소프트웨어 프로덕트 라인에서 핵심 자산으로서 요구사항을 관리하는 방법

(An Approach to Managing Requirements as a Core Asset in Software Product-Line)

문 미 경 † 염 근 혁 ††
(Mikyeong Moon) (Keunhyuk Yeom)

요 약 소프트웨어 프로덕트 라인 공학의 목표는 일련의 유사한 소프트웨어 시스템의 공통성과 구별되는 특성을 이해하고 제어함으로써 시스템의 체계적인 개발을 지원하는 것이다[1]. 이것은 소프트웨어 개발 시 나오는 산출물들을 핵심 자산으로 만들어 놓고 이를 체계적으로 재사용 할 수 있도록 지원하기 위한 프레임웍 역할을 한다. 현재 많은 기술들이 프로덕트 라인 공학 관련하여 연구되고 있지만, 그 초점이 소프트웨어 아키텍쳐나 상세 설계 또는 코드에 맞추어져 있다[2]. 소프트웨어 프로덕트 라인 공학에서는 컴포넌트의 공급, 조립뿐만 아니라 조립공정까지 특정 요구나 변화에 신속히 적응할 수 있도록 관리하는 것이 중요한데, 이는 요구사항 분석 단계에서부터 이루어져야 한다.

소프트웨어 프로덕트 라인 공학에서 요구사항은 전통적인 시스템 개발에서와 마찬가지로 모든 개발의 기초가 되는 부분이며, 다른 핵심 자산의 공통성과 가변성의 성질을 결정짓게 만들 수 있는 기준이 된다. 그러나 요구사항들을 다 반영하기도 전에 변경이 발생하는 수많은 경험을 해 온 것처럼, 올바른 요구사항을 획득하고 이를 분석, 관리한다는 것은 결코 쉬운 일이 아니다. 특히, 여러 개의 시스템을 개발할 때 사용할 요구사항은 공통성과 가변성의 속성을 가지게 된다. 그러므로 계획할 수 있는 변화에는 충분히 안정적이면서, 반면에 예측하지 못하는 변화에 잘 적응하고 개조될 수 있도록 유연성을 지닌 핵심 요구사항을 개발, 관리하기 위한 체계적인 방법이 필요하다.

본 논문에서는 소프트웨어 프로덕트 라인에서 핵심 자산의 하나인 도메인 요구사항을 관리하는 방법에 대하여 제안한다. 이를 통해 도메인 요구사항에 대한 재사용성을 증대시키고 시스템의 목표를 정확히 세우는 데 투자되는 많은 시간과 노력을 감소시켜 준다. 이는 결과적으로 소프트웨어 개발 시간과 비용을 줄이고, 생산성을 향상시키는 등의 장점을 가져다준다.

키워드 : 요구사항 관리, 요구 공학, 도메인 공학, 핵심 자산 개발, 프로덕트 라인 공학

Abstract The goal of product line engineering is to support the systematic development of a set of similar software systems by understanding and controlling their common and distinguishing characteristics. The product line engineering is a process that develops reusable core assets and develops a set of software-intensive systems from a common set of core assets in a prescribed way. Currently, many software development technologies are accomplished in context of product line. However, much of the product line engineering research have focused on the reuse of work products relating to the software's architecture, detail design, and code. The product lines fulfill the promise of tailor-made systems built specifically for the needs of particular customers or customer groups. In particular, commonality and variability play central roles in the all product line development processes. These must be treated already during the requirement analysis phase.

Requirements in product line engineering are basis of software development just like as traditional system development engineering, and basis of deciding other core assets' property - commonalities and variabilities. However, it is difficult to elicit, analyze and manage correct requirements. Therefore,

· 본 연구는 한국과학재단 목적기초연구(R01-2003-000-10197-0)지원으로 수행되었음
† 비 회 원 : 부산대학교 컴퓨터 및 정보통신연구소 연구원
mkmoon@pusan.ac.kr
†† 종신회원 : 부산대학교 컴퓨터공학과 교수
yeom@pusan.ac.kr
논문접수 : 2003년 11월 4일
심사완료 : 2004년 6월 16일

it is necessary to develop systematic methods which can develop and manage requirement as core asset, which can be stable in anticipative change and can be well adapted to unpredictable change.

In this paper, we suggest a method of managing requirements as core asset in product line. Through this method, the reuse of domain requirements can be enhanced. As a result, the cost and time of software development can be reduced and the productivity can be increased.

Key words : Requirement Management, Requirement Engineering, Domain Engineering, Core Asset Development, Product-line Engineering

1. 서 론

오늘날 빠르게 변화하는 시장에 프로덕트들을 성공적으로 가져가기 위한 방법 중 한 가지는 프로덕트 라인 공학이다. 소프트웨어 프로덕트 라인 공학의 목적은 일련의 비슷한 소프트웨어 시스템들의 공통성을 이해하고 제어하며, 특징들을 구별시킴으로써 체계적으로 소프트웨어 프로덕트 패밀리를 개발할 수 있도록 지원하는 것이다. 이때 도메인 공학 관련한 많은 연구들이 프로덕트들의 공통성과 가변성을 관리하면서 핵심 자산들을 개발하는데 초점을 두었다. 도메인 공학 프로세스에서 산출하는 핵심 자산들에는 요구사항, 아키텍처, 컴포넌트뿐만 아니라 테스트 플랜, 개발 문서 등이 포함된다. 이중, 도메인 요구사항은 전통적인 시스템 개발에서와 마찬가지로 모든 개발의 기초가 되는 부분이며, 다른 핵심 자산의 공통성과 가변성의 성질을 결정하는 기준이 된다.

도메인 요구사항과 관련한 기존 연구를 두 가지 측면에서 살펴보면, 도메인에서 가장 핵심이 되는 공통성과 가변성(Commonality and Variability : C&V) 분석에 관련한 것과 요구사항에 관련한 것이다. C&V 분석에 대한 기존 연구들에서는 공통성과 가변성을 추출하는 객관화 된 근거 없이 개발자의 직관이나 도메인 전문가의 경험에 의해 그 결정을 내리고 있다. 또한 C&V는 시스템의 기능의 유무에 따라 공통성이 우선 결정되고, 상세 수준으로 내려가면서 가변성이 추출된다. 즉, 공통성과 가변성이 같은 수준에서 추출 될 수 있는 것이 아님에도 불구하고, 기존 연구에서는 그 수준을 구분하지 않고 있다. 가변성 역시 핵심 자산마다 추출될 수 있는 상세정도가 다르고 그 종류가 다르지만, 기존 연구들에서는 핵심 자산의 구분 없이 일괄되게 가변성을 취급하고 있으며 대부분 암시적으로 컴포넌트를 그 대상으로 하고 있다. 요구 공학 측면에서 살펴보면, 요구공학이 독립된 자체의 프로세스를 가지고 있지만, 그 세부 단계에 프로덕트 라인의 개념이 포함된, 즉 C&V를 고려한 요구 공학 프로세스를 제시하지 못하고 있다. 요구사항 식별 단계에서 찾아진 C&V가 요구사항 분석, 모델에 반영되어 이어지지 못한다면 가변성의 추적성을 가질 수 없게 된다. 이는 요구사항의 가변성으로부터 아키텍처와 컴포넌트 가변성으로 이어지지 못하게 하는 문제점들을 야기시키게 된다.

본 논문에서는 프로덕트 라인에서 핵심 자산의 하나인 도메인 요구사항을 관리하는 방법에 대하여 제안한다. 도메인 요구사항의 관리에서 이루어지는 세부 활동으로, 도메인 범위를 결정하는 활동, 도메인 요구사항을 추출하고 일반화시키는 활동, 도메인 요구사항을 분석, 모델링 하는 활동, 도메인 요구사항 변경을 계획하는 활동이 있다. 요구사항 수준에서 C&V의 개념을 결합하기 위해 도메인 요구사항 메타 모델이 제시되며, C&V의 근거를 위해 두 가지 메트리스를 이용한다. 도메인 요구사항은 명세 최소 단위로 분할하여 일반화를 수행함으로써 그 수와 복잡성을 줄인다. 일반화 된 요구사항은 재사용성을 높이기 위해 다시 결합되는 규칙을 제시한다. 본 논문에서 제시하는 도메인 요구사항 관리 방법을 통해 도메인 공학 프로세스와 소프트웨어 개발 프로세스 상에서 요구사항들이 체계적으로 재사용 될 수 있도록 하며 요구사항으로 인한 오류의 기회를 줄여줌으로써 결과적으로 생산성의 향상을 가져오게 한다.

2. 관련 연구

2.1 공통성과 가변성 분석

도메인 내의 관련된 시스템들의 공통성과 가변성을 분석하기 위하여 도메인 분석 기법이 사용될 수 있다. 1990년대 초 SEI(the Software Engineering Institute)의 Feature Oriented Domain Analysis (FODA) 방법론의 시작으로, 시스템의 집합 중에서 주도적인 또는 독특한 피처(feature)를 인식하는 것에 기초하여 도메인을 분석하는 방법들이 나왔다[3,4]. FODA에서 '피처(feature)'라는 것은 구현되고 테스트되고 배포, 유지되어져야 하는 기능적 추상화를 뜻한다. 피처 모델링 방법에 Reuse-Driven Software Engineering Business (RSEB)의 프로세스를 통합한 FeatuRSEB(the Featured RSEB) 방법이 있다[5]. 이 방법은 피처 모델을 만드는 활동과 병행하여 유즈케이스 모델을 만든다. 2000년도 이후에 소프트웨어 프로덕트 라인 개발에서 다양성을 표현하기 위한 연구에 피처가 사용되었다[6]. Feature Description Language (FDL)에서는 피처 그래프를 텍스트 형

태로 표현하기 위해 개발된 언어로써 프로덕트 라인 아키텍쳐의 가변성을 추출, 기술하기 위하여 피쳐 그래프를 텍스트 형태로 바꾼 것이다[7].

이처럼 도메인을 분석하는 분야에서 피쳐는 다방면에서 사용되어졌다. 그러나 피쳐의 속성을 결정짓게 하는 논리적인 근거에 대해서는 제시하고 있는 것이 없었다. 피쳐의 분류는 모두 도메인 분석가의 경험이나 직관(heuristic)에 의존되어 있었다[8]. 또한 대부분이 유사한 모델링 접근법을 택하였지만, 그것은 피쳐를 이용한 것이지, 요구사항을 의미한 것이 아니다[9].

2.2 가변성 모델링 기법

가변성 모델링 기법에 패턴을 사용하는 연구가 있다[10]. 여기서 판별식(discriminant)은 하나의 시스템을 다른 시스템과 구분 짓는 어떤 피쳐를 의미하게 되고 이를 패턴과 연관지어 가변성을 모델링한다. 가변성 또는 선택성이 발생하는 부분을 베이스 클래스로 표현하고, 가변되어지는 값들을 서브 클래스의 집합인 영역(realm)으로 묶는다. 이때, 영역에 속하는 서브 클래스가 어떻게 결합되는지에 따라 single adapter pattern, multiple adapter pattern, optional pattern으로 구분한다. UML 확장 기법을 사용하여 가변성을 모델링 한 연구도 있다[11]. 여기서는 UML 모델에 스테레오 타입으로 <<variationPoint>>를 기술하고 각 가변값(variant)들을 상속관계로 표현한다. 여기서는 단지 그래픽한 표현에 초점을 두고 있지만, 본 논문에서 사용하게 될 UML을 확장시켜 사용한다는 점에서 모델링 기법을 이용해 볼 수 있었다. [12] 연구에서는 프로덕트 라인에서 가변성을 모델링하기 위해 매개변수화(parameterization), 정보 은닉(information hiding), 상속(inheritance), 가변성 포인터(Variation Point)를 사용하는 방법들을 제시하고 있다. 이러한 연구들은 특정 핵심 자산의 특징들을 구분하지 않고 모델링 기법에 초점을 두고 있었다. 그러므로 이러한 기법들을 요구사항이라는 특정 핵심 자산에 세분화 시켜 사용할 수 있을 것이다.

2.3 도메인 요구 공학

요구사항은 모든 소프트웨어 계획 수립 및 소프트웨어 개발 활동의 기초를 제공한다. 소프트웨어 개발에서 요구공학(Requirements Engineering)이라는 용어를 사용하면서 요구사항에 관련된 모든 활동들 즉, 요구사항 추출, 분석, 기술 및 관리의 공학적 접근방법을 체계적으로 정리하고 있다. 요구 공학 프로세스는 그림 1에서 보는 것처럼 4개의 주요 분야로 구성된다[13].

비록 4개의 분야들은 그들의 프로세스와 목적이 서로 상의하여 그림 상에 구별되고 순서화되어 표현되었지만, 이들은 프로젝트 개발 단계에서 거의 동시에 수행되기도 하며, 특히 time-to-market 프로젝트인 경우 이 4가지 활동들은 같은 시간대에 수행되도록 요구된다[14]. 이 4가지 활동들이 프로덕트 라인에서 수행될 때는 각 활동마다 C&V를 고려해야 한다. 표 1은 요구 공학 기본 활동들에 대하여 간략히 언급하고 도메인 요구공학에서 초점을 두어야 하는 점을 추가 정리함으로써 두 가지의 차이점을 인식하고자 한다[15].

최근 들어 요구사항 재사용과 관련한 연구들이 몇몇 제시되었다. [16]는 요구사항을 추출하고 구조화하기 위

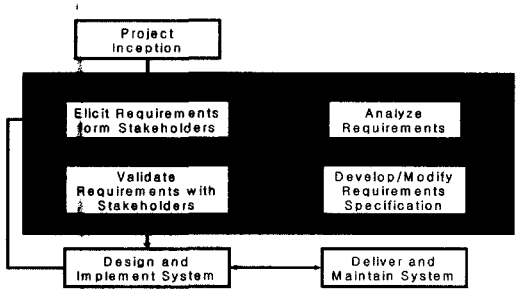


그림 1 요구 공학 프로세스(RE Process)

표 1 요구공학과 도메인 요구 공학의 주요 활동

요구사항 추출 (Eliciting Requirements)	요구사항 추출은 고객과 시스템 사용자, 그 외 시스템 개발에 관심을 가진 자들과 의사 소통함으로써 요구사항을 발견하는 과정이다.	도메인 분석 기법을 이용하여 예상되는 가변성을 명확히 추출하고 그 범위를 확정하는 것에 초점을 둔다.
요구사항 분석 (Analyzing Requirements)	초기 추출된 요구사항들에 대해 갈등(conflict), 중복(overlap), 누락(omission), 불일치성(inconsistency)등을 분석, 정제한다.	C&V를 식별한다. 가변성이 일어날 수 있는 부분을 지적한다.
요구사항 명세 개발 (Developing Requirements Specification)	다른 시스템 관심자들 사이에 요구사항들을 효율적으로 전달하기 위하여 각각의 요구사항들이 서로 모호하지 않고 간결하고 이해력 있게 작성하는 과정이다.	가변될 수 있는 부분은 특정 프로덕트에 따라 확장되고, 인스턴스 되며, 채워질 수 있도록 상징적인 대체자(symbolic placeholder)를 명세서에 포함한다.
요구사항 검증 (Validating Requirements)	요구사항의 완전성, 정확성, 일관성, 명백성을 보증하는 과정이다.	프로덕트 라인 개발 시의 요구사항 뿐만 아니라 특정 시스템 개발 시의 요구사항에 대해서도 검증이 실행되어야 한다.

한 방법으로 정의 계층화 방법(definition hierarchy method)을 제시하였고, [17]는 요구사항을 n-차원적, 계층적인 프로덕트 라인을 위하여 요구사항을 구조화하는 기법을 연구하였다. 이들은 이후 단계인 아키텍처를 생성하고 유도하기 위하여 도메인을 구조화시키는 것에 초점을 맞추고 있다.

3. 도메인 요구사항

'도메인 요구사항'이란 용어를 정확히 사용하기 위해 먼저, 본 논문에서 언급하는 '도메인'에 대하여 정의 내린다. 도메인에 대한 각기 다른 정의들을 많은 문헌에서 발견할 수 있다. 각기 다른 정의들 사이의 주된 차이점은 지식(knowledge)에 관심을 두는가, 또는 어플리케이션에 초점을 두는가 하는 것이다. [19]은 도메인을 지식의 분야 또는 일련의 개념 또는 현장 종사자들이 이해하고 있는 용어(terminology)에 의해 특성화되어지는 행위들이라고 정의한다. 어플리케이션의 관점에서는 도메인을 공통된 특성을 공유하는 현재와 미래의 어플리케이션 집합이다라고 정의한다[20]. 우리는 여기서 후자의 정의를 인용하여 사용한다. 본 논문에서는 '도메인'이란 한정어를 붙여 요구사항을 다루고 있다. '도메인'이란 용어로 인해 우리가 중점을 두어야 할 특징은 '핵심' 요구사항을 인식해 내야 하는 것과 자체적으로 변화할 수 있는 가변성을 식별하고 예측, 계획할 수 있어야 한다는 것이다. 즉, 도메인 요구사항은 시스템들의 공통적인 골격에 대한 요구사항들과 함께 시스템마다 다소 상의하게 나타나는 부분들을 일정수준 추상화시킨 형태를 취할 것이다.

본 논문에서는 도메인 요구사항에 대한 텍스트적 정의를 바탕으로 도메인 요구사항에 대한 메타 모델을 제시한다. 메타 모델을 기반으로 한 요구사항 개발은 모델 요소들 사이의 추적가능성(traceability)을 보장해준다[21]. 모델 요소들 간의 추적가능성은 모델들 간의 일관성을 유지하기 위한 전제조건이 된다. 그림 2는 도메인 요구사항 메타 모델을 나타낸다. 보통 요구사항을 계층적으로 조직화한다. 이 스타일은 이것을 읽는 사람이 시스템의 기능을 이해하는데 도움을 주고 요구사항 작성자가 변경이 필요한 부분을 찾는 것을 도와준다[22]. 요구사항은 크게 두 가지로 분류할 수 있다[23]. 하나는 기능적 요구사항으로 시스템이 수행하는 기능성을 나타내고, 다른 하나는 비기능적 요구사항으로 기능성을 달성하는데 있어서 부가되는 제약 조건 및 품질 특성을 나타낸다. 도메인 요구사항도 계층적으로 조직하여 크게 기능적 요구사항과 비기능적 요구사항으로 나눈다.

기능적 요구사항은 PR(Primitive Requirement)을 가지게 된다. 여기서 PR은 도메인내의 시나리오들을 분석하여 atomic 레벨 컴포넌트의 의미상의 최소 단위인 semantic primitive[24]로 정의한다. 하나의 PR은 여러 개의 PRelement들로 구성된다. 도메인 요구사항은 가변성을 가지는데, 이는 상세 수준에 따라 속성과 가변성 지점으로 나누어진다. 상위 수준에서 PR은 도메인 내 시스템에 나타나는 빈도수에 따라 공통성과 선택성의 속성을 가지게 된다. 공통성/선택성 속성을 가진 PR은 좀 더 하위 레벨로 내려갔을 때, 가변성 지점을 식별해 낼 수 있게 된다. 즉, 도메인 요구사항이 가지는 속성은 크게 공통성, 선택성이 있고, 이는 먼저 기능의 유무를 우선적으로 판단하여 속성이 결정됨을 의미하게 된다. 공통성이나 선택성의 속성을 지닌 기능은 좀 더 하위 수준에서 가변성 지점을 식별해내게 된다. 비기능성 요구사항은 여러 개의 품질특성으로 이루어진다. 일반적으로 품질적 요소는 하나 또는 그이상의 기능적 요구사항과 관련되어 있다[23]. 또한 동시에 다른 품질과 같이 존재함으로써 긍정적이거나 부정적인 영향을 받게 된다. 그러므로 비기능적 요구사항을 구성하는 품질 특성은

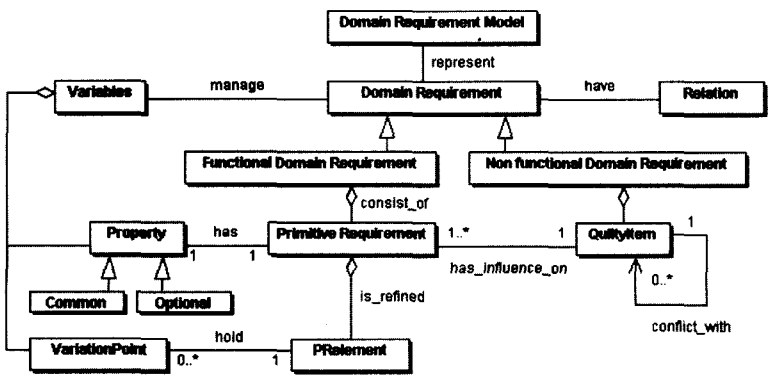


그림 2 도메인 요구사항 메타 모델

기능적 요구사항과 다른 품질 특성과의 연관관계를 가지고 있어야 한다.

4. 도메인 요구사항 관리

위의 메타 모델을 바탕으로 본 논문에서 제시하는 도메인 요구사항 관리의 주요 활동들과 각 활동들을 통해서 나오는 산출물의 관계를 도표로 나타내면 그림 3과 같다. 각 활동들은 각각 독립적이고 특정 순서를 가진 듯 표현되었지만, 실제 이들은 상호 배치되고, 반복적으로, 소프트웨어 시스템 개발 전 과정에 걸쳐 일어난다.

4.1 도메인 범위 결정

도메인을 일련의 '관련된 시스템의 집합'이라고 정의할 때, 도메인의 경계는 모호해 진다. 그러므로 도메인 요구사항 공학의 전제 단계로서, 도메인의 범위를 명확히 구분 짓는 활동이 이루어져야 한다. 도메인의 범위에 따라, 시스템을 개발할 때, 시스템이 도메인에 속하는 것이지 아닌지를 결정지을 수 있기 때문에 이 활동은 가장 우선되면서 가장 중요한 활동이다. 그러나 도메인은 가변성을 항상 내포하고 있기 때문에 그 범위를 정확히 "1에서부터 10까지이다"는 식으로 명시하지는 못한다. 그러므로 도메인의 범위를 정한다는 것은 각각의 시스템이 사용하는 개념과 용어가 일치한다는 것과 시스템의 스타일이나 제약 사항들이 일치한다는 것으로 정할 수밖에 없다.

4.1.1 Terminology 정의서

Terminology 정의서에서는 도메인에서 사용하는 기본 개념들에 대하여 정의를 내리고 용어를 정리해 놓는다. 또한 용어들 사이의 관련성을 기술한다. Terminology 정의서를 통하여, 도메인 내에 산재해 있는, 또는 앞으로 또다시 산출될 요구사항들이 비록 같은 의미를 내포하지만, 다른 용어로 표현되는 것을 막을 수 있다. 즉, Terminology 정의서는 도메인 내의 수많은 '용어'들에 대하여 우선적으로 일반화 과정을 수행시킴으로써 용어의 수와 복잡성을 줄이게 한다. 이는 다음 단계에서 이루어질 요구사항 '문장'에 대한 일반화 과정의 선행 단계이다.

4.1.2 도메인 아키텍쳐 정의서(Domain Architecture Definition)

도메인 아키텍쳐란 최상위 수준에서 추상화된 시스템의 모델이다. 여기서는 도메인내의 시스템이 공통으로 공유하고 있는 시스템 스타일이나, 시스템 개발 제약사항 등을 기술하게 된다. 이것이 도메인 범위를 결정하는데 필요한 이유는, 본 논문에서 정의한 도메인이 공통된 특성을 공유하는 어플리케이션 집합을 뜻하기 때문이다. 도메인 내에 어떤 시스템이 포함되는지를 판단하려고 했을 때, 이는 앞 단계에서 설명한 terminology가 동일해야 하고 시스템 모델 또는 스타일도 같아야 한다.

4.2 도메인 요구사항 추출 및 일반화

도메인 요구사항을 수집하기 위해서는 도메인에 속하는 기 개발된 시스템들과 차후 개발될 시스템들 다수를 분석해야 한다. 극단적인 경우, 개발하려는 도메인이 기존의 시스템을 전혀 가지고 있지 않는다 할지라도, 가장 유사한 도메인을 찾아 그에 속하는 시스템을 분석해 보는 것이 좋다. 도메인 요구사항은 특정 한 고객이나 한 시스템을 위한 것이 아니다. 도메인 요구사항은 앞으로 나타날 임의의 고객과 임의의 시스템의 요구를 만족할 수 있게끔 미리 예측하여 분석되어야 한다. 소프트웨어 공학 분야에서 앞으로의 예측을 위한 가장 신뢰할 수 있는 자료로서 기존의 경험치를 중요하게 여겨왔다. 또한, 모든 요구사항이 똑같이 중요한 것은 아니다. 유인 우주선의 요구사항에는 인스턴트 오렌지 주스와 구명장비가 포함되어 있을 것이다. 그러나 분명히 전자는 후자보다 중요하지 않다. 오렌지 주스가 없다고 해서 발사중지 조치를 취하지는 않겠지만 구명 장비가 고장 나면 발사 명령을 취소할 것이다[22]. 도메인 요구사항에 우선순위는 모든 시스템에 가장 자주 나타나는 공통성 속

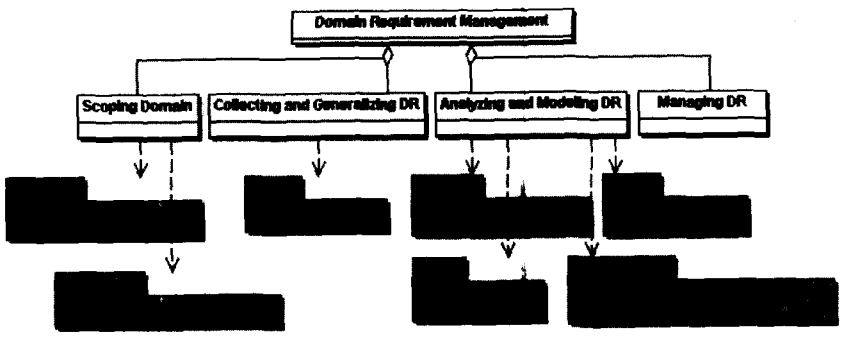


그림 3 도메인 요구사항 관리의 주요 활동과 산출물의 관계

성과 선택적으로 나타나는 선택성 속성으로 표현할 수 있다.

그러므로 본 논문에서는 기존의 시스템들의 요구사항을 수집하고 이로부터 도메인 속성을 추정하기 위해 메트릭스를 만든다. 이 메트릭스는 지금까지 도메인 전문가의 본능적인 감각(heuristic)에 의존해서 인식하게 되는 공통성과 선택성의 추출에 좀 더 객관화 된 근거를 제시하는 역할을 한다.

4.2.1 PR-Context 메트릭스

도메인내의 시나리오들을 분석하여 atomic 레벨 컴포넌트의 의미상의 최소 단위인 semantic primitive[24]로 나눈다. 나누어진 요구사항의 semantic primitive를 여기서는 PR(primitive requirement)라고 정의한다. 레거시 시스템들의 모든 PR들을 그림 4에서 나타낸 바와 같이 메트릭스 형태로 병합한다. 메트릭스의 행과 열이 만나는 곳에 시스템이 PR을 가지고 있다는 의미에서 'O'을 표시하고 그렇지 않은 경우 'X' 표시를 한다. 초기 과정에는 새로운 PR을 연속적으로 만들어 가며 요구사항들을 추출하지만, 분석이 진행되는 동안 레거시 시스템들로부터 동일한 PR이 계속 나오게 된다. 이 메트릭스는 통계적으로 PR이 어떠한 속성을 가질 수 있는지를 결정짓게 해 준다.

4.2.2 도메인 요구사항 일반화

도메인 요구사항은 두 가지 형태로 일반화가 이루어진다. 첫째, 메트릭스의 열에 대한 일반화이다(그림 4에서⇦). 임의의 시스템들이 동일한 PR들로 분해되는 경우, 이들을 'context'라는 용어를 사용하여 하나로 묶고 동일하게 취급될 수 있도록 한다. 즉, 동일한 요구사항들로 이루어진 시스템들은 같은 문맥 하에 존재한다고 볼 수 있기 때문에 하나로 그룹 시킬 필요가 있다. 이것은 메트릭스 열의 수를 줄여준다.

두 번째는 메트릭스 행에 대한 일반화이다(그림 4에서⇩). PR-Context 메트릭스를 분석하여 공통성, 선택성을 지닌 요구사항들로 분류를 한다. 그림 4의 ①처럼 대다수의 context에서 나타나는 PR인 경우에 이를 해당 도메인 내에서 공통성을 가진 요구사항[C]으로 결정한다. 여러 context에 선택적으로 나타나는 PR인 경우에는 그와 대치할 수 있는 PR이 존재하는 지를 확인한다. 그림 4의 ②와 같이, 만약 서로 비슷한 성질을 가진 요구사항들이 발견될 시에는 이를 그룹지어 하나의 일반화된 요구사항으로 만든다. 그림 4에서 PR y_1, PR y_2, PR y_3는 서로 대신할 수 있는 요구사항들이어서 이를 하나의 일반화된 PR y_g로 만들고 'O','X'는 열별로 합친다. 이것은 열별로 합쳐진 'O','X'에 따라 가변성을 지닌 공통성 요구사항[CV]과 가변성을 지닌 선택성 요구사항[PV]으로 나뉜다. 즉, 그림 4에서 CV 속성을 지닌 일반화된 PR y_g는 나타나는 형태가 조금씩 변하긴 하지만, 모든 시스템이 가지고 있는 요구사항임을 의미한다. 또한 그림 4의 ③처럼 다른 요구사항과는 별개의 성질을 가지고 여러 context에서 선택적으로 나타나는 PR인 경우에는 이를 선택성 요구사항[P]으로 인정을 한다. 이 과정을 통해 많은 PR들이 일반화됨으로써 메트릭스의 행의 수가 줄여들게 된다.

이와 같이 무작위로 추출된 PR들을 메트릭스를 사용하여 분석하고 정리하여 그 속성을 구분시키는 과정을 여기서는 도메인 요구사항 일반화 과정이라 일컫는다. 앞 단계에서 도메인 용어에 대한 일반화가 이루어졌다면, 이 단계에서는 요구사항을 기술하는 문장에 대한 일반화가 이루어진 것이다.

4.3 도메인 요구사항 분석 및 모델링

분석단계에서 사용자와 개발자의 의사소통을 위하여 그림으로 생각을 표현하는 도구들이 개발되어 활용되고 있다. 이런 분석 기법들을 제대로 활용하면 의사소통의 많은 문제점을 해결할 수 있다. 본 논문에서는 요구사항

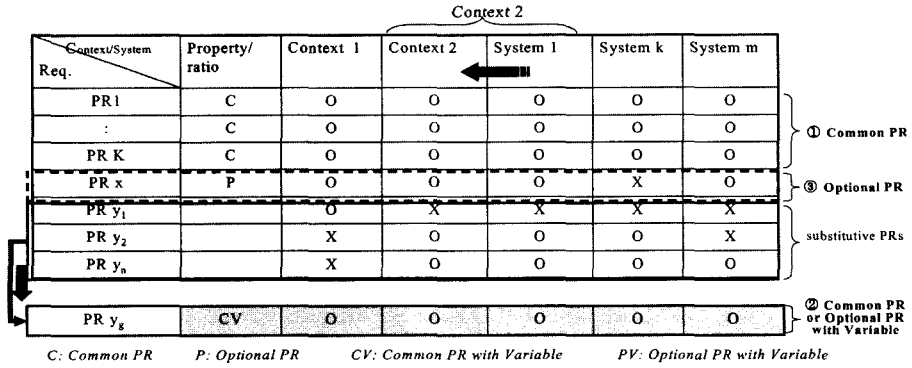


그림 4 PR-Context 메트릭스

을 분석하기 위해 다음과 같은 기준 하에 분석 기법들을 선택하였다. 첫째, 도메인 내부의 기능적인 측면을 워크플로우 지향적인 분석기법을 이용하여 유즈케이스 모델을 만든다. 유즈케이스의 속성을 결정하기 위한 PR-Usecase 매트릭스를 만든다. 둘째, 유즈케이스이 자세한 서술을 위하여 PR-명세서(specification) 형식을 사용하여 기술한다. 셋째, PR-명세서에서 식별된 가변성 지점을 도메인 분석 모델에서 도식화한다. 넷째, PR의 관련성을 분석하여 도메인 제약 명세서를 작성한다.

4.3.1 도메인 유즈케이스 모델(Domain Usecase Model)

도메인 유즈케이스 모델은, 소프트웨어 요구사항을 분석하고 명세하기 위한 여러 방법 중 가장 많이 이용되는 유즈케이스 모델링 방법을 사용하여 만들어진다. 유즈케이스 모델은 시스템의 기능과 환경에 대한 모델이다[25]. 본 논문에서 '도메인 유즈케이스 모델은 도메인의 속성이 표현된 유즈케이스 모델이다'라고 정의한다. 도메인 유즈케이스 모델은 도메인 외부 액터 모델, 도메인 유즈케이스 다이어그램, PR-Usecase 메트릭스로 구성한다. 도메인 유즈케이스 모델은 UML을 사용하여 나타낸다. 그러나 이때, 도메인 속성을 표현하기 위해서는 기존의 UML 요소에 한계가 있음으로 UML의 스테레오 타입 확장 메커니즘을 사용하여 이를 해결 한다. 표 2는 도메인 유즈케이스 모델 작성을 위해 필요한 스테레오 타입을 정리한 것이다.

표 2 도메인 유즈케이스 모델에 대한 UML 확장

common Actor	Actor	<<common>> (default)
optional Actor	Actor	<<optional>>
common Usecase	Usecase	<<common>> (default)
optional Usecase	Usecase	<<optional>>

• 도메인 액터 모델

액터는 시스템의 외부에 존재하면서 시스템과 직접 상호 작용하는 사람 혹은 외부시스템이다. 액터를 고려할 때 중요한 것은 사람이나 직위보다는 '역할'을 먼저 생각해야 한다[26]. 외부 액터 모델은 각각의 액터에 대한 정의와 책임(responsibilities)을 기술한다. 액터들에 대한 관계는 일반화 관계, 집합 관계를 가질 수 있고, 이는 그림 5에서 보는 것과 같이 액터 다이어그램을 통해서 나타낸다.

• 도메인 유즈케이스 다이어그램

유즈케이스는 액터가 어떤 특정한 목적을 달성하기 위해 시스템 내부에서 수행하는 활동의 집합이다. 유즈케이스는 기능적으로 응집력(cohesive)이 있어야 한다[26]. 외부 액터와 유즈케이스, 그들간의 관계를 통해서 도메인의 문맥을 나타내기 위해 최상위 수준의 유즈케이스(top-level usecase) 다이어그램을 그린다. 선택적으로, 다른 유즈케이스와 많은 관계를 가지고 있는 유즈케이스에 대하여 하위 수준의 유즈케이스(low-level usecase)를 그릴 수 있다. 도메인 유즈케이스 다이어그램은 PR-Usecase 메트릭스를 근거로 하여 도메인 유즈케이스 속성을 반영하도록 수정되어진다. 도메인 유즈케이스 다이어그램에는 공통성과 선택성 유즈케이스가 명시된다. 가변적 속성은 공통성 또는 선택성 유즈케이스 내에 포함되어 있는 속성이기 때문에 PR-명세서에서 기술된다. 도메인 속성이 선택성인 경우 스테레오 타입을 이용하여 다이어그램에 표현한다.

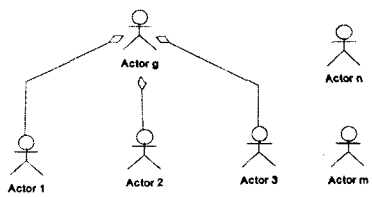
• PR-Usecase 메트릭스

도메인 유즈케이스 다이어그램이 도식화되면, 각각의 유즈케이스의 속성을 결정하기 위해 PR-Usecase 메트릭스를 만든다. 메트릭스에는 유즈케이스 이름, PR, PR 속성들이 표시된다. PR-Usecase 메트릭스는 PR-Context 메트릭스와 유즈케이스를 바탕으로 각각의 유즈케이스가 어떠한 PR로 이루어져있는가를 알 수 있게 해준다. 이를 근거로 유즈케이스를 재구성하도록 해주고, 유즈케이스의 속성을 결정할 수 있도록 해준다. 다음 그림 6은 PR-Usecase 메트릭스의 구축 형태를 보여준다.

PR-Usecase 메트릭스를 분석하여 다음과 같은 경우 유즈케이스를 재구성한다.

(1) 유즈케이스에 포함된 PR이 여러 유즈케이스에 걸쳐 있는 경우이다(그림 6과 7의 ①경우). 이 때는 공통적으로 겹치는 부분을 따로 분리하여 독립된 유즈케이스로 만든 후, 이를 <<include>> 관계로 연관시킨다.

(2) 유즈케이스가 선택성 PR을 포함하고 있는 경우이며(그림 6과 7의 ② 경우), 이 때는 선택성 PR들을 분



Actor name	Actor definition	Responsibilities
User	The actor who uses the digital thermostat to control the temperature of a room	• Change the state of the digital thermostat • Change the desired temperature of the room
Actor 1		
Actor 2		
Actor 3		

그림 5 도메인 액터 모델

Usecase / Req.	Property/ ratio	Usecase1	Usecase2	Usecase3	Usecase...	Usecase_n
PR1	C	O				
PR 2	C					
PR k	C		O		③	
PR m	CV		①		O	
PR n	C			②	O	
:	C			O		④
	P					O
	PV					

C: Common PR CV: Common PR with Variable P: Optional PR PV: Optional PR with Variable

그림 6 PR-Usecase 메트릭스

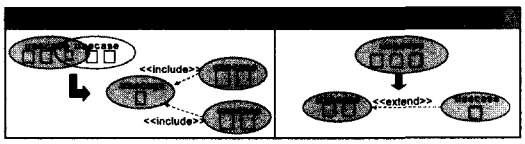


그림 7 PR-Usecase 메트릭스 기초한 유즈케이스 재구성

리하여 <<extend>> 관계로 연관시킨다.

이 과정을 거쳐 재구성된 유즈케이스의 속성은 다음과 같이 분류한다.

공통성 유즈케이스(common use-case) : 유즈케이스가 그 도메인 내에 반드시 있어야 하는 요구사항들 즉, common PR들(C 또는 CV)을 가지고 있는 경우, 공통성 유즈케이스로 분류된다(그림 6의 ③).

선택성 유즈케이스(optional use-case) : 시스템의 프로세스를 처리하는데, 반드시 존재하지 않아도 가능한 유즈케이스를 나타낸다. 이는 선택성 요구사항 즉, optional PR들(P 또는 PV)로 구성된 유즈케이스인 경우이다(그림 6의 ④).

4.3.2 PR-명세서(PR Specifications)

각 Usecase에 대한 상세 서술을 그림 8의 PR-명세서 형식으로 기술한다. 기능적 요구사항의 PR 속성(property)과 상대적 비(ratio)는 PR-Context 메트릭스로부터 결정되었으며 그 결과를 기술한다. PR은 두 가지 측면으로 구성요소들을 나열하게 된다. dynamic 측면은 시간의 흐름에 의해 기술되는 요소들이다. 이것은 세부 활동 중 유즈케이스 명세서의 바탕이 된다. 그러나 모든 시스템이 워크플로우 지향적(workflow oriented)이지 않기 때문에 정적인 측면의 기술도 필요하다. static 측면의 기술에서 정적 구조 관점에서 컴포넌트의 존재 등을 찾아낸다.

두 가지 측면의 PRelement 기술에서 가변성 지점을 식별하게 된다. 그림 9는 도메인 요구사항에서 찾아지는 가변성 종류에 대한 분류를 나타낸 것이다. 먼저 beha-

xx%	
PR1a. [control]은 제어흐름이 가변 될 수 있는 지점을 나타낸다.	*[control]*
PR1b. [c]는 하나의 프로세스가 가변 될 수 있는 지점을 나타낸다.	*[computation]*
PR1c. 특히, [ec]는 그 가변이 외부 서비스에 의해서 처리될 수 있을 경우를 나타낸다.	
PR1x. [d]PR이 다루는 데이터의 가변성 지점을 나타낸다.	*[data]*
PR1x. [I] 최상위 시스템 인터페이스에 나타나는 가변성 지점을 나타낸다.	*[interface]*

그림 8 요구 명세서 안에 기술되는 기능적 요구사항

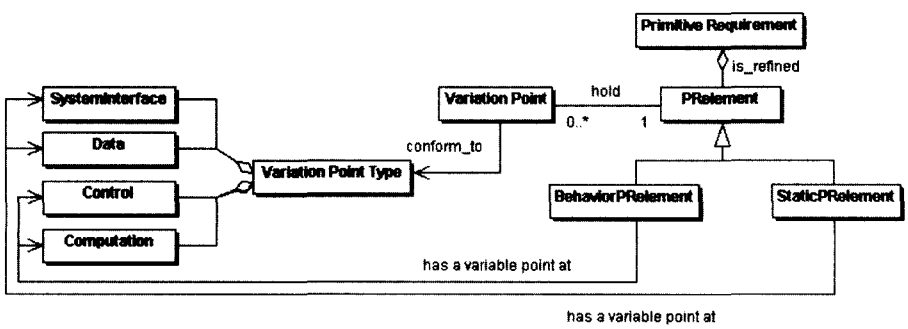


그림 9 도메인 요구사항의 가변성 지점 종류

vior 측면에서는 computation과 control에 대한 가변성을 식별한다. static 측면에서는 data와 system interface에 대한 가변성을 인식한다.

- computation의 가변성은 플로우차트에서 하나의 프로세스에 대한 가변성을 의미하는 것으로 하나의 프로세스가 비즈니스 규칙이나 법규를 포함한다던지, 또는 외부 서비스에 의해 처리되는 경우가 이에 해당한다.
- control 가변성은 플로우차트에서 제어 상태의 가변성을 의미하게 된다. 즉, 선택적 상황에서 판단되는 기준이 가변된다던지, 일정한 흐름의 패턴이 가변되는 경우가 이에 해당한다.
- data의 가변성은 시스템에서 다루는 data의 구조에서 나타나는 가변성을 의미한다.
- system interface의 가변성은 외부 엑터와 시스템과의 상호 작용에서 나타나는 외부 interface의 가변성을 의미한다.

가변성을 분류시켜놓음으로써 요구사항을 관리할 때 나타날 수 있는 변화에 계획적으로 대처할 수 있으며, 이후 단계에서 개발될 아키텍처, 컴포넌트의 가변성에 근거가 되는 값을 제공해 줄 수 있다. 예를 들어, 요구명세서로부터 data 가변성만을 따로 분류해 놓고 보면 이들은 이후, 컴포넌트 개발 시 entity bean에 나타날 수 있는 가변성을 예측할 수 있게 해 준다.

그림 10은 비기능적 요구사항에 대한 명세서의 형식을 보여준다. 비기능적 요구사항은 품질적 요소로 나누어 기술한다. 품질적 요소에는 도메인에서의 우선 순위를 high, medium, low로 나누어 정한다. 일반적으로 품질적 요소는 하나 또는 그이상의 기능적 요구사항과 관련되어 있다[23]. 각각의 품질적 요소는 그와 연결되어 있는 기능적 요구사항을 기술하게 된다. 또한 각각의 품질적 요소가 함께 존재하는 다른 품질적 요소에 의해 어떠한 영향을 받는지를 기술하게 된다.

4.3.3 PR 분석 모델 (PR Analysis Model)

PR 수준에서 식별된 가변성 속성은 도메인 유즈케이스 다이어그램으로 표현되었다. PR 명세서에서 식별된 PRelement 수준의 가변성 지점들은 PR 분석 모델로 도식화된다. 이 모델에서는 3가지 분석 클래스 - boundary 클래스, entity 클래스, control 클래스 -를 이용한다. 이 모델에서 3가지 분석 클래스들은 상위 수준에서 유즈케이스에 참여하는 일련의 객체들을 식별하고 이들이 내포하는 가변성 지점을 나타내기 위한 도구로서 사용된다. PR 분석 모델에서 사용하게 될 UML 구성체와 확장 메커니즘인 스테레오 타입을 정리해 놓은 것이 표 3에 나와 있다.

표 3 PR 분석 모델에 대한 UML 확장

computation VP	control class	<<v.p:C>>
external computation VP	control class	<<v.p:EC>>
control VP	control class	<<v.p:Ctl>>
Data VP	entity class	<<v.p:D>>
System Interface VP	boundary class	<<v.p:I>>

4.3.4 도메인 제약 사항 명세서

도메인 요구사항에 대한 일반화와 분석 과정에서 도메인 요구사항이 PR 단위로 쪼개졌었다. 쪼개진 PR은 기능적인 응집도에 따라 유즈케이스로 묶일 수 있다. 또한 PR은 서로간의 연관 관계를 분석함으로써 더 많은 묶음을 만들어 낼 수 있다. 이는 쪼개어져 있는 PR의 재사용 단위를 높이기 위해 다시 결합시키는 규칙을 만들어 내는 활동을 의미한다. 이는 기능적 PR들 간의 관계를 분석함으로써 관계를 추출할 수 있다. 기능적 PR과 비기능적 요구사항의 관계는 비기능적 요구사항 명세서에서 기술된 것을 바탕으로 명세할 수 있다. 비기능적 요구사항들 간의 관계 또한 비기능적 요구사항 명세서를 통해 찾을 수 있다. 표 4는 식별되는 6가지 도메인 요구사항 관계에 대하여 정리한 것이다.

4.4 도메인 요구사항 관리

요구사항을 관리한다는 것은 요구사항의 변경으로 인해 발생하는 모든 관련 활동들을 다루는 아주 광범위한 의미이다. 본 논문에서는 도메인 요구사항의 변경이 의

	high	Each NFR is related to one or more functions.	+ : has a positive effects
			- : has a negative effects
			± : has no connection with each other NFR
:	medium		
	low		

그림 10 비기능적 요구사항 대한 명세서

표 4 도메인 요구사항들 간의 관계

기능적 요구사항 vs. 기능적 요구사항	
Depend-On ($PR_1 \rightarrow PR_2$)	하나의 PR이 다른 PR을 '요구(require)'하거나 '필요(need)'로 하는 경우이다. 예를 들어, PR_2가 PR_1의 처리 결과를 사용한다던지, PR_1이 끝난 후에만 PR_2가 시작될 수 있다면 PR_1과 PR_2의 관계는 'depend-on' 관계로 규정한다.
Generalization ($PR_g \perp PR_s$)	PR이 하나 이상의 방법으로 인스턴스될 수 있는 경우이다. 예를 들어, PR-Context Matrix에서 substitutive PR_s들로 (PR_{y1}, PR_{y2}, PR_{yn})이 결정이 되고 이들이 하나의 일반화된 generic PR (PR_{yg})로 바뀌었다면, PR_{yg} 와 (PR_{y1}, PR_{y2}, PR_{yn})과의 관계는 "generalization" 관계로 규정한다.
Alternative ($PR_1 \mid PR_2$)	하나의 PR이 다른 PR들과 대체될 수 있는 경우이다. 예를 들어, PR-Context Matrix에서, substitutive PRs 들로 (PR_{y1}, PR_{y2}, PR_{yn})이 결정되었다면, PR_{y1},과 PR_{y2}, PR_{yn} 사이의 관계는 "alternative" 관계로 규정한다.
Refinement ($PR_p \downarrow PR_c$)	하나의 PR은 상세 수준에서 더 세분화 될 수 있다. 예를 들어, PR 명세서에서, PR과 PRelement 와의 관계는 "refinement" 관계로 규정한다.
기능적 요구사항 vs. 비기능적 요구사항	
Is_Effected ($PR_{nfr1} \models PR_{f1}$)	비기능적 요구사항들은 하나 이상의 기능적 요구사항들과 관계를 가지고 있다. 비기능 요구사항 명세서에서 하나의 Quality item과 PR과의 관계가 "is_effected" 관계로 규정한다.
비기능적 요구사항 vs. 비기능적 요구사항	
Conflicting ($PR_{nfr1} \nrightarrow PR_{nfr2}$)	비기능적 요구사항들이 동시에 존재함으로서 부정적 측면을 발생시키는 경우이다. 비기능 요구사항 명세서에서 negative 관계를 가지고 있는 qality item들 사이의 관계를 "conflicting" 관계로 규정한다.

미하는 바를 정리하고 발생할 수 있는 상황, 대처하기 위한 기법에 대하여 설명한다.

4.4.1 변경 관리(Change Management)

먼저, 하나의 시스템 개발 시 발생하는 요구사항 변경과 도메인 요구사항 변경의 차이점에 대하여 논하고자 한다. 그림 11은 시스템을 개발하는 과정에서 요구사항 한 부분의 변경으로 인해 구현되는 시스템의 하나 이상의 부분에 영향을 받는 모습을 보여준다. 요구사항의 변경은 계획되지 않은 것이 대부분이고, 시스템은 이러한 변경을 곧바로 수용할 수 있도록 관리체계가 만들어져야 한다. 즉, 하나의 시스템 개발 시의 변경 관리는 예측하지 못한 변경과 '즉각적'인 수용 체계에 대한 방법을 의미한다.

그에 반해, 도메인 요구사항의 변경 관리는 두 가지 상황으로 나누어 고려해 볼 수 있다. 첫째, 도메인 요구사항이 각 시스템 개발 시 필요로 하는 시스템 개발 요구사항으로 형상화 될 때 발생하는 변경 관리이다(그림 12의 ①). 시스템 개발 요구사항은 도메인 요구사항에서 추출된 공통성 요구사항을 포함할 것이고, 선택성 요구사항을 결정할 것이다. 또한 공통성, 선택성 요구사항이 가지고 있는 가변성 항목이 시스템 특성에 맞게 고정될 것이다. 이것 외, 물리적 요구사항, 디자인 제한 사항, 시스템 고유의 요구사항 등 도메인 요구분석 시에 결정하지 못하는 사항들에 대하여 추가된다. 이러한 도메인 요구사항의 변경은 도메인 요구사항 분석 시 이미 예측되고 계획된 것이다. 두 번째 상황은, 도메인 내 개발되어 나오는 시스템의 변화로 인한 도메인 요구사항의 변경 관리이다(그림 12의 ②). 도메인 요구사항은 기 개발된 시스템을 기반으로 추출된다. 그러나 도메인 내에 속하는 시스템의 개발이 증가함에 따라 그것이 다시 도메인 요구사항을 변화시킬 수 있게 된다. 예를 들어, 도메인 내의 요구사항A라는 것이 모든 시스템에 공통적으로 나타나는 요구사항이었지만, 시간의 흐름에 의해, 또는 기술의 변화로 인해, 또는 다른 외적 환경의 변화로 인해 요구사항A가 더 이상 시스템에 나타나지 않게 된다면 이 기능은 점차적으로 공통적 속성을 잃게 된다. 반대로 새로운 요구사항이 그 도메인의 아주 중요한 기능으로 추가될 수도 있다. 이러한 도메인 요구사항의 변경은 '즉각적'인 것이 아니다. 이것은 서서히 나타나는 변경 요소들이고 이러한 변경 요소들을 계속적으로 관찰하고 이것을 도메인 산출물에 반영시키는 방법을 이용함으로써 변경 관리를 수용할 수 있다.

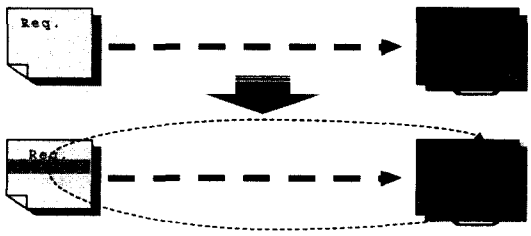


그림 11 요구사항 변경으로 인해 시스템이 영향을 받는 모습

4.4.2 추적 가능성(Traceability)

요구사항 추적가능성이란 요구사항 일생을 앞(for-

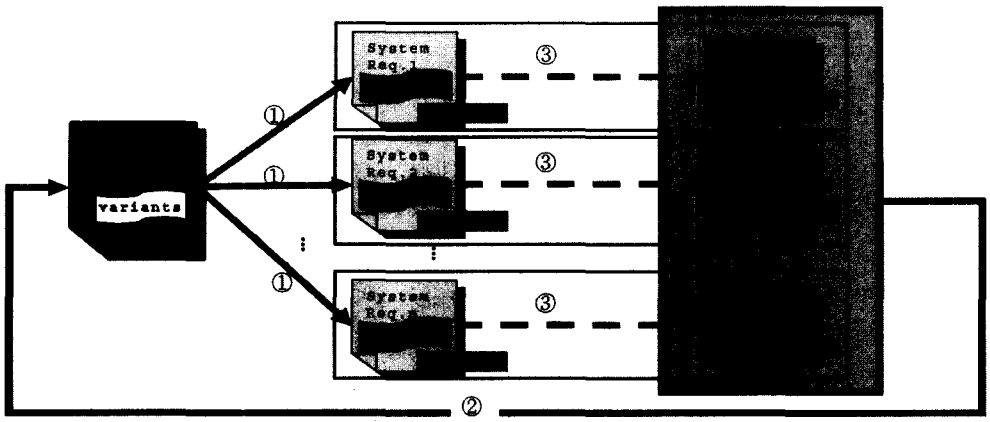


그림 12 도메인 요구사항에서의 변경 관리

ward), 뒤(backward) 방향으로 기술하고 따라갈 수 있는 능력을 말한다[27]. 요구사항 추적가능성은 변화의 영향과 결과를 분석하기 위한 기초자료라는 점에서 요구사항 관리의 핵심이 된다. 도메인 요구사항의 개발과 관리는 요구사항 추적가능성(Requirements traceability)을 향상시켜준다.

그림 12의 ③을 하나씩 독립적으로 본다면 이것은 바로 앞 절에서 설명한 시스템에 영향을 주는 요구사항의 변경을 나타낸다. 즉, 각각을 독립적인 개발 과정으로 본다면 즉각적인 요구사항 변화에 반응할 수 있는 변경 관리 기법을 요구하게 된다. 그러나, 앞뒤 문맥을 살펴보면 도메인 요구사항으로부터 시스템 요구사항이 나온 것을 알 수 있고, 개발된 시스템은 다시 도메인 요구사항에 반영되고 있음을 알 수 있다. 이것은 시스템 개발 시 발생할 수 있는 변경들이 도메인 요구사항 분석 시 이미 예상되어 계획되어있을 확률을 높여준다. 왜냐하면, 도메인 내에서 개발되는 시스템이 가질 수 있는 변경사항들은 또한 도메인 내에 있게 된다. 예를 들어, 시스템 요구사항1에서 발생하는 변경요소1이 시스템1에 곧바로 적용시키는 것이 아니라, 다시 역방향(backward)으로 추적하여 도메인 요구사항을 보게 된다. 도메인 요구사항에서는 이미 계획하고 있었던 변경항목들 중에 변경요소1이 들어있게 된다. 그러면 도메인 요구사항으로부터 개발된 도메인 산출물들을 전방향으로(forward) 추적하여 미리 계획된 변경구조를 확인 할 수 있게 된다. 도메인 문맥에서 개발되는 시스템은 요구사항 추적가능성을 보장받을 수 있으며, 그로 인해 변경 관리의 수행을 용이하게 해 준다.

5. 사례 연구

본 논문에서 제안한 도메인 요구사항 관리 방법을 뉴스 정보 저장소(News Information Repository: NIR) 도메인에 적용하였다. 뉴스 정보 저장소는 웹 기반으로 고객에게 텍스트, 이미지, 동영상 형태의 기사를 제공한다. 이 도메인의 외부 시스템으로는 데이타베이스 시스템, 메일 서비스 시스템, 모바일 서비스 시스템, 빌링 시스템 등이 존재한다. 도메인에 존재하는 대표적인 시스템으로 KBS, MBC, SBS 등 방송국 포털 사이트에 포함되어 있는 뉴스 정보 저장소와 조선일보, 동아일보 등 신문사에서 제공하고 있는 뉴스 정보 저장소가 있었다. 그 외, 몇몇 회사와 학교에서 자체적으로 개발하여 사용하고 있는 뉴스 정보 저장소들이 있다. 본 논문에서는 산재해 있는 뉴스 정보 저장소 요구사항을 분석하여 공통성 요구사항을 식별하고 가변 될 수 있는 요구사항들을 일반화시켜 재사용될 수 있는 핵심 자산으로서 개발하였다. 다음절에서 기술하는 각 개발 요구사항 산출물은 서로 보완적으로 정보를 주고받으며 병행적으로 구축되었다.

5.1 뉴스 정보 저장소 도메인 범위 결정

뉴스 정보 저장소 도메인의 범위를 결정하기 위하여 도메인에서 사용되는 용어들을 일치시키고 정의 내렸다. 또한 이 도메인에 속하는 시스템들이 가지는 공통된 아키텍처 스타일에 대하여 기술하였다. 뉴스 정보 저장소 도메인 내의 시스템들은 데이타 중심적 아키텍처 스타일을 가진다. 그 모습을 그림 13에서 보여준다.

또한 이 도메인은 웹기반 시스템들을 개발하는 것이라고 제한하였기 때문에 이에 대한 논리적 분할 모습과 가능한 한 물리적 분할 배치 모습을 몇 가지 경우로 나누어 그려주었다(그림 14). 실제 도메인 분석 시에는 컴포넌트가 물리적으로 어떻게 배치되는가 하는 것까지는 다루지 않는다. 그러나 논리적으로 어떠한 분할로 나누어질 수 있는가와 부가적으로 이것들이 어떻게 배치될

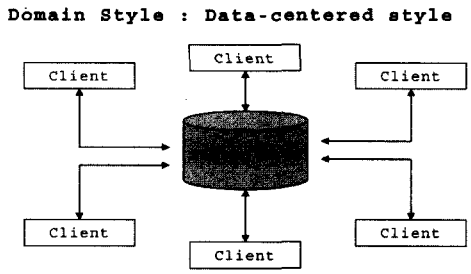


그림 13 뉴스 정보 저장소 도메인 스타일

수 있는가를 최상위 수준에서 설명해 준다면 도메인에 속하는 시스템을 선택할 때 좀 더 용이할 것이다.

5.2 뉴스 정보 저장소 도메인 요구사항 추출 및 일반화

뉴스 정보 저장소의 요구사항을 도메인에 속하는 레거시 시스템들로부터 추출하였다. 표 5는 PR-Context 메트릭스를 구축하여 각 요구사항의 속성을 확률적으로 결정하였고, 가변성이 있는 요구사항들은 일반화시켰다. 로그인, 회원등록, 뉴스검색, 뉴스편집 등은 모든 레거시 시스템에 존재하는 것으로 추출되었기 때문에 공통성 속성으로 판단하였다. 기사를 발송하는 요구사항은 시스템마다 가변성을 가지고 추출되었기 때문에 이를 하나로 묶어 일반화된 형태로 변형시켰다. 이때, 'O'와 'X'를

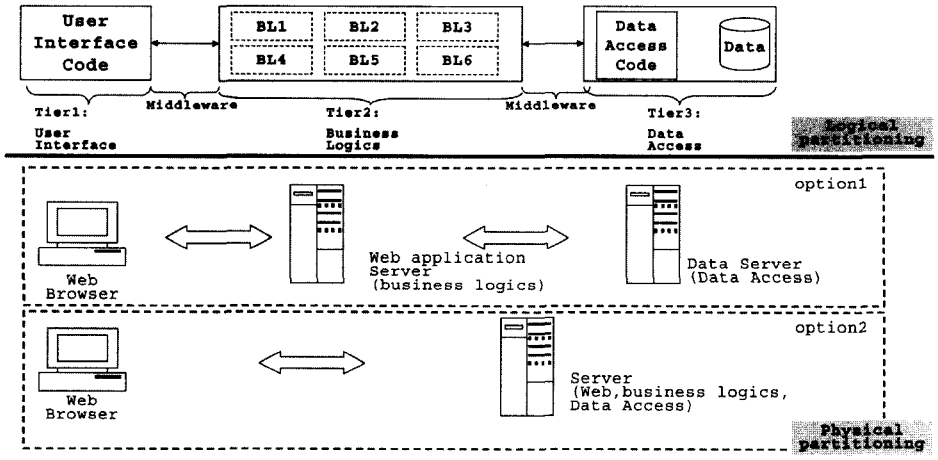


그림 14 웹이용(Web-enable) 도메인의 논리적/물리적 분할

표 5 뉴스 정보 저장소 PR-Context 메트릭스

PR No.	PR		property/ratio	MBC	KBS	SBS	YTN	전자신문	조선일보	중앙일보
PR1	Login		C100%	O	O	O	O	O	O	O
PR2	Logout		C100%	O	O	O	O	O	O	O
PR3	Register		C100%	O	O	O	O	O	O	O
PR4	Modify the member information		C100%	O	O	O	O	O	O	O
PR5	Withdraw		C100%	O	O	O	O	O	O	O
	:		C100%	O	O	O	O	O	O	O
PR7	Write the news		C100%	O	O	O	O	O	O	O
PR8	Search the news		C100%	O	O	O	O	O	O	O
	:									
PR10	Write the opinion		C42.8%	O	O	X	X	X	X	O
PR13	Add the article at scrapbook		P28.5%	X	X	X	X	O	O	X
PR16	Search the scrapbook		P14%	X	X	X	X	X	O	X
PR17	Open the scrapbook to the public		P14%	X	X	X	X	X	O	X
PR27	Send the article		C100%	O	O	O	O	O	O	O
	PR27i1	e-mail to the member		O	O	O	X	O	O	O
	PR27i2	mobile to the member		O	O	O	O	X	X	X
PR33	Sell news contents		P28.5%	X	X	X	O	O	X	X
	PR33i1	the newsletter		X	X	X	X	O	X	X
	PR33i2	the article		X	X	X	O	X	X	X

합집합 시키게 되고 그 결과 이 요구사항은 가변성을 가졌지만은 모든 시스템에 공통적으로 나오는 속성을 가진 것으로 판단할 수 있었다. 반면에 기사 내용을 유료화하여 판매하는 요구사항도 가변적 속성을 가졌지만은, 이것은 도메인 내 시스템에 선별적으로 나타나는 선택적 속성의 가변성이었다.

5.3 뉴스 정보 저장소 도메인 요구사항 분석 및 모델링

뉴스 정보 저장소 요구사항을 분석하여 모델링하였다. 그림 15는 뉴스 정보 저장소의 외부에 존재하면서 도메인을 사용하는 또는 사용되는 액터들에 대한 모델이다. 표 6은 부가적으로 기술되는 액터 기술서이다.

유즈케이스의 속성을 결정하고 유즈케이스를 일반화 형태로 변형하기 위해 PR-Usecase 메트릭스를 구축한다(표 7). SearchNews 유즈케이스에는 선택성 요구사

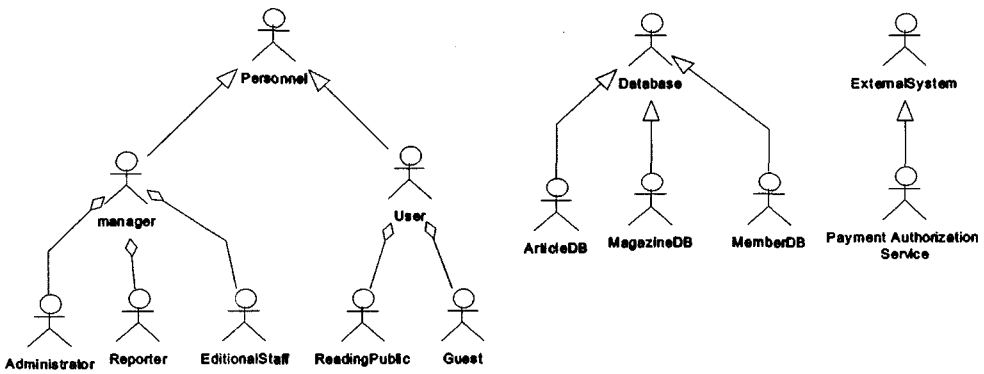


그림 15 뉴스 정보 저장소 액터 모델

표 6 액터 기술서

Reporter	뉴스 정보 저장소에 기사를 입력하는 personnel actor : 기자	기사를 작성하여 저장소에 등록한다. 기사를 수정한다. 기사를 삭제한다.
ReadingPublic	뉴스 정보 저장소 시스템을 사용하는 personnel actor : 독자회원	기사를 검색한다. 특정 기사에 의견을 단다. 기사를 스크랩 한다.

표 7 PR-Usecase 메트릭스

PR No.	PR \ Usecase	Property	Login	Register	SearchNews	...
PR1	Login	C100%	O			
PR2	Logout	C100%	O			
PR3	Register	C100%		O		
PR4	Modify the member information	C100%		O		
PR5	Withdraw	C100%		O		
PR7	Write the news	C100%			O	
PR8	Search the news	C100%			O	
PR9	Show the news	C100%			O	
PR10	Write the opinion				O	
PR11	Modify the opinion				O	
PR12	Delete the opinion				O	
PR13	Add the article at scrapbook				O	
PR14	Delete the article in scrapbook				O	
PR15	Read the article in scrapbook				O	
:	:					
PR27	Send the article					
	PR27i1 e-mail to the member					
	PR27i2 mobile to the member					

항을 가지고 있었고, 이를 4.3.1에 설명한 변형 기준에 따라 분리하였다. 분리된 선택성 요구사항을 추가적으로 도식화하여 그린 유즈케이스 다이어그램이 그림 16에 있다.

표 8은 뉴스 정보 저장소에 대한 요구 사항들 중 '회원등록'에 대하여 PR 명세서를 작성한 것이다. 회원 등

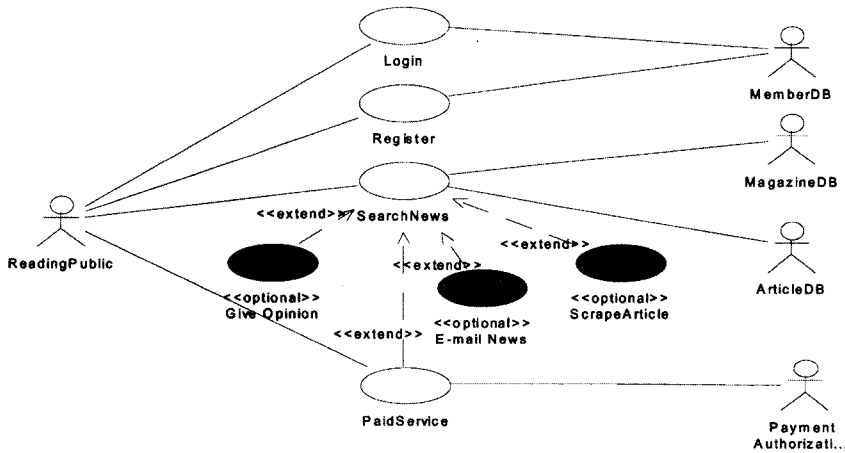


그림 16 선택성 유즈케이스가 표현된 유즈케이스 다이어그램

표 8 뉴스 정보 저장소의 회원등록 요구 명세서

PR No. PR2		
Rules	**PRs**	**Variation Point**
Behavior PRElement	PR3a. [ec]실명확인을 한다. PR3b. [회원 등록 기본자료]를 입력 처리한다. PR3c. [ec]*주소, 우편번호를 (자동)입력시킨다.* PR3d. [회원 등록 부가자료] 입력 처리한다. PR3e. [c]유료 회원의 등급을 처리한다. PR3f. [ec][control]회비를 결재한다.	a[ec] realname checking service c[ec] address searching service e[c] 유료회원 등급 규칙 적용 f[ec] external payment service f[control] 신용카드, 무통장입금, 핸드폰결재, 상품권 결재, 쿠폰 사용, 전자화폐 사용
	b1 [회원 등록 기본자료]가 모두 기록되지 않은 경우 에러메시지를 보낸다. b2. *입력된 아이디가 사용 가능한지를 체크한다.* b3. *비밀 번호가 일치하는지를 검사한다.* b4. 올바른 주민등록번호인지를 확인한다. : f1 신용카드 결재인 경우 f1.1 카드 번호와 유효기간을 입력한다. f1.2 주민등록번호와 비밀번호 앞 두 자리를 입력한다. f1.3 승인 버튼을 눌러 인증을 기다린다. f2. 무통장 입금을 선택한 경우 : f3 핸드폰 결재를 신청한 경우 f3.1 핸드폰 번호와 주민등록 번호를 입력한다. f3.2 승인 번호를 입력한다.	f1[ec] payment authorization service system f3[ec] handphone payment authorization service system
Static PRElement	회원등록 기본자료	[아이디/비밀번호/이름/주민등록번호/주소/우편번호/전화번호1/전화번호2/e-mail주소]
	회원등록 부가자료	[직업/취미/수입 등]
	MemberProfileDB object 생성	회원등록 기본자료 + 회원등록 부가자료, 시스템 사용 회원 정보
	회원등록 interface	
	결재 시스템 interface	

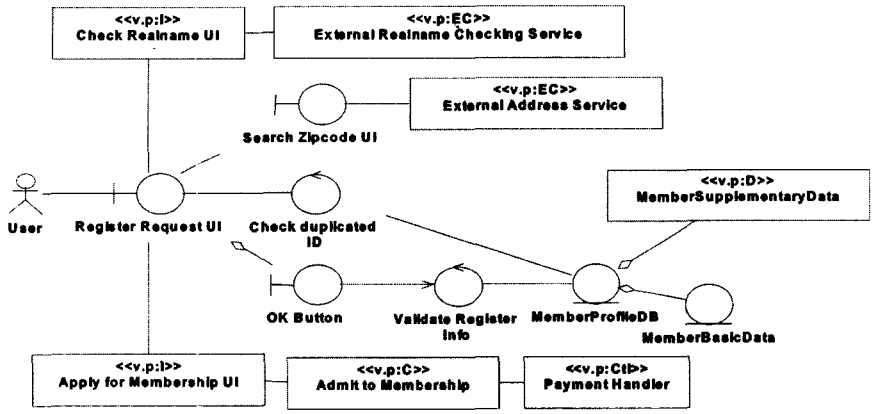


그림 17 *PR1:Register*에 대한 PR 분석 모델

록 요구사항은 PR-Context 메트릭스에 의해 도메인에서 아주 높은 공통성 속성을 가진 것으로 확인된다. 회원 등록에 대한 요구사항의 behavior PRelement를 기술하면서 주소 입력은 외부 서비스를 이용하여 처리할 수 있음을 [ec](external computation)로 명시하였고, 유료 회원 등급을 적용하는 요구사항은 가변성이 있는 항목이므로 [c](computation의 가변성)을 표시하였다. 유료 회원의 회비를 결재하는 방식은 여러 가지 선택사항이 있을 수 있으므로 이 시점에 [controll을 명시하여 제어의 가변성이 발생할 수 있음을 표현하였다. static PRelement에는 회원등록과 관련하여 찾아지는 회원등록 기본자료, 회원등록 부가자료, 이 자료들을 다루는 MemberProfileDB, 회원 등록 외부 인터페이스 등을 식별하였고 발생할 수 있는 가변성을 기술하였다.

회원 등록 PR 명세서에 대한 PR 분석 모델이 그림 17에서 보여준다. 이 다이어그램을 통하여 회원 등록 시 구성되는 인터페이스를 확인할 수 있고, 데이타의 흐름, 필요로 하는 외부 시스템 - 실명확인 서비스(Realname Checking Service)와 주소서비스(External Address Service) - 과 데이타베이스들을 알 수 있다. 그림 18은 뉴스 정보 저장소 도메인 제약 사항 명세서의 일부분을 보여준다. 이 명세서를 통해, 회원 등록 후, 로그인을 할 수 있고, 로그인 후, 뉴스를 작성하거나, 회원 정보를 수정, 회원 탈퇴, 로그 아웃 등을 할 수 있음을 알 수 있다. 또한 하나의 일반화 된 PR(PR_{3f}, PR_{27})이 여러 개로 인스턴스 됨을 알 수 있으며, 이때, 반드시 하나의 인스턴스만이 선택되어야 하는 경우는 1로 명시되었고, 여러 개가 복합적으로 선택 가능한 경우는 +로 명시할 수 있다. 하나의 비기능적 요구사항(PR_{nfr1} 또는 PR_{nfr2})은 이와 연관된 기능적 요구사항들(PR_1, PR_2, PR_3)을 기술할 수 있으며, 이때, 서로 conflicting 관계를 가지고 있는

$PR_3 \rightarrow PR_1$
$PR_3 \downarrow\{[PR_{3a}] + PR_{3b} + PR_{3c} + PR_{3d} + [PR_{3e} + PR_{3f}]\}, PR_{3e} \rightarrow PR_{3f}$
$PR_1 \rightarrow PR_2, PR_1 \rightarrow PR_7 \ldots$
$PR_{3f} \hookrightarrow (PR_{3f1} \mid PR_{3f2} \mid PR_{3f3} \mid PR_{3f4} \mid PR_{3f5} \mid PR_{3f6})^+$
$PR_{27} \hookrightarrow (PR_{27f1} \mid PR_{27f2})$
:
$PR_{9a} \hookrightarrow (PR_{9a1f} \mid PR_{9a2})^+$
$PR_{9c} \hookrightarrow (PR_{9c1f} \mid PR_{9c2})$
$PR_{nf1} \ominus (PR_1, PR_2, PR_3)$
$PR_{nf2} \ominus (PR_1, PR_3, PR_5)$
$PR_{nfr1} \nleftrightarrow PR_{nfr2} \Rightarrow (PR_1, PR_2, PR_3) \nleftrightarrow (PR_1, PR_3, PR_5)$
$\Rightarrow$ Quality (nf1,nf2)-decision-point: PR_1, PR_3

그림 18 뉴스 정보 저장소 도메인 제약 사항 명세서

비기능적 요구사항들(PR_{nfr1}과 PR_{nfr2}) 사이에는 서로 겹치는 기능적 요구사항(PR_1, PR_3)을 식별할 수 있게 된다. 여기서 식별된 기능적 요구사항이 비기능적 요구사항을 결정하는 설계 결정 변수 지점임을 알 수 있게 해준다.

5.4 평가

본 절에서 뉴스 정보 저장소 도메인의 범위를 한정하고, 도메인에 속한 요구사항들의 속성을 객관적인 지표를 만들어 분석하고, 이를 여러 측면 모델링한 모습을 보여주었다. 이를 통해 다음과 같은 사항들을 입증하였다.

- 본 연구를 통해 요구사항의 공통성과 선택성 속성을 객관적으로 판단할 수 있는 지표를 제공하였다.
- 본 연구를 통해 요구사항 수준에서 공통성의 범위를 식별하고 가변성의 수준을 달리하여 그 속성과 가변성 지점을 명확히 추출하였다.
- 본 연구가 실제 프로젝트에 적용될 수 있었다.

6. 결론 및 향후 연구 과제

본 논문에서는 프로덕트 라인에서 핵심 자산중 하나인 도메인 요구사항을 체계적으로 관리하는 방법을 제안하였다. Time-to-Market을 중요시 여기는 오늘날, 프로덕트 라인 공학과 요구공학 분야는 각각 중요한 연구의

한 분야로서 연구가 활발히 진행중이다. 그러나, 두 연구 분야의 결합은 아주 부족한 상태이다. 시스템 요구사항들도 프로덕트 라인 안에서 재사용될 수 있는 핵심 자산이고, 일정한 범위의 도메인 내에서 요구사항들을 개발, 분석, 관리한다면 요구사항의 재사용은 향상될 것이다. 본 논문에서는 먼저, 도메인 요구사항을 관리하기 위해 필요로 하는 많은 산출물들 사이에 일치된 개념을 사용하고 연관성을 높이기 위해 도메인 요구사항 메타모델을 제시하였다. 도메인 요구사항 관리는 요구사항 범위결정, 요구사항 추출과 일반화, 요구사항 분석과 모델링, 요구사항 변경 관리로 세분화시켜 수행하였다. 각 세부활동에서는 도메인의 골격을 제공하는 공통성을 찾는 데에 중점을 두었고, 공통된 골격에 변화의 살을 붙이는 가변성에 대하여 명확하게 표현하려고 노력하였다.

향후 연구 과제는 프로덕트 라인에서 또 다른 핵심자산이 될 수 있는 도메인 아키텍처를 개발 관리하는 방법에 대하여 연구한다. 도메인 아키텍처는 도메인 요구사항과 관계가 잘 정립되어 있어야 하고 이 또한 공통성과 선택성, 가변성의 속성을 반영하여 명시적으로 나타낼 수 있어야 한다. 핵심 자산으로서 도메인 산출물을 잘 관리함으로써 체계적인 재사용을 보장해 주게 되고, 이는 결과적으로 시스템 개발 시간과 비용을 줄여주는 효과를 가져다준다.

참 고 문 헌

[1] Muthig, D., Atkinson C., "Model-Driven Product Line Architecture," G. Chastek, editor, Software Product Lines: In Proceedings of the Second Software Product Line Conference (SPLC2), San Diego, U.S.A., Aug. 2002, Heidelberg, Germany: Springer Lecture Notes in Computer Science Vol. 2379, 2002, pp.110-129.

[2] Faulk, S. R, "Product-line requirements specification (PRS): an approach and case study," In Proceeding Fifth IEEE International Symposium on Requirements Engineering, 2001, pp.48-55.

[3] Kang, K. C., "Feature-Oriented Domain Analysis for Software Reuse," Joint Conference on Software Engineering, 1993, pp.389-395.

[4] SEI in Carnegie Mellon University, "Feature-Oriented Domain Analysis," URL:http://www.sei.cmu.edu/str/descriptions/foda_body.html

[5] Griss, M. L., Favaro, J., and d'Alessandro, M., "Integrating Feature Modeling with the RSEB," In Proceedings of 5th International Conference on Software Reuse, Victoria Canada, June, IEEE, 1998, pp.76-85.

[6] van Gurp, J., Bosch, J., and Svahnberg, M., "On the notion of variability in software product lines," Proceedings on Working IEEE/IFIP Conference on Software Architecture, 2001, pp.45-54.

[7] van Deursen, A., de Jonge, M., and Kuipers, T., "Feature-Based Product Line Instantiation Using Source-Level Packages," the Proceedings of Second Product Line Conference (SPLC2), 2002, pp.217-234.

[8] Frakes, W., Prieto-Diaz, R., and Fox, C., "DARE-COTS: A Domain Analysis Support Tool," Proceedings on XVII International Conference of the Chilean Computer Science Society (Valparaiso, Nov, 1997), pp. 73-77, 1997.

[9] Mannion, M., "Using First-Order Logic for Product Line Model Validation," In Proceedings of Second Product Line Conference (SPLC2), 2002, pp.176-187.

[10] Keepence, B., Mannion, M., "Using patterns to model variability in product families," IEEE Software, Vol: 16, Issue: 4, 1999, pp.102-108.

[11] Clauß, M., "Generic Modeling using UML extensions for variability," OOPSLA 2001 Workshop on Domain Specific Visual Languages, 2001.

[12] Webber, D., Gomaa, H., "Modeling Variability with the Variation Point Model," In Proceedings of the Seventh International Conference on Software Reuse (ICSR 7), 2002, pp.109-122.

[13] I. Sommerville and G. Kotonya, Requirements Engineering: Processes and Techniques, John Wiley & Son Ltd. 1998.

[14] McPhee, C., Eberlein, A., "Requirements Engineering for Time-to-Market Projects," In Proceedings of the Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems, April 2000, pp.17-24.

[15] Clements, P., Northrop, L., Software Product Lines: Practices and Patterns, Addison Wesley, 2001.

[16] Kuusela, J., Savolainen, J., "Requirements Engineering for Product Families," In Proceedings of the Twenty-Second International Conference on Software Engineering (ICSE'00), Limeric, Ireland, June 2000. pp.60-68.

[17] Thompson, J.M., Heimdahl, M.P.E., "Structuring product family requirements for n-dimensional and hierarchical product lines," Requirements Engineering, 8:42-54, April 2003.

[18] Leffingwell, D., Widrig, D., Managing Software Requirements, A Unified Approach, Addison-Wesley, 2000.

[19] Bass, L., Clements. P., Donohoe, P., McGregor, J., Northrop, L., "Fourth Product Line Practice Workshop Report," Software Engineering Institute, USA, Nov. 1999.

[20] Berard, E., Essays in Object-Oriented Software Engineering, Prentice Hall, 1992.

[21] Streifferdt, D., "Traceability for System Families," Software Engineering, In Proceedings of the 23rd International Conference on ICSE, May 2001,

pp.803-804.
[22] Davis, A., Software Requirements :Objects, Functions and States, Englewood Cliffs, N.J.:Prentice Hall, 1993.
[23] Kotonya, G., Sommerville, I., "Requirements Engineering with Viewpoints," Software Engineering Journal, Volume: 11 Issue: 1, Jan. 1996, pp.5-18.
[24] Digre T., "Business Object Component Architecture," IEEE software Vol.15, No.5, September/ October, 1998, pp.60-69.
[25] Larman, C., Applying UML and Patterns 2/E: An Introduction to Object-Oriented Analysis and Design and the Unified Process, Prentice Hall. 2002.
[26] Firesmith, D. G., "Use Case Modeling Guidelines," In Proceedings of Technology of Object-Oriented Languages and Systems, TOOLS 30., Aug. 1999, pp.184-193.
[27] Gotel, O., 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.

문 미 경

1990년 2월 이화여자대학교 전자계산학과(학사). 1992년 2월 이화여자대학교 전자계산학과(석사). 2002년 2월 부산대학교 컴퓨터공학과 박사 수료. 1998년 3월~1999년 2월 안산1대학 사무자동학과 겸임교수. 관심분야는 도메인 공학, 프로덕트 라인 공학, 컴포넌트 기반 소프트웨어 개발, 도메인 분석 및 아키텍쳐 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, 객체지향 소프트웨어 개발방법론, 요구 검증 등임