 측정이 수행되어야 한다.

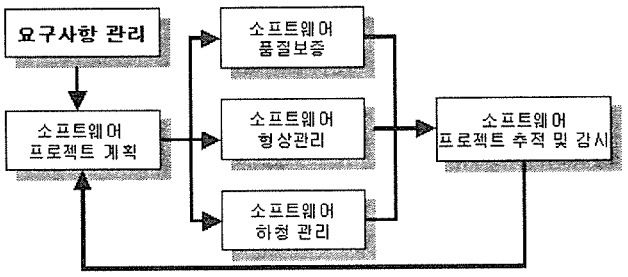

그림 2 CMM 레벨 2 Repeatable

3. 요구사항 관리 프레임워크

동일한 프레임워크의 기반위에서 여러가지 기능을 가지는 어플리케이션이 상호협력하여 하나의 소프트웨어를 구성하듯이, 서로 다른 종류의 전문가들의 협력을 통해 소프트웨어의 요구사항을 도출해 내는 작업 역시 일관된 하나의 프레임워크가 필요하다. 이러한 프레임워크를 구성하는 요소로서 실질적인 액티비티를 정의하고 있는 요구사항 관리 프로세스, 각 프로세스별로 적용 가능한 요구사항 관리 기법, 그리고 식별된 요구사항들을 정제해 나가는 과정이나 요구사항을 관리하는 과정에서 필요한 도구가 있을 수 있다.

본 논문에서는 그림 3 과 같이 요구사항 관리지원 환경을 구축하기 위하여 요구공학 기반의 단계별 활동과 관리 기법 및 관리도구를 통합하는 프레임워크를 제시한다[5].

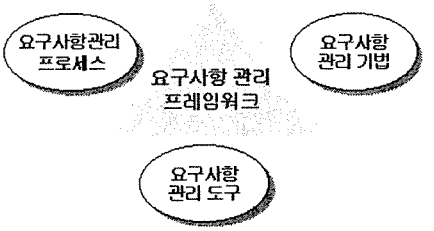

그림 3 요구사항 관리 프레임워크

3.1 요구사항 관리 프로세스

요구사항 관리를 위해 필요한 액티비티로는 어떤 것들이 있으며, 누가 어느 시점에서 작업하여 어떤 산출물을 생성하는지 등을 정의한다. 요구사항 관리 프로세스는 전체 5 단계로 나뉘어져 있으며, 각 단계는 아래의 그림 4 와 같다[2][14][4].

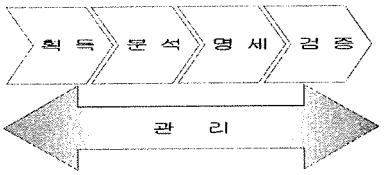

그림 4 요구사항 관리 프로세스

첫째 획득 단계는 고객의 문제가 무엇인지를 파악하여, 문제로부터 기인되는 고객의 요구사항들을 추출하는 단계이다. 또한 요구사항의 추출 이전에 현재 시작하고자 하는 시스템이 과연 현실적으로 구축 가능한지 혹은 현재의 문제점을 해결할 수 있는 솔루션이 될 수 있는지에 대한 검토가 필요하며, 이를 토대로 요구사항을 어떤 식으로 관리할 지에 대한 계획이 수립되어야 한다. 이러한 사전 준비가 완료되고 나면 실제로 여러 가지 추출기법들을 프로젝트의 성격에 맞게 운영함으로써, 정확한 고객의 요구사항을 추출할 수 있다.

표 1 획득단계의 활동

단계	액티비티	태스크
획득	문제분석	현시스템의 문제점들을 파악한다. 프로젝트 관련자와 최종사용자를 식별한다. 시스템의 목적을 정확히 설정한다.
	시스템 구현 가능성 분석	시스템 구축 가능성과 효과를 분석한다.
	요구사항 관리 계획 수립	요구사항 타입을 정의한다. 요구사항 타입별 추적성을 설정한다. 요구사항 타입별 특성을 정의한다. 정의된 요구사항 관리체계를 도구에 매핑한다.
	시스템 환경 정의	시스템 사용자 조직의 특성을 파악한다. 시스템구축 관련 제약사항들을 식별한다.
	공통업무용어 정의	업무용어사전을 작성한다.
	초기요구사항 추출	요구사항을 추출할 대상을 결정한다. 초기 요구사항을 수집한다.

두 번째 분석 단계에서는 요구사항 획득 단계를 통해 수집된 초기 요구사항들을 분류하여 이미 정의되어 있는 요구사항 타입 중 어디에 해당되는지를 파악하여 할당하고, 각각에 알맞은 특성들을 부여함으로써 서로간의 연관관계를 파악한다. 이런 작업을 통하여 구축해야 할 시스템의 모습을 유스케이스 모델링 방법을 통하여 표현한다. 유스케이스 모델링은 시스템의 기능적인 요구사항을 처리하며, 나머지 비기능적인 요구사항들은 별도의 문서에 선언적인 문장들로 명세한다. 또한 고객과의 원활한 의사소통과 시스템간의 효율적인 인터페이스를 위해 고객과 시스템, 시스템과 타시스템간의 상호작용을 담당하는 영역클래스(Boundary Class)에 대한 간략한 명세서를 작성하여 이해를 돕도록 한다.

표 2. 분석단계의 활동

단계	액티비티	태스크
분석	요구사항분류 및 우선순위 식별	액터를 식별한다.
		식별된 초기요구사항별로 각 특성을 식별한다.
		초기요구사항을 특성기반으로 분류하고 정제해 나간다.
		기능적인 요구사항들은 유스케이스 모델로 매핑한다.

표 2. 분석단계의 활동(계속)

단계	액티비티	태스크
분석	요구사항분류 및 우선순위 식별	유스케이스와 비기능적 요구사항들의 우선순위를 식별한다.
	요구사항 분석모델 확립	개략적인 유스케이스 명세서를 작성한다.
		유스케이스를 재구성한다.
		정제된 유스케이스를 상세화한다.
		비기능적 요구사항들을 상세화한다.
		요구사항 베이스라인을 설정한다.
	인터페이스 분석	유스케이스 명세서를 기반으로 영역클래스를 식별한다.
		영역클래스 명세서를 작성한다.

세 번째 명세 단계에서는 고객으로부터 추출하여 정형화된 형태로 분석한 요구사항들을 일관성있게 문서화 하는 단계로, 본 논문에서는 요구사항 자체에 대한 명세서로 소프트웨어 요구사항 명세(SRS:Software Require ment Specification)를 작성한다.

소프트웨어 요구사항 명세문서는 프로젝트 전체에 걸쳐서 시스템의 구축을 제어할 수 있는 문서로 모든 프로젝트 개발팀과 관련 있는 문서이다. 각각 참여자별 관점에서 본 소프트웨어 요구사항 명세문서의 사용 목적은 아래와 같다.

- 요구사항 분석가 : 실제 소프트웨어 요구사항 명세 문서를 작성하는 역할을 한다.
- 설계자 : 클래스의 역할과 오퍼레이션, 속성 등을 정의할 때나 클래스를 구현 환경으로 적용시킬 때 소프트웨어 요구사항 명세 문서를 근거로 작업하게 된다.
- 운영자: 클래스 구현시에 참고자료로 사용한다.
- 프로젝트 관리자 : 이터레이션 계획시의 참고자료로 사용한다.
- 테스트시험자 : 시스템의 성능 검증시의 참고자료로 사용한다.

표 3. 명세단계의 활동

단계	액티비티	태스크
명세	요구사항 명세	초기 요구사항 명세서를 작성한다.
		정제된 요구사항들을 수집한다.
		소프트웨어 요구사항명세서를 작성한다.

네 번째 검증 단계에서는 요구사항 자체가 과연 적절한 소스로부터 추출되어 잘 분석되었는지를 검증함과 동시에 이러한 요구사항이 형식화된 요구사항 정의서에서 제대로 잘 표현되고 있는지를 검증한다. 또한 요구사항을 획득, 분석, 명세하고 관리하는 일련의 과정상에 문제점이 없는지를 검증하는 단계이다[9].

표 4. 검증 단계의 활동

단계	액티비티	태스크
검증	요구사항 검토팀 구성	요구사항 검토팀을 구성한다.
	정제된 요구사항 검증	정제된 요구사항의 타당성을 검증한다.
		요구사항간의 추적성을 검토한다.
		요구사항의 결함을 보완한다.
		검토/보완된 요구사항에 대해 고객의 동의를 획득한다.
	요구사항 관련문서 검증	업무용어 사전을 검증한다.
		초기 요구사항 문서를 검증한다.
		유스 케이스 모델을 검증한다.
		비기능적 요구사항 명세서를 검증한다.
		소프트웨어 요구사항 명세서를 검증한다.
	요구사항관리 프로세스 검토	요구사항관리 프로세스 자체가 효율적인지를 검토한다.

마지막으로 관리 단계에서는 프로젝트 생명주기를 통하여 개발 관련자로부터 제시되는 변경요구나 개발 중에 발생하는 문제보고에 의해 시스템 요구사항의 변경요소를 식별 및 정의하고, 변경된 요구사항을 추적하여 영향분석을 하며, 또한 변경된 형상과 상태를 레파지토리에 저장하고 관리하는 활동과 태스크를 정의함으로써, 시스템 요소의 완전성과 정확성을 증명하는 관리적, 기술적 절차를 수행한다[10].

표 5. 관리 단계의 활동

단계	액티비티	태스크
관리	요구사항 변경관리	변경관리 정책을 정의한다.
		각 요구사항에 유일한 식별자를 할당한다.
		전역 시스템 요구사항을 식별한다.
		휘발성 요구사항을 관리한다.

표 5. 관리 단계의 활동(계속)

단계	액티비티	태스크
관리	요구사항 추적관리	요구사항 추적 정책을 정의한다.
		요구사항 추적 매뉴얼을 작성한다.
		요구사항 변경 영향 분석을 수행한다.
	요구사항 형상관리	요구사항 데이터베이스를 구축한다.
		변경내용을 데이터베이스에 저장한다.
		거부된 요구사항을 기록한다.

.3.2 요구사항 관리 기법

3.2.1 시나리오/Goal 기반 요구사항 획득

초기 요구사항을 획득하는 기존의 기법들이 요구 사항의 단편적 획득에 그치기 때문에, 시나리오/Goal 요구사항 추출 기법들을 통해 기능/비 기능적 요구사항을 보다 명확하게 추출할 수 있도록 한다. 또한 요구사항의 품질 속성(quality attribute)을 요구사항 획득, 분석 단계부터 고려하여 개발하도록 함으로써 명확한 요구 사항에 대한 이해와 시스템의 품질 향상을 가져다 줄 수 있는 기반을 제공한다.

본 논문에서는 요구사항 추출을 위해 다음과 같은 절차를 제시한다.

1) 비즈니스 지식 획득
2) 시나리오 생성
3) 품질 속성 매핑
4) 시나리오 분류
5) 상향식 목표 추출
6) 하향식 목표 추출
7) 최종 요구 사항 선택

요구사항을 초기 단계에 추출해 내는 것은 비즈니스에 대한 지식이 어느 정도인가에 따라서 결정된다. 그러므로 비즈니스에 대한 지식을 얻기 위한 방법으로 질의를 사용하여 비즈니스 지식을 획득하고, 획득된 지식으로 메타 모델을 만든다. 이 메타 모델에 의해 생성된 개체들을 중심으로 시나리오를 통한 요구사항을 생성하고 생성된 요구사항을 품질 속성과 매핑하는 단계를 거친다. 최상의 목표로부터 그것을 달성하기 위한 하위 목표를 식별하는 기법인 하향식 목표 추출로 이런 절차를 검증하는 순서를 거친다[7][8][12].

3.2.2 Use Case 를 이용한 요구사항 모델링

요구사항 획득과정을 통해 추출된 초기요구사항들 중 시스템의 기능적인 측면과 관련된 요구사항들을 유스케이스를 이용하여 구조화한다. 먼저 시스템을 사용하는 사람들의 역할 혹은 인터페이스를 해야 할 대상이 되는 시스템을 대변하는 액터를 식별한다. 정확한 액터의 식별을 통해 구축해야 할 시스템의 영역을 정확히 정의할 수 있다. 다음으로 액터별로 관련되어 있는 요구사항들을 분류한다. 이때 하나의 그룹에 속하는 요구사항들의 사이즈를 예측하여, 각각의 그룹이 균등한 크기의 기능성을 제공할 수 있도록 분류해 나간다. 이렇게 식별된 요구사항 그룹을 대표할 수 있는 기능의 명칭 혹은 업무의 명칭을 액터의 입장에서 부여함으로써 하나의 유스케이스가 식별된다.

각 유스케이스가 포함하고 있는 요구사항들을 액터와 시스템간의 대화형식으로 유스케이스 명세서에서 기술한다. 유스케이스 모델은 액터와 유스케이스, 유스케이스와 유스케이스간의 관계를 표현한 유스케이스 다이어그램과 유스케이스 명세서로 구성된다.

유스케이스를 이용한 요구사항 모델링의 장점은 선언적인 문장으로 나열되어 있던 기능적인 요구사항들을 시스템 사용자 입장에서 하나의 문맥상의 대화형식으로 기술함으로써, 고객에게 분석한 내용에 대해 쉽게 설명하고 동의를 구할 수 있다는 것이다[14].

3.2.3 품질 요구사항을 위한 자동 분류

소프트웨어 시스템의 품질 속성이란 그 소프트웨어 시스템의 품질에 영향을 주는 속성이다. 품질 속성은 소프트웨어 개발 단계 중 요구사항 분석 단계에서 이해당사자들에 의해 품질 요구사항으로 추출된다. 품질 요구사항간의 충돌을 프로젝트 초기에 발견하기 위해서는 초기 수집된 품질 요구사항들을 각 품질 속성별로 분류하는 작업이 필요하다.

품질 요구사항 분류를 위한 작업은 그림 5 와 같이 두 단계로 이루어진다. 먼저 문장의 의미를 대표하는 키워드 및 키워드간의 조합을 표현할 수 있는 분류언어를 개발하여 문장간의 분류를 돕는다. 다음으로 문장간의 유사도 측정을 통해 누락된 문장을 재분류 하도록 한다[15].

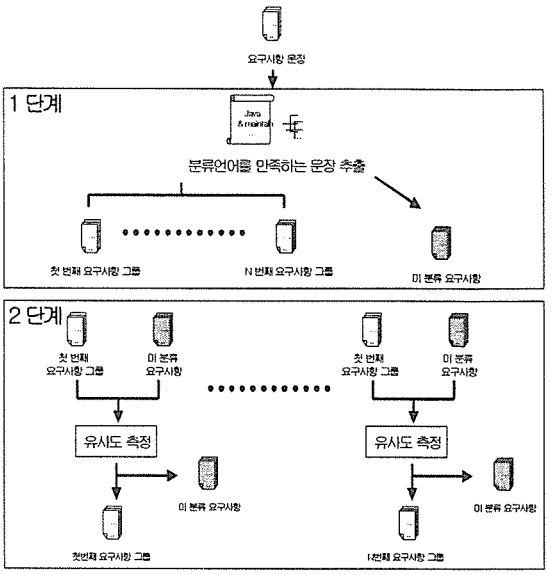

그림 5 분류언어와 유사도를 이용한 2 단계 분류 기법

3.2.4 유사도 측정을 이용한 요구사항 변경관리

효율적인 요구사항 분석 작업과 요구사항분석서의 유지보수를 위해 요구사항 분석을 지원하는 자동화된 도구의 개발이 필요하며, 이러한 도구는 요구사항 분석과 전체 프로젝트 수행을 위해 좋은 기초를 제공해 줄 것이다.

본 논문에서는 문서간의 의존성(dependency)과 문장간의 연계성(traceability), 불일치성, 불명확성 그리고 불완전성 등의 문제점들을 효율적으로 관리할 수 있는 요구사항 분석 기법을 사용한다. 제안된 요구사항 분석 기법은 정보 검색 기술의 유사도(similarity)검사 기법을 기반으로 한다.

이 기법은 공기 정보(co-occurrence information)를 이용하여 문서간 유사도를 측정하고, 문서 내에서의 일관성이 결여된 문장과 불명확성을 가진 문장을 찾아주는 통합 기법이다.

요구사항 분석 시 발생하는 오류를 효과적으로 줄이고 수정하기 위하여 문서간의 유사도 측정을 위해 기존의 색인 추출 방법인 슬라이딩 윈도우 모델과 의존 구조 모델을 결합하여 각 모델이 가지는 단점을 보완한다.

먼 거리 공기 정보는 의존 구조 모델에 의해 찾을 수 있고, 문장간 유사도 측정을 위해서는 슬라이딩 윈도우 기법과 Salton 의 코사인 계수를 이용하여 요구사항의 충돌과 일관성이 결여된 문장을 찾을 수 있는 효율적인 방법을 제공한다. 모호한 문장을 찾기 위해서는 미리 구축된 사전과 형태소 분석기를 이용하여 요구사항 분석서에 포함

된 모호한 문장을 쉽게 찾을 수 있도록 한다. 제안된 기법은 언어 분석의 비교적 하위 단계인 형태소 분석과 구문 분석만을 이용하여 유사도를 측정하기 때문에 비교적 단순하고 쉽게 구현된다. 유사도 측정 기법을 이용하여 연관된 문서를 쉽고 빠르게 찾아 사용자 요구사항 분석 시 발생하는 오류의 분석과 수정에 효과적으로 대처할 수 있다[11].

3.3 요구사항 관리 도구

요구사항의 획득 및 자동분류를 위하여 기존의 요구사항 관리 도구들의 기능에 추가적으로 자연어처리 엔진을 이용하여 요구사항을 자동으로 분류하고 분석한다. WRMS(Web-Based Requirements Management System)의 시스템 구성요소를 간단히 살펴보면 요구사항 및 요구사항과 관련된 여러 가지 정보를 저장하기 위한 DBMS(Database Management System), 웹상에서 동적으로 입력되는 내용들을 처리하기 위한 어플리케이션 서버(Application Server), 실제로 요구사항을 관리하게 될 코어 어플리케이션(Core Application), 사용자와 상호작용 하게 되는 웹 서버(Web server)로 구성된다. 여기에 제시되는 WRMS 의 개요적 구조(Overall Architecture)는 그림 6 과 같다.

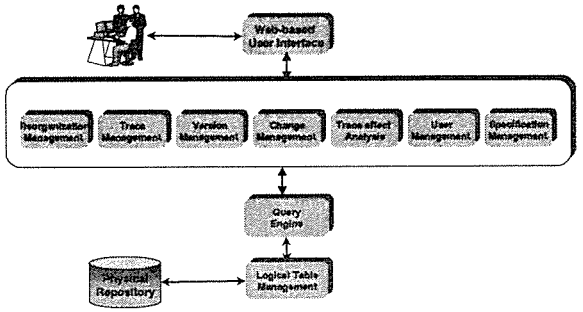

그림 6 WRMS 체계도

사용자들은 웹 기반의 사용자 인터페이스를 통하여 요구사항들을 입력하게 되고 입력된 요구사항으로 본 도구는 다음과 같은 기능을 수행하게 된다.

1) 웹기반 환경에서의 요구사항 추출
   사용자가 원하는 시스템 요구를 충족시키기 위한 요구사항의 속성을 추출하고 각각의 추출된 요구사항들에 번호를 부과하고 계층적 입출력을 하여, 효과적인 요구사항 관리를 지원하다.
2) 추출된 요구사항을 재구성
   개개의 추출된 요구사항들을 형식기준에 맞게 구성한다. 분석가는 사용자가 입력한 요구사항들을 비교, 분석하여 반복, 충돌, 불일치 등을 제거하고, 요구사항들을 주제별로 분류한다.
3) 요구사항들 간의 연계성 지원
   주제별로 분류된 요구사항들간의 유사도 계산을 통한 연계성을 유지한다.
4) 요구사항들의 변경관리
   새로운 요구사항의 추가, 기존 요구사항의 변경 및 삭제로 인해 파급되는 영향을 분석하고 변경되는 사항에 대한 모든 기록을 보관, 관리한다.
5) 사용자 보안관리
   요구사항 문서들에 관한 보안 유지를 위해 각 사용자들에 대한 접근 권한을 유지 관리한다.
6) 명세화 관리
   기술된 요구사항들을 표준형식에 근거하여 작성, 관리한다.
7) 버전 관리
   요구사항들의 문서버전을 관리한다.
8) 변경영향 분석기능
   요구사항의 변경으로 인한 영향 분석을 한다.

본 도구는 UML 의 방식에 의해 분석, 설계되었다. 따라서, 시스템의 설계도도 UML 에서 정의하는 다이어그램들로 이루어진다. 이들은 바로 Use Case Diagram, Sequence Diagram, Class Diagram 들이다. Use Case Diagram 에서는 시스템에서 제공하는 기능들을 한눈에 볼 수 있고, 외부와의 상호 작용을 알 수 있다. Sequence Diagram 에서는 세부적인 기능들간의 메시지 전달과정을 볼 수 있다. Class Diagram 에서는 클래스와 클래스들간의 관계를 보여준다. 이들 세 다이어그램들로부터 상위 단계의 기본 개념에서부터, 하위 단계의 세부적인 처리부분까지 알 수 있다[13].

4. 요구사항관리 프로세스의 커스터마이징 방안

앞서 3 장에서 소개한 요구사항 관리 프레임워크는 보편적으로 소프트웨어를 개발하는 데 있어서 정확한 요구사항을 추출하고 관리하기 위해 필요한 액티비티들의 합집합이라 할 수 있다. 물론 앞서 설명된 모든 액티비티들을 모두 수행할 만한 환경을 갖춘 조직이라면 프레임워크의 즉각적인 적용에 문제가 없다. 그러나, 여기서 한가지 생각할 것은 모든 소프트웨어 개발시에 적절한 전문가가 투입되고, 충분한 일정과 비용이 허락되

어지지 않는다는 점이다. 즉, 소프트웨어 개발팀의 규모나 일정, 예산 혹은 개발하고자 하는 소프트웨어가 사용되어질 비즈니스의 특성 등에 따라 적절한 형태로 프로세스를 커스터마이징할 필요가 있다. CMM 5 단계에서 정의하는 바와 마찬가지로 프로세스는 정적으로 문서상에 기술되어 있는 것이 아니라, 조직의 상황에 따라 변화하고 적응해 가야 한다.

현재까지 발표되어 있는 요구사항 관리를 위한 프로세스들은 그 내용면에서 완벽한 일관성을 유지하고 전체적으로 수행되어야 할 작업에 대해서 정의하고 있다 하더라도, 소프트웨어 프로세스팀을 별도로 가지지 않은 조직에서 실질적인 프로젝트에 적용하는 작업 자체에 어려움을 겪음으로써 그 요구사항 관리 프로세스가 무용지물이 되는 경우가 허다한 실정이었다.

이러한 실질적인 문제를 해결하기 위한 프로세스 커스터마이징 메커니즘을 제시하자면, 다음과 같다. 우선, 3.1 절에서 기술한 프로세스의 최소단위인 태스크를 하나의 독립적인 컴포넌트로 식별한다. 여기서 하나의 태스크를 이루고 있는 항목들을 메타모델화한 것을 나타내면, 아래 그림 7 과 같다.[16]

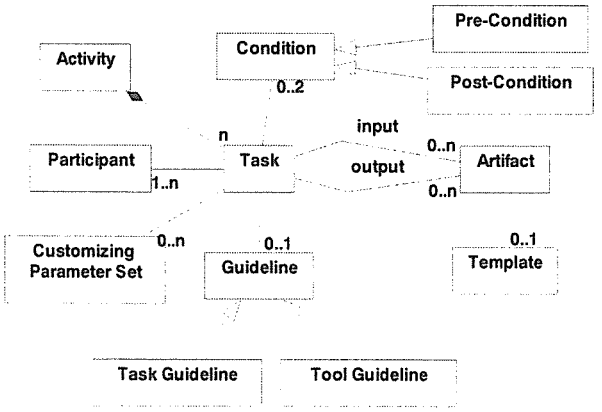

그림 7. 태스크의 메타모델

그림 7 에서 보면 하나의 태스크는 그 각각이 커스터마이징을 위한 파라미터 집합을 가지고 있다. 예를 들어 " 업무용어사전을 작성한다" 는 태스크의 경우, 프로젝트의 형태에 따라 이 태스크는 요구사항 프로세스에 포함될 수도 있고, 그렇지 않을 수도 있다. 프로젝트의 형태가 내부개발 프로젝트일 경우, 대부분의 개발자들은 이미 수년간 해당 시스템을 유지보수해온 내부인력일 것이다. 따라서 별도의 업무용어에 관한 정의 없이도 요구사항을 추출하거나 명세하는 데 있어서 서로의 의사소통에 문제가 없다. 이런 경우에 무조건적으로 모두가 이미 알고 있는 업무용어들을 다시 수집하여 그 정의를 내리고 문서화 하는 것은 오버헤드가 될 수 있다. 그러나 반대로, 프로젝트의 개발형태가 외주프로젝트로 개발자들 대부분이 해당 업무에 대해 생소한 상태라면, 시스템 구축을 위한 서로간의 원활한 의사소통을 위해 업무용어집은 반드시 필요하다. 이런 경우 " 프로젝트의 형태" 는 하나의 커스터마이징 파라미터가 되며 이 파라미터의 실질적인 인스턴스 값은 " 내부개발 프로젝트" , " 외주프로젝트" 등이 될 수 있다. 이러한 관계를 태스크의 메타모델을 기반으로 나타내면 아래 그림 8 과 같다.

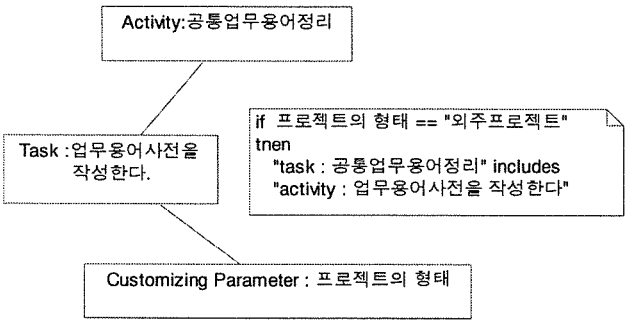

그림 8. 태스크의 커스터마이징 규칙 표현

각각의 태스크가 프로세스에 포함되는 지의 여부가 결정이 되었다면, 다음으로 고려해야 할 것은 프로세스간의 수행 순서를 결정하는 것이다. 요구사항 프레임워크상의 프로세스 정의에서는 프로세스간의 선후관계가 설정되어 있으나, 커스터마이징의 결과 중간 태스크들이 선택되지 않음에 따라 그 선후관계의 연결이 끊어질 수 있다. 따라서, 프로세스 간의 수행 순서는 각각의 태스크에 정의되어 있는 입력산출물(Input Artifact), 출력산출물(Output Artifact), 사전조건(Pre-Condition)과 사후조건(Post Condition)들 간의 연관관계를 참조하여 결정한다.

프로세스를 이루고 있는 가장 작은 작업단위인 태스크를 하나의 독립된 컴포넌트로 정의함으로써, 하나의 태스크가 특정 프로젝트에 적용이 되어야 하는지 아닌지, 태스크의 작업시점이 어떤 태스크가 수행된 이후여야 하는지에 대한 정보를 태스크 자체가 단독적으로 보유하게 된다.

태스크와 관련된 항목들을 나타낸 것이 그림 8 이라면, 태스크를 바깥에서 바라보는 모습은 그림 9 과 같다.

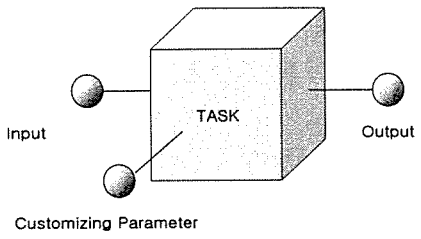

그림 9. 태스크 컴포넌트

3 장에서 제시한 요구사항 프로세스는 위의 그림과 같은 태스크 컴포넌트가 조합되어, 액티비티 컴포넌트를 이루고, 다시 이러한 액티비티 컴포넌트들의 조합이 단계 컴포넌트를 이루어, 5 개의 단계를 가지는 형태를 갖추었다고도 볼 수 있다. 이처럼 프로세스를 무형의 지침이나 변화할 수 없는 문서가 아닌 하나의 컴포넌트로 식별함으로써, 다양한 개발환경과 도메인에 대해 능동적으로 적용해 나갈 수 있다.

현재 대표적인 몇몇의 커스터마이징 파라미터만을 식별한 상태로, 커스터마이징의 판단근거가 될 수 있는 모든 파라미터들을 식별해 내고, 파라미터들간의 상호작용을 분석해 내는 것이 본 연구의 향후 진행방향이다.

5. 결론

본 연구에서는 기존의 요구사항 관리가 분석과 문서의 생성에 관련되어 전체 개발 공정에 대한 해결책이 미흡하였던 문제를 해결하기 위하여, 소프트웨어 개발 비용과 위험부담을 경감시키며 품질향상을 이룰 수 있도록 요구공학 관리 프로세스와 CMM 에서 제시하는 활동을 기반으로 안정적인 요구사항의 생성 및 변경에 대한 관리기법 및 도구 등이 포함된 프레임워크의 설계를 제시하였다. 이를 통하여 연구에 포함된 내용은 다음과 같다.

1) 소프트웨어 개발 향상을 위한 요구사항 관리 프로세스 정의
2) 안정성과 완전성을 위한 요구사항 관리 기법
3) 요구사항 관리 지원도구의 설계

본 논문에서 제시하는 프레임워크를 통하여 체계적인 요구사항 관리에 의한 소프트웨어 개발 생산성의 향상, 안정성있는 요구사항 관리에 대한 기술 습득과 같은 기대효과를 얻을 수 있다.

향후에는 제시된 요구사항관리 프레임워크를 실제 업무에 적용 및 사례분석을 통하여 평가하여, 각 업무 도메인과 개발 환경에 따라 적절하게 적용할 수 있도록 프로세스와 활동을 커스터마이징 및 컴포넌트화하는 연구와 또한 제시된 도구의 프로토타입을 근간으로, 웹 환경의 요구사항 관리도구 개발을 위한 연구가 계속될 것이다.

참고문헌

[1] G. Kotonya, I. Sommerville, "Requirements Engineering Process and techniques", John Wiley & Sons Ltd, 1998.

[2] I. Sommerville, P. Sawyer, "Requirements Engineering - A Good Practice Guide", John Wiley & Sons Ltd, 1997.

[3] CMU SEI, "The Capability Maturity Model - Guidelines for Improving the Software Process", Addison Wesley, 1995.

[4] INCOSE, "Requirement Management," http://incose.org/workgrps/rwg, 1996.

[5] Enrique Garca Alczar, Antonio Monzn, "A Process Framework for Requirements Analysis and Specification," IEEE 4th International Conference on Requirement Engineering, June, 2000.

[6] D. A. Leffingwell, "A Field Guide to Effective Requirements Management Under SEI's Capability Maturity Model", Rational Software Corporation. 1998.

[7] A. I. Anton, "Goal-based requirements analysis," Proc. of the 2nd IEEE Int'l Conference on Requirements Engineering, 1996.

[8] Holbrook, H., "A Scenario-based Methodology for Conducting Requirement Elicitation", ACM Software Engineering Note, Vol.15, No.1, 1990.

[9] Sooyong Park, Harksoo Kim, Youngjoong Ko and Jungyun Seo. "Implementation of an efficient requirements analysis supporting system using similarity measure techniques.", Information and Software Technology, Vol. 42, Issue 6, April 2000, pp. 429-438.

[10] Dean Leffingwell, Don Widrig, "Managing Software Requirements – A Unified Approach", Addison-Wesley, 2000.

[11] Sooyong Park, Hoh Peter In, Soonhwang Choi, "Automated Support to Quality Requirements Classification for Supporting WinWin", 2nd International Conference on Computer and Information Science, pages. 43-58,2002.8