Journal of Korea Multimedia Society Vol. 8, No. 3, March 2005(pp. 398-412)

레거시로부터 CBD로의 재공학을 위한 메타 모델 설계 기법

조은숙[†], 김철진[††]

요 약

레거시 시스템을 새로운 하드웨어 플랫폼과 새로운 소프트웨어 개발 패러다임에 맞도록 이주하려는데 대한 관심이 증가하고 있다. 그 이유는 레거시 시스템의 유지보수에 많은 비용이 들고 기존 시스템에 대한 문서 정보가 부족하다는데 있다. 레거시 시스템을 새로운 시스템을 변환하기 위해 스크린 스크래핑, 래핑, 부분 변환, 재개발 등등의 여러 다양한 접근법들과 도구, 그리고 방법론들이 제시되어왔다. 그러나 이러한 접근법들은 대부분이 코드 변환 혹은 일부 모델 변환 수준이기 때문에 아키텍처나 요구사항 수준에서의 변환까지는 체계적으로 제시하지 못하였다. 따라서, 본 논문에서는 이러한 한계점을 극복하기 위하여 아키텍처와 요구사항 단계까지 적용할 수 있는 3차원 공간 개념을 적용한 메타 모델 기반의 접근법을 제시하고자 한다. 이러한 통합 모델은 재공학에 있어서 역공학 단계인 코드에서 아키텍처 그리고 순공학 단계인 아키텍처에서 코드로의 자연스러운 변환 혹은 협력 진화(Co-evoluton)를 유도한다.

A Design Technique of Meta-Model for Reengineering from Legacy to CBD

Eun-Sook Cho[†], Chul-Jin Kim[††]

ABSTRACT

There is an increasing interest in migration legacy systems to new hardware platforms and to new software development paradigms. The reason is that high maintenance costs and lack of documentation. In order to migrate or transform legacy system, various approaches such as screen scrapping, wrapping, semi-development, and re-development, tools, and methodologies are introduced until now. However, architecture or requirements level's transformation is not suggested because most of those approaches focus on code-level transformation or a few model-level transform. In this paper, we suggest a meta-model driven approach applying 3D space concept, which can be applied into architecture and requirement phase. Proposed integrated model drives seamless migration or co-evolution from code to architecture of reverse engineering and from architecture to code of forward engineering.

Key words: Legacy System(레거시 시스템), Reverse Engineering(역공학), Re-Engineering(재공학), Meta-Model(메타 모델)

※ 교신저자(Corresponding Author) : 조은숙, 주소 : 서울특별시 성북구 하월곡2동(136-714), 전화 : 02)940-4595, FAX : 02)940-4194, E-mail : escho@dongduk.ac.kr
접수일 : 2004년 8월 13일, 완료일 : 2004년 9월 23일
[†] 정회원, 동덕여대 정보학부 데이터정보전공 전임강사
[††] 가톨릭대학교 컴퓨터 정보 공학부 초빙교수
(E-mail : cjkim@otlab.ssu.ac.kr)
※ 이 연구는 한국학술진흥재단의 신진교수 연구과제(D00396) 지원으로 수행되었음.

1. 서 론

레거시 시스템을 컴포넌트 기반 시스템으로 변환하기 위한 지금까지의 접근법은 주로 래핑(Wrapping) 혹은 스크린 스크래핑(Screen Scrapping) 위주의 재공학 기법들이다[1-3]. 이와 같은 방법들은 주로 사용자 인터페이스 부분을 주로 변환하는 접근

법들이다. 따라서 시스템의 비즈니스 로직이나 데이터 부분은 여전히 레거시 시스템으로 운영되고 있는 형태이다. 이와 같은 형태의 변환 전략은 레거시 시스템을 컴포넌트 기반 혹은 웹 기반 등의 새로운 형태의 시스템으로 변환한다고 하기에는 여러가지로 미약한 부분들이 많이 존재한다. 그 이유는 시스템의 주요 부분인 비즈니스 로직이나 데이터에 대한 변환이 전혀 이루어지지 않기 때문이다.

또한 국내의 경우에 있어서 레거시 시스템을 변환하는 사례들을 살펴보면, 레거시 코드를 목표 시스템의 코드로 일 대 일 매핑하여 변환하는 경우가 비교적 많은 편이다. 이처럼 코드에서 코드로 매핑해서 변환하게 되면, 여러가지 문제점이 발생된다. 예를 들어 COBOL과 같은 절차적 언어로 구축 된 레거시 시스템을 Java나 EJB로 구축될 목표 시스템으로 변환하는 경우에 있어서, 이와 같은 경우에는 일 대 일 매핑이 완전하게 이루어질 수 없다. 그 이유는 서로 지향하는 패러다임이 다르기 때문이다. 따라서 이와 같이 레거시 코드를 단순하게 새로운 목표 언어의 문법에 맞춘 코드로의 변환은 시스템 변환에 있어서 기존 레거시 시스템에 새로운 요구사항을 반영하여 변환할 수도 없을 뿐만 아니라 새로운 패러다임에 맞도록 아키텍처나 기타 설계들이 이루어지지 않은 상태에서 변환하였기 때문에 변환된 시스템이 품질이나 유지보수성이 향상될 수가 없다.

이와 같은 접근 방법들을 적용하는 경우에 있어서는 비록 새로운 시스템이 구축되었다고 할지라도 그 시스템은 여전히 이전의 레거시 시스템이 안고 있는 복잡도나 유지보수 문제 등의 부담을 그대로 안고 있다고 볼 수 있으며, 품질 향상에 있어서도 그 효과가 기대하기 어렵다.

따라서, 본 논문에서는 이러한 기존의 접근 방법들의 문제점들을 인식하여 보다 체계적이면서 기존 시스템에 새로운 요구사항을 반영하여 보다 나은 차세대 컴포넌트 기반 시스템으로의 변환을 지원하기 위해 모델 기반의 재공학 접근 기법을 제시하고자 한다.

본 논문은 다음과 같이 구성된다. 2장에서는 지금까지 연구되어 온 재공학 접근 방법들을 소개하고, 각각의 방법들의 특징과 또한 한계점들을 제시한다. 3장에서는 본 논문에서 제시하고자 하는 메타 모델의 기초가 되는 3차원 공간 개념을 설명한다. 4장에서는 모델 기반의 재공학 프로세스의 개념과 메타 모델의 전체 아키텍처와 메타 모델들 간의 관계들을 정의한다. 5장에서는 본 논문에서 제안한 메타 모델을 기반으로 재공학에 적용한 사례를 설명한다. 5장에서는 기존의 연구와 본 논문에서 제시한 모델 기반의 재공학 접근 기법과의 비교 연구를 통한 평가 내용을 제시한다. 마지막으로 6장에서 결론 및 향후 연구과제를 제시한다.

2. 관련 연구

레거시 시스템을 새로운 플랫폼에 맞추어 변환하는 접근법에 대한 연구는 다양하다. 한가지 방법은 새로운 요구사항을 가지고 레거시 시스템 전체를 처음부터 다시 재개발하는 방법이다. 이 방법은 요구사항 명세, 설계, 그리고 구현 등이 좋은 경험과 기술을 가지고 모두 새롭게 출발할 수 있다는 장점을 갖는 반면에, 많은 비용과 시간 소요, 높은 위험도, 기존 레거시 시스템에 대한 문서의 부재로 인한 레거시 시스템에 대한 지식과 비즈니스 규칙들을 이해하는데 걸리는 노력 등의 단점들을 가지고 있다. 여러 연구들에서 이 방법은 바람직한 방법이 아님을 주장하고 있다[4]. 따라서 여러 연구들에서는 점진적인 변환이 바람직한 방법으로 소개하고 있으며, 이와 같은 방법을 산업체에서도 적용하고 있다.

점진적인 변환 방법은 먼저 현존 코드에 역공학 기법을 적용해서 설계나 아키텍처 모델들을 복구한다. 그런 후에, 추출된 정보들을 기반으로 순공학 기법을 적용하여 새로운 목표 시스템으로 변환하는 것이다. 이때 두가지 접근 방법을 적용할 수 있다. 하나는 새로운 요구 사항들을 반영하여 새로운 소프트웨어 시스템을 개발하는 접근이고, 또 다른 하나는 기존 레거시 시스템의 컴포넌트들을 재사용하여 변환하는 것이다[5].

이 장에서는 현재까지 연구되어 온 여러 가지 재공학 접근 방법들에 대해서 소개하고, 기존 방법들에 있어서의 한계점을 제시한다.

2.1 재공학 접근 방법

Sneed와 Majnar[1]는 레거시 시스템을 객체 지향의 클라이언트/서버 시스템으로 변환하는 방법과 그 경험을 보고하고 있다. 이 방법에서는 작업(job), 트랜잭션(transaction), 프로그램(program), 모듈(module), 그리고 프로시저(procedure) 등과 같은 요소

들에 대해 캡슐화 시킨 래핑 기법을 이용하여 변환하고 있다[5]. De Lucia[3]도 레거시 시스템을 객체지향 플랫폼으로 변환하는 접근 기법을 제시하고 있는데 이들은 역공학 단계에서 객체 식별을 위한 기법을 사용하고 있다. 그런 후에 식별된 객체들을 래퍼들로 캡슐화 하여 변환하고 있다.

1.2 기존 연구의 한계점

소프트웨어 재사용에 대한 관심이 높아지면서 재공학 방법론에 있어서 이전보다 체계적이면서 다양한 접근법들이 소개되고 있다. 그러나 이러한 접근법들은 아직까지도 몇가지 한계점들을 가지고 있다.

첫째, 현존하는 재공학방법론들은 재공학 과정 동안에 많은 모델들을 만들어 내고 적용하지만 모델들간의 연결고리가 제대로 설정되어 있지 못한 경우가 있다. 이로 인해 과다한 문서의 도출로 인한 부담과 시스템 변환 과정에 있어서 이러한 모델들이 실질적으로 반영되지 못하는 경우가 발생하고 있다. 이는 결국 레거시 시스템의 자원에 대한 재사용성이 저하되는 결과 또한 초래하고 있다.

둘째, 방법론 마다 모델들이 동일한 의미를 내포하면서 서로 다르게 표현하거나 변환 과정에서 사용되지 않는 불필요한 정보들이 추출되는 경우가 존재한다. 이는 모델이 갖는 표현 수준이나 추상화 정도를 제대로 반영하지 못하는 결과를 야기한다. 따라서 특정 단계에서 필요로 하는 정보가 표현되지 못하는 경우가 존재하기도 하고 그로 인하여 다른 모델 개발에도 영향을 주기도 한다. 본 논문에서는 이 문제점을 해결하기 위하여 3차원 공간 개념을 적용하여 단계별 혹은 표현 수준에 따른 모델들로 계층화하여 제시한다.

셋째, 방법론마다 추출되거나 적용되는 모델들이 어느 한쪽 측면만을 강조하는 경우가 있다. 이로 인해 새로운 목표 시스템으로의 변환에 있어서 아키텍처의 구성이나 시스템의 다른 부분들을 표현하는 데 있어서 어려움을 초래한다. 본 논문에서는 이러한 문제점을 구조적, 행위적, 기능적 측면에서 표현할 수 있는 다양한 모델들에 대한 정보를 메타 모델로 제시한다.

넷째, 방법론마다 여러모델들이 추출되고 개발되는데 모델들의 추상화 레벨이 다르고 다양하다. 당연히 시스템 개발에 있어서도 추상화 레벨에 따라 다양한 모델들이 개발될 수 있다. 그러나 기존 방법론들에서는 이러한 레벨에 따른 모델들 간의 아키텍처가 제대로 형성되어 있지 못하다. 본 연구에서는 이를 통합하여 모델들의 추상화 레벨에 따라 아키텍처를 구축하여 제시한다.

3. 3차원 소프웨어 공간

이 장에서는 소프트웨어 개발에 있어서 모델 기반의 소프트웨어를 3차원 공간을 통하여 표현하는 방법을 제시한다. 여기서 말하는 3차원은 다음과 같다. 1차원(D1)은 메타 레벨에 대한 차원(Meta Dimension)을 의미하고, 2차원(D2)은 소프트웨어 공학 차원을 말한다. 3차원(D3)는 산출물의 표현 혹은 묘사의 정도를 나타내는 차원을 의미한다.

3.1 1차원 메타 피라미드

1차원 메타 피라미드가 그림 1에서 제시되고 있다. 각 레벨 별로 몇가지 예제들이 표현되고 있다. 이 메타 피라미드에서 가장 명확한 레벨은 모델 레벨인 M1이다. 이것은 일반적으로 프로그램들이 실행되는 레벨이다. 이 레벨을 어플리웨어(Appliware)라고도 한다. 이 레벨에 속하는 개체들은 특정 어플리케이션 도메인 즉, 금융, 원자력, 병원 등에 종속되게 된다. 예를 들어 계정 혹은 고객과 같은 개념들은 금융 모델의 요소이다. 반면에 진료나 환자 등은 병원 모델에 요소이다.

반대로 메타웨어는 특정 어플리케이션 도메인에 독립적이다. 메타 모델 레벨 M2는 소프트웨어 어플리케이션들의 생산을 관리하기 위해 사용된다. 따라서M2에서는 클래스, 메소드, 모듈, 프레임워크, 동적 라이브러리 등과 같은 모든 소프트웨어 공학 개념들을 기술해야 한다. 본 논문에서 제시하는 재공학 메타 모델에서도 재공학에 관련된 개념들을 여기서 정의하게 된다.

1차원 메타 피라미드의 최고점인 M3는 메타 메타 모델 레벨이다 이는 메타 모델들이 어떻게 기술되어야 하고, 관리되어야 하는지를 기술한다. 예를 들어 MDA 표준은 MOF(Meta Object Facilities)[6]를 사용하도록 제안하고 있다. MOF는 UML 메타 모델뿐만 아니라 독단적으로 소프트웨어 메타 모델들을 기술할 수 있도록 하는 장치이다. 따라서, 메타 모델

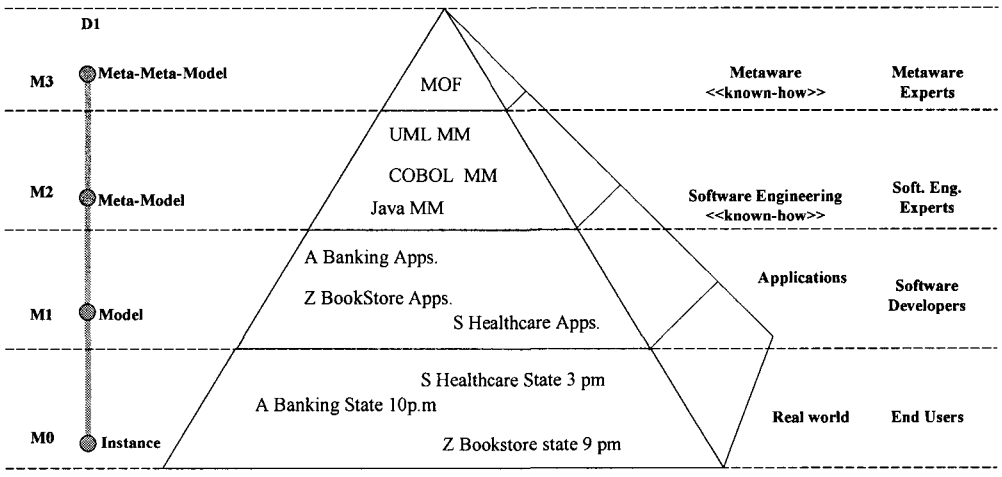

그림 1. 1차원: 메타 피라미드(D1)

을 제시하였을 경우, 이 메타 모델에 MOF에 맞게 정의되었는지를 증명함으로써 제시한 메타 모델의 정당성을 증명할 수 있다.

3.2 2차원 공학 피라미드

이 차원은 소프트웨어 개발 생명 주기의 각 단계에서 산출되는 산출물들을 기반으로 개발할 소프트웨어를 구조하는데 그 목표를 두고 있다. 예를 들면, 요구사항 정의서와 아키텍처 문서 그리고 구현 산출물들 간의 관계와 구별을 명확하게 한다. 여기서는 생명 주기를 크게 4가지 단계인 요구사항 분석, 아키텍처 정의, 설계, 그리고 구현 단계로 구분하여 정의한다. 실제 소프트웨어 개발 단계들을 단순화 시켜서 4단계로 정의한 것이다. 그림 2에 나타난 것처럼 공학 피라미드는 생명 주기의 4단계와 각 단계별로 개발할 산출물들의 모습을 예로 보여주고 있다.

3.3 3차원 묘사 피라미드

소프트웨어 개발 혹은 재공학 과정에서 생산되는 산출물들의 정보는 묵시적인 표현에서부터 매우 구체적인 표현에 이르기 까지 다양한 방법으로 표현될 수 있다. 요구사항 정의 단계에서는 산출물들에서 표

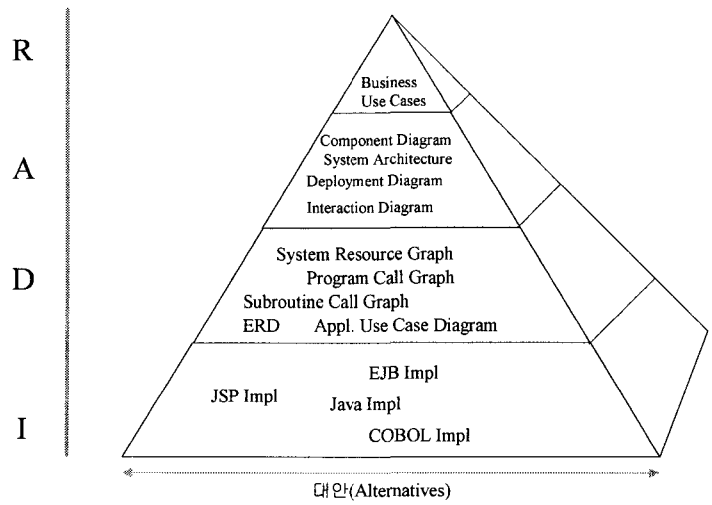

그림 2. 2차원: 공학 피라미드

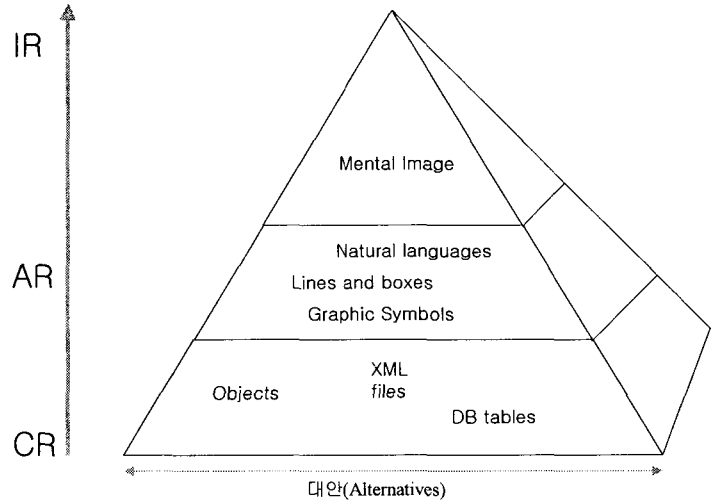

그림 3. 3차원: 묘사 피라미드

현하는 정보가 보다 추상적으로 표현되지만, 구현 단계에서는 산출물들의 표현 정보가 매우 구체적으로 표현되게 된다. 이처럼 이 3차원 피라미드를 통해 재공학 과정에서 개발 되는 산출물들을 추상화 수준에 따라 구별하여 표현함으로써 많은 산출물들이 어떠한 정보를 표현하며, 그 정보의 수준이 어느 수준인지를 구별하여 표현할 수 있게 된다. 이는 표현 정보에 있어서 산출물들 간의 관련성을 제공해 줄 뿐만 아니라 개발되는 산출물들의 각 단계별로 표현해야 할 표현의 수준을 잘 나타내고 있는지를 파악할 수 있도록 해 준다. 그림 3은 3차원 묘사 피라미드를 보여주고 있다. 묘사의 수준은 다양하겠지만, 여기서는 간단히 3 단계로 구별하여 표현한다.

4. CBD로의 재공학을 위한 메타 모델

재공학 방법론마다 재공학 프로세스의 단계들을 다르게 나누어 정의할 수 있다. 그러나 이는 프로세스 내의 단계들의 굵기(Granularity)의 문제이지 표현하는 내용들은 거의 공통적인 내용들을 담고 있다. 따라서 본 논문에서는 기존의 재공학 프로세스들을 기반으로 하여 보편적인 형태의 재공학 프로세스 모델을 정의하여 이를 기반으로 각각의 단계에서 추출되는 모델들에 대한 메타 모델들을 제시하고자 한다.

4.1 재공학 메타 모델

이 절에서는 재공학 과정 동안에 적용되는 모델들에 메타 모델을 3.1절에서 제시한 3차원 피라미드를 기반으로 제시한다. 먼저 전체 재공학 프로세스에 적용되는 모델들을 표현하기 위해 2차원 피라미드를 이용하여 각 단계별 적용되는 모델들과 모델들 간의 관련성을 메타 모델로 표현하다. 그리고 나서 1차원과 2차원의 병합, 2차원과 3차원의 병합, 그리고 1,2,3차원의 병합을 통해 통합 차원에서의 메타 모델 아키텍처를 제시한다. 1차원 피라미드에서는 주로 메타 모델 계층에 해당하고 2차원과 3차원 피라미드에서는 전 계층을 다루게 된다.

재공학은 역공학과 순공학 과정을 합한 과정을 말한다. 역공학은 원시 코드로부터 설계 정보, 아키텍처 정보, 그리고 기타 정보들을 추출하는 것을 말하며, 순공학은 요구사항 정의에서부터 출발하여 분석, 설계 정보를 거쳐 소스코드를 개발하는 것을 말한다. 재공학 과정에서는 이 두 단계의 과정들을 모두 거치게 된다. 이 2차원 재공학 메타 모델에서는 역공학 단계와 순공학 단계에 적용되는 모델들에 대한 정보를 메타 모델로 표현하며, 그림 4에 나타난 것처럼 이를 요구사항 분석, 아키텍처 설계, 그리고 상세 설계 등의 3 단계별로 계층화 시켜 표현한다. 그리고 중앙선을 기점으로 왼쪽은 역공학 단계에서 적용되는 모델들이고, 오른쪽은 순공학(컴포넌트 기반 개발) 단계에 적용되는 모델들을 의미한다.

그리고 말발굽 모양이 의미하는 것은 카네기 멜런 대학의 SEI에서 제시한 CORUM[9,10] 모델을 적용한 것으로서, 상위 계층으로 역공학을 실시하여 변환

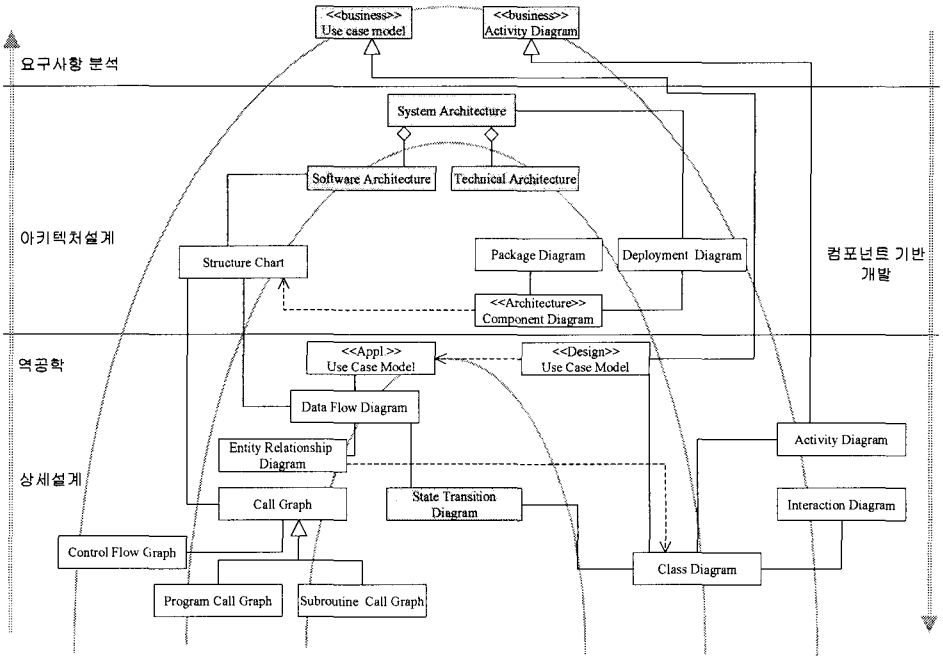

그림 4. 재공학 메타 모델 아키텍처

할 경우에 보다 더 재공학의 범위가 커지고, 새로운 요구사항까지 반영하여 시스템을 현대화 시킬 수 있다는 의미가 된다. 예를 들어 상세 설계 단계에서의 변환만 하게 되면 아키텍처는 그대로 레거시 시스템의 아키텍처를 따르고 다만 설계 정보만 변환하게 된다는 의미가 된다. 그러나, 본 논문에서는 재공학에서 요구사항까지 기존 레거시 시스템에 반영하고자 하여, 요구사항 분석까지 이르는 메타 모델을 정의하여 제시하는 것이다.

4.2 역공학 단계 메타 모델

현존 레거시 시스템들을 살펴 보면 설계 정보나 아키텍처 정보는 거의 없고 소스 코드와 데이터만이 존재하는 경우가 대부분이다. 따라서, 레거시 시스템을 새로운 목표 시스템으로 재공학 하기 위해서는 우선적으로 역공학 과정을 걸쳐야 한다. 본 논문에서는 이러한 역공학 과정을 통해서 추출해야 하는 모델들에 대한 메타 정보를 메타 모델로 표현하여 정의하고 있으며, 어느 업무에서 어떠한 모델들이 추출되어 정의되어야 하고, 각 단계에서 추출된 모델들 간의 관련성도 의존성(dependency)을 통하여 제시하고 있다.

그림 5에 제시된 것처럼 역공학 단계에서 수행되는 업무를 크게 상세 설계, 아키텍처 설계, 그리고 요구사항 분석으로 3계층을 나누어, 각 계층에서 표현되어야 할 모델들과 모델들의 정보들을 UML의 클래스 다이어그램 표기법을 이용하여 정의하고 있다. 일례로 유스 케이스 모델과 같은 경우는 상세 설계 계층과 요구사항 분석 계층에 두개 존재하는데 서로 표현하는 정보의 수준이 다르다. 그 이유는 상세 설계에서는 소스 코드로부터 설계 정보를 추출하여 표현하는 수준이기 때문에 유스케이스 명세 정보도 함께 추출이 가능하다. 그리고 이때의 유스 케이스의 굵기(granularity)는 함수 혹은 함수들의 그룹이 될 수 있다. 그러나 요구사항 분석 계층은 이보다 훨씬 추상화 된 계층이기 때문에, 이 때의 유스케이스는 비즈니스 프로세스 혹은 업무 단위의 유스 케이스가 된다. 즉, 어플리케이션 유스 케이스들의 그룹이 하나의 비즈니스 유스케이스로 대응이 될 수도 있다는 의미이다. 따라서 이 차이를 구별하기 위해 스테레오타입(stereotype)으로 ≪business≫와 ≪appl.≫으로 구별하여 표현하고 있다. 그리고 역공학 단계는 소스 코드에서 설계 정보 그리고 다시 분석 정보로 복구되기 때문에 상세 설계 계층에서 아키텍처 계층, 그리고 요구사항 계층으로 상향식으로 모델들이 추

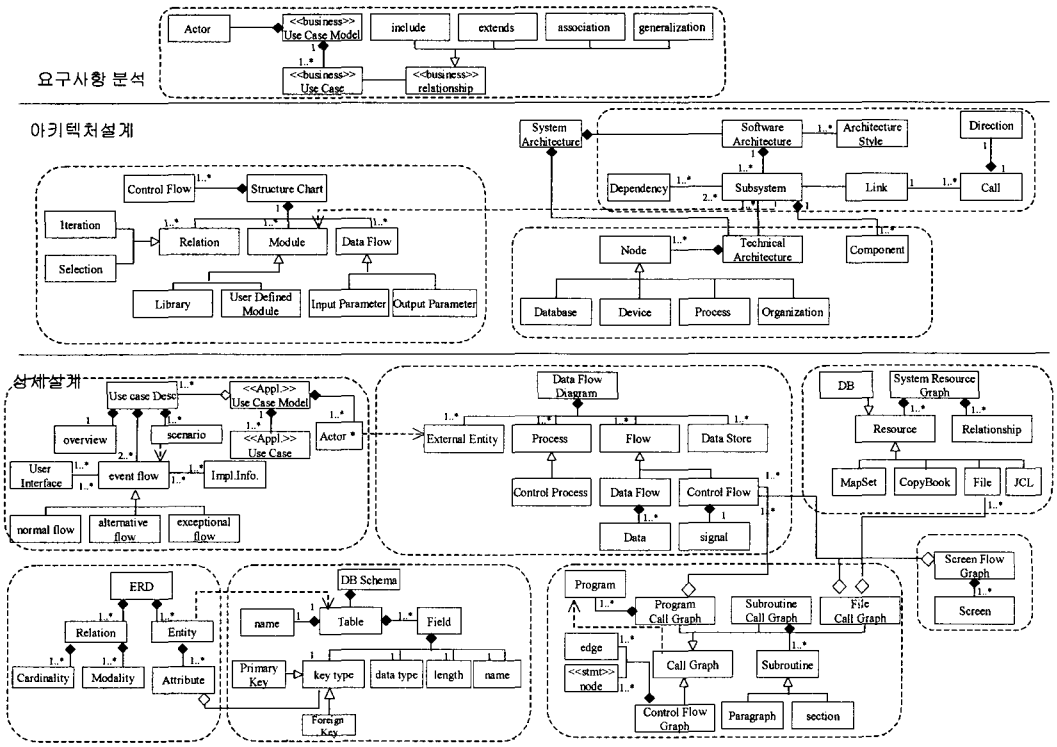

그림 5. 역공학 단계의 통합 메타 모델 아키텍처

출되어 정의 및 명세된다.

4.3 CBD 단계 메타 모델

여기서 말하는 CBD 단계는 재공학에 있어서 순공학을 의미하는데, 순공학의 접근 패러다임이 컴포넌트 기반의 소프트웨어 설계이기 때문에 CBD 단계로 표현한 것이다. 이 단계는 순공학에서 의미하는 과정들을 거쳐서 새로운 컴포넌트 기반 소프트웨어를 개발하게 된다. 그러나 기존 순공학과의 차이점은 시스템을 전부 새롭게 개발하는 것이 아니라 기존에 존재하는 시스템을 새로운 패러다임에 맞춰서 변환하는 것이다. 따라서 이 단계에서는 재공학 전략에 따라 모델 선정이 달라질 수 있다. 즉, 새로운 요구사항을 전혀 반영하지 않고 다만 기존 시스템의 기능만 변환할 경우에는 요구사항 분석의 요구사항 명세 부분의 모델은 반영되지 않을 것이다. 그러나, 보편적인 재공학 형태는 기존 레거시 시스템에 새로운 요구 사항들을 추가하여 시스템을 변환한다. 이런 경우는 그림 6에 제시된 모델들이 모두 반영되게 된다.

그림 6을 보면 일부 모델들은 역공학 단계의 메타 모델에 표현된 모델들과 중복되어 표현되고 있다. 예를 들어 아키텍처 정의 부분의 모델들이 그러한 경우이다. 이는 그림 4에 표현된 것처럼 이 모델이 역공학 단계에서도 사용되고, CBD 단계에서도 사용되기 때문이다. 그러나 역공학 단계의 모델보다는 컴포넌트 다이어그램이나, 패키지 다이어그램 등과 같이 일부 모델들이 더 추가되어 표현된다.

4.4 역공학과 CBD 통합 메타 모델

그림 6은 역공학 단계의 메타 모델들과 CBD 단계의 메타 모델들을 통합한 모델을 보여주고 있다. 그림 7에 나타난 것처럼 역공학 단계에서 추출된 모델들이 CBD 단계에서 어떤 모델로 어떻게 적용되고 매핑되는지 그 관계가 의존도 관계로 표현되어 있다.

이를 통해 역공학 단계의 모델들이 CBD 단계에서 어떤 모델로 연결되어 표현되는지를 쉽게 파악할 수 있을 뿐만 아니라, 레거시 시스템의 모델들의 표현 수준과 모델들의 추출되는 단계 정보 등도 계층적으로 표현되어 있어서 이해가 용이하다.

4.5 3차원 공간에서의 모델 간의 관계

지금까지 단계별 혹은 통합 메타 모델을 살펴 보았

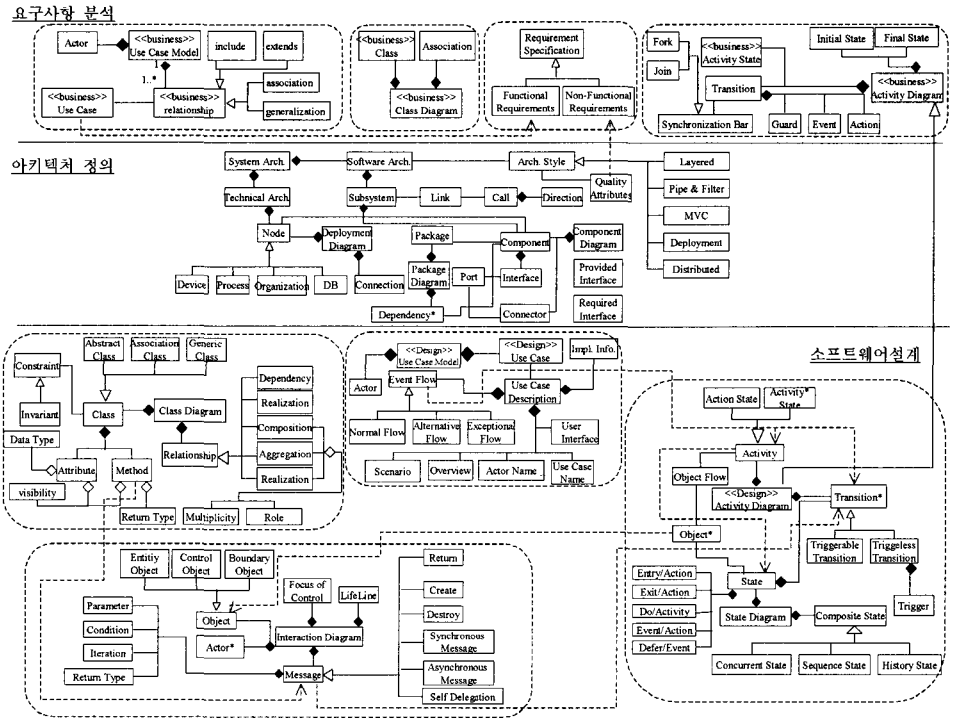

그림 6. CBD 단계 메타 모델

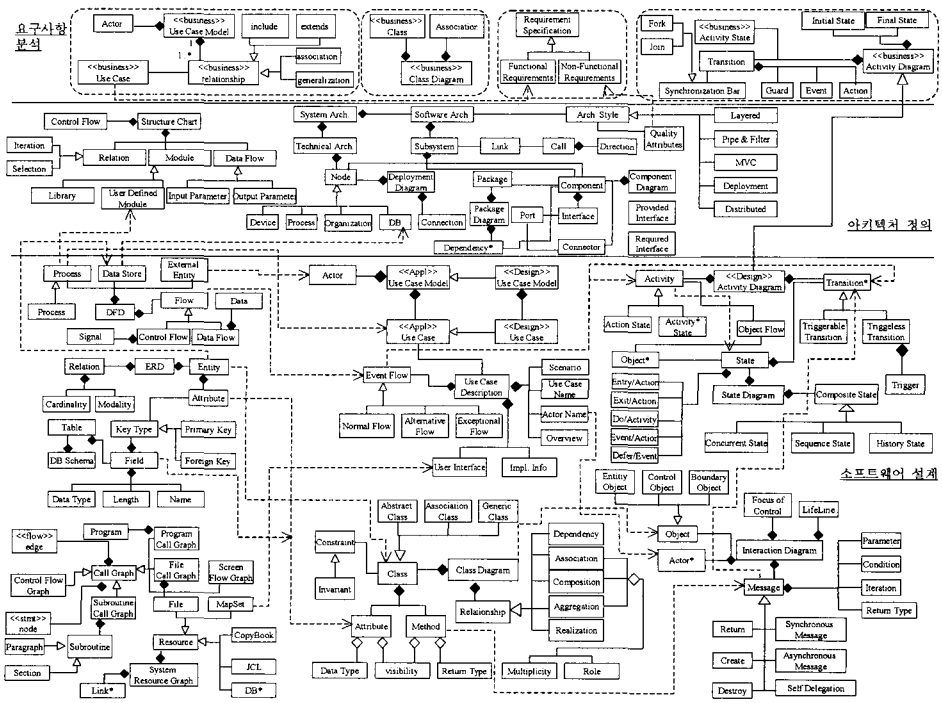

그림 7. 역공학 + CBD = 재공학 통합 메타 모델

다. 여기서는 이러한 통합 메타 모델의 기반이 된 3차원 공간 개념을 적용하여 본 논문에서 제시한 모델들과 그 모델들이 어떠한 단계에서 적용되는 지를 제시한다.

그림 8에 나타나 있는 것처럼 1차원의 메타 피라미드 D1, 2차원 공학 피라미드 D2, 3차원 묘사 피라미드 D3의 개념을 총체적으로 반영하여 각 차원 별로 본 논문에서 제시한 모델들 가운데 일부가 표현되었다.

예를 들면, 유스케이스모델의 경우, D1으로는 메타 모델(MM)에서 비즈니스 유스케이스 모델이 정의되어 있고, D2로는 요구사항(Requirements) 계층에 해당하고, D3로는 추상화 묘사 수준에 해당한다.

5. 실험 및 평가

5.1 사례 연구

이 절에서는 4장에서 정의한 메타 모델을 기반으로 실제 현업 업무에 적용한 사례들 가운데 리스 영업 관리 시스템을 사례 연구로 제시한다. 이 시스템은 IBM AS/400에서 운영되는 COBOL 기반의 시스템이다. 이 시스템은 소스코드가 3만 라인이고 프로그램이 124개이다. 한 리스 회사가 3개의 회사를 합병하게 되어서 이기종의 시스템 3개를 함께 연동하기 위해서 이기종의 시스템들을 단일화되고 현대화된 시스템으로 변환하였다. 특히 기존의 메인 프레임으로 인한 유지 보수 비용과 서로 다른 유형의 시스템 3개를 하나로 효율적으로 운영해야 할 필요에서 재공학을 하게 되었다. 이 시스템의 재공학 목표는 크게 4가지이다. 첫째는 구조적 시스템을 컴포넌트 기반 시스템으로 변경하는 것이다. 둘째는 IBM/AS 400에서 운영되던 시스템을 J2EEE 플랫폼으로 변경하는 것이다. 셋째는 시스템의 일부 비즈니스 로직은 변경하고 일부는 래핑하는 것이다. 넷째는 새로운 요구사항을 함께 반영하여 컴포넌트로 개발하여 레거

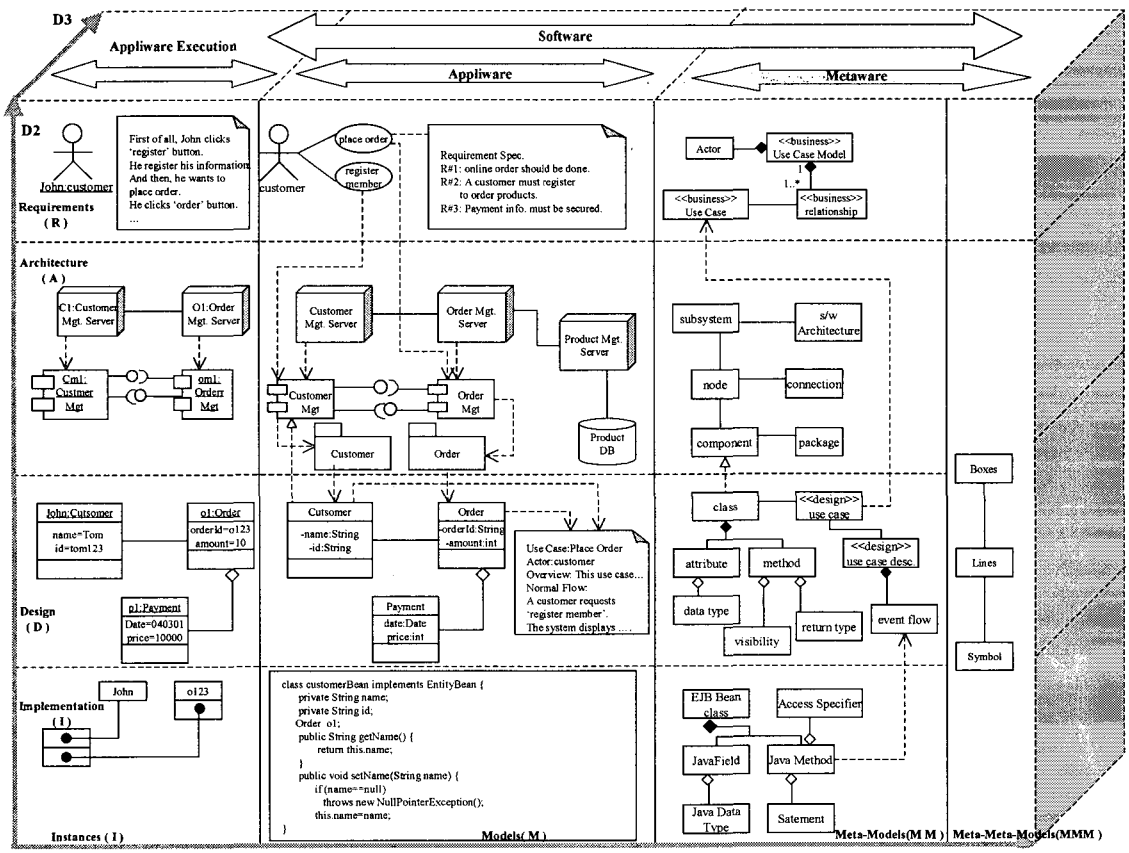

그림 8. 3차원 공간 개념을 통합한 모델 관계

시 시스템을 확장하는 것이다.

이 시스템을 재공학 하는 과정에서 본 논문에서 제시한 모델을 기반으로 추출한 모델들을 제시한다.

그림 9은 소스코드로부터 리스 영업 업무 서브 시스템에 대하여 서브루틴들 간의 호출 관계를 표현한 서브루틴 호출 그래프(subroutine call graph)이다. 이는 상세 설계의 서브루틴 호출 그래프 메타 모델을 기반으로 생성된 것이다. 이는 참여했던 회사인 K사의 역공학 도구를 통해 산출된 형태이다.

그림 10은 COBOL 소스 코드를 기반으로 하여 추출된 서브루틴 제어 흐름 그래프의 일부이다. 제어 흐름 그래프는 하나의 프로그램 혹은 하나의 서브루틴 단위로 해당 프로그램이나 서브루틴 내에서의 제어 흐름 정보를 표현해 주는 모델이다. 이 예제는 서브 루틴에 대하여 제어 흐름을 보여주고 있다.

그림 11은 소스코드로부터 기능들을 추출하여 생성한 리스 영업 관리 시스템의 영업 상담 부분에 대한 어플리케이션 유스 케이스 다이어그램이다. 이는 역공학 단계 메타 모델에서 설계 단계의 유스 케이스 모델에 해당한다. 이 시스템은 새로운 기능적 요구사항은 추가되지 않았기 때문에 유스 케이스 모델은 역공학 단계와 CBD 단계 모두 동일 수준으로 표현

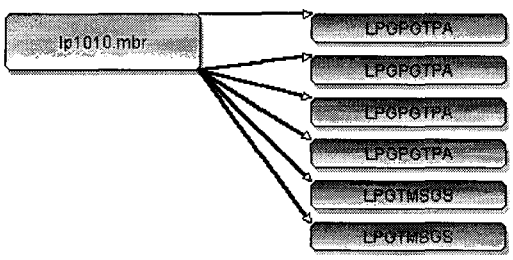

그림 9. 서브 루틴 호출 그래프

```
100   1100-WORK-CLR.
101       INITIALIZE              GT-WORK-A.
102       MOVE   SPACE      TO   WK-JOBID.
103       MOVE   ZERO       TO   WORK-A.
104       MOVE   ZERO       TO   IND-A.
105       MOVE   ZERO       TO   SCR-CTL.
106       MOVE   ZERO       TO   SCR-SAVE-A.
107       IF     GMS-CODE01 =  11  OR  12 OR 01 OR 02
108              PERFORM  G060-MSG-CALL  THRU  G060-EXIT
109       ELSE   MOVE   SPACE   TO   SS-SHELPMSG
110              MOVE   ZERO    TO   GMS-CODE01
111              MOVE   ZERO    TO   GMS-RTN19
112              MOVE   SPACE   TO   GMS-NAME11.
113       MOVE   SPACE      TO   SCR-TERMID SCR-FORMT.
114       MOVE   GT-JOBID   TO   WK-JOBID.
115       MOVE   WK-JOBID1  TO   SS-SNAME.
```

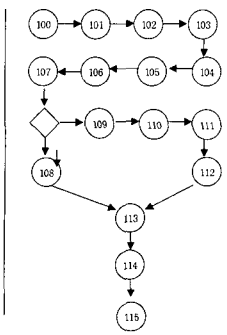

그림 10. 제어 흐름 그래프

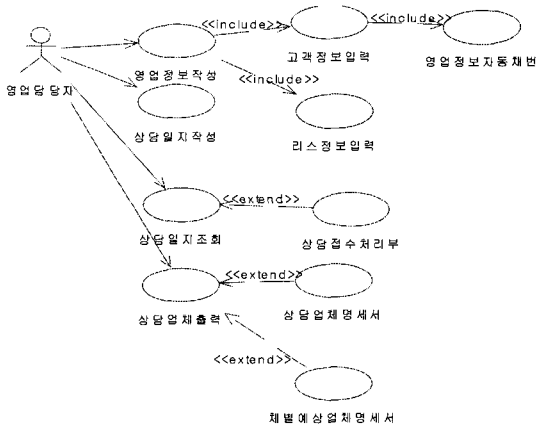

그림 11. 어플리케이션 유스 케이스 모델

되었다.

그림 12는 상담 업무에 대한 구조도이다. 상담 업무를 구성하는 서브 모듈들 단위로 구성된 구조도 형태이다.

그림 14는 그림 13의 개체 관계도를 컴포넌트화 단계에서 클래스 다이어그램으로 변환한 예제이다. 역공학 단계의 개체 관계도를 추출하면 개체 관계도의 개체는 클래스로, 속성은 클래스의 속성으로 대응되어 변환가능하며, 개체 관계도의 Cardinality나 Modality도 클래스 다이어그램의 다중성(Multiplicity)로 변환이 가능하다. 여기 예제는 역공학 단계의 개체 관계도를 클래스 다이어그램으로 변환한 부분만 보여주지만, 실제 컴포넌트화 단계 진행시에 새로운 요구사항이 추가될 경우에는 개체 관계도의 모델에 새로운 개체들이 더 추가적으로 반영되어 클래스 다이어그램을 작성하게 된다.

그림 15는 컴포넌트들을 추출하여 컴포넌트 기반 시스템으로 변환할 목표 시스템의 컴포넌트 다이어그램을 보여주고 있다. 그림 15에 제시된 바와 같이 공통 관리 기능을 담당하는 'CommonMgmt'라는컴포넌트가 있으며, 이 컴포넌트에서 여러 다른 컴포넌트들과 상호 작용하며 컴포넌트들 간의 연동이 이루어지게 된다.

5.2 평가

이전 절에서는 사례 연구를 통하여 본 연구에서 제시한 아이디어가 실제 재공학 과정에 어떻게 적용될 수 있는지를 제시한 것이다. 이 절에서는 본 논문

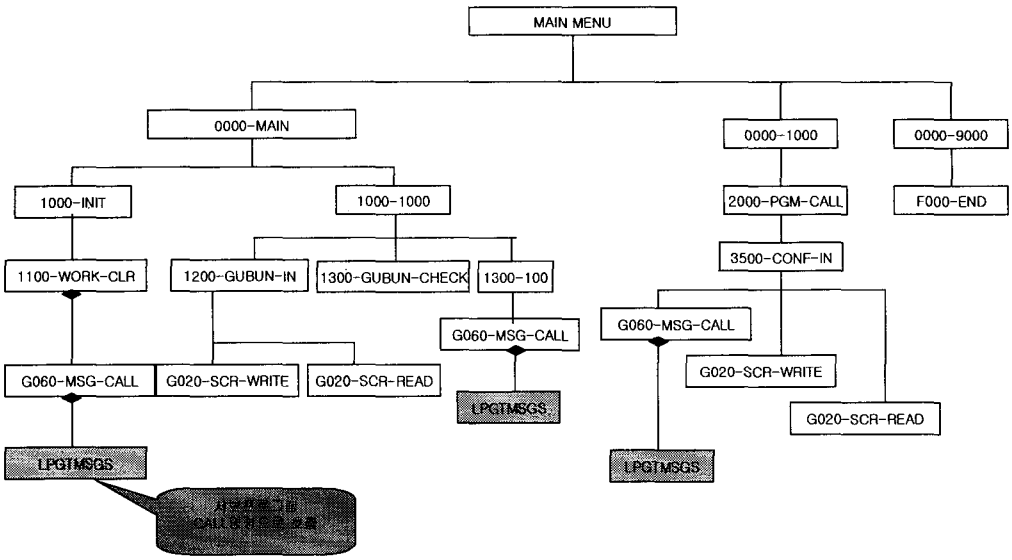

그림 12. '상담' 업무에 대한 구조도

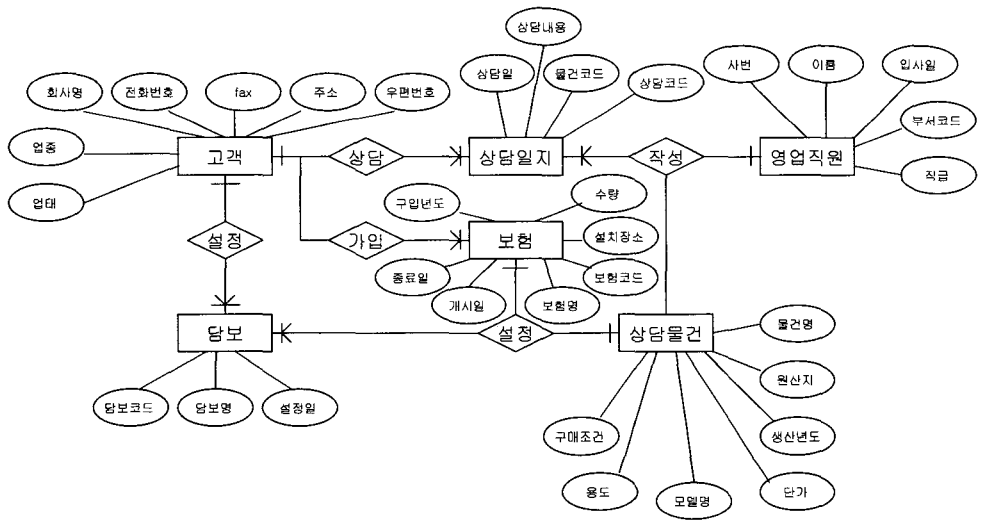

그림 13. 개체 관계도

에서 제시한 메타 모델을 적용해서 모델과 아키텍처 기반 하에서 레거시 시스템을 재공학 했을 경우와 코드 기반으로 재공학 했을 경우에 있어서 변환된 목표 시스템의 품질 특성[12]과 방법론 측면에서 어떠한 차이가 발생하는지를 제시함으로써 본 논문에서 제시한 접근법의 타당성을 증명하고자 한다.

5.2.1 품질 특성을 통한 비교

금융, 리스, 통신 분야 등 여러 유형의 레거시 시스템에 모델 기반의 접근을 통한 재공학을 실시한 경험과 코드 기반의 재공학을 실시한 경험을 바탕으로 얻는 데이터를 가지고 비교한 결과가 그림 16에 제시되어 있다. 그림 16에 제시된 것처럼 코드 기반으로 재공학 한 시스템과 모델 기반을 적용하여 재공학 한 시스템간의 차이가 발생하는데 우선적으로 개발된 소스 코드의 길이에서 차이가 난다. 모델 기반으로 적용한 시스템이 코드 기반으로 적용했을 때에 비해서 코드의 길이가 줄어 든다. 물론 이 경우는 새

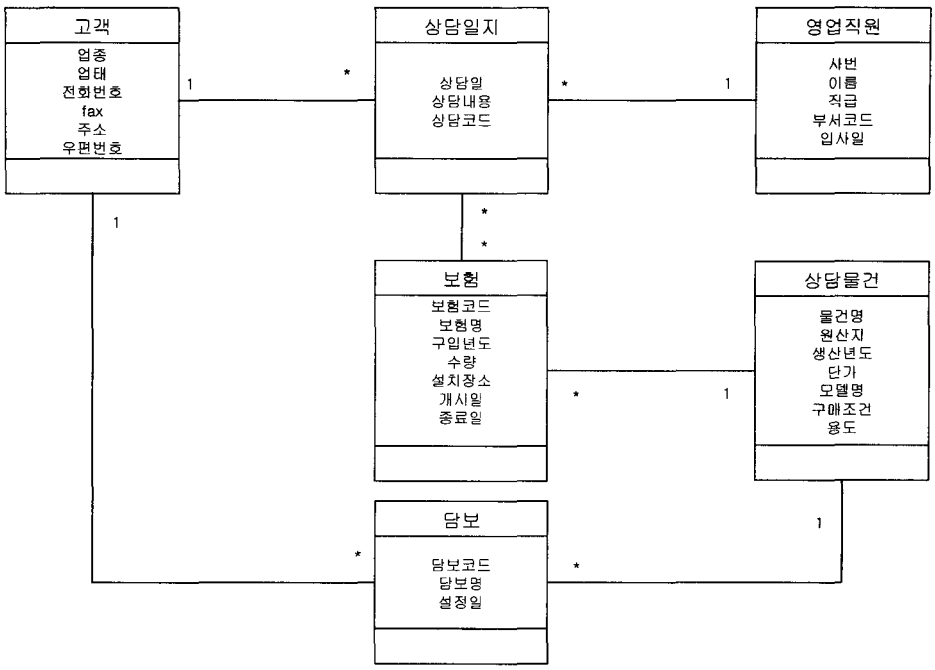

그림 14. 클래스 다이어그램

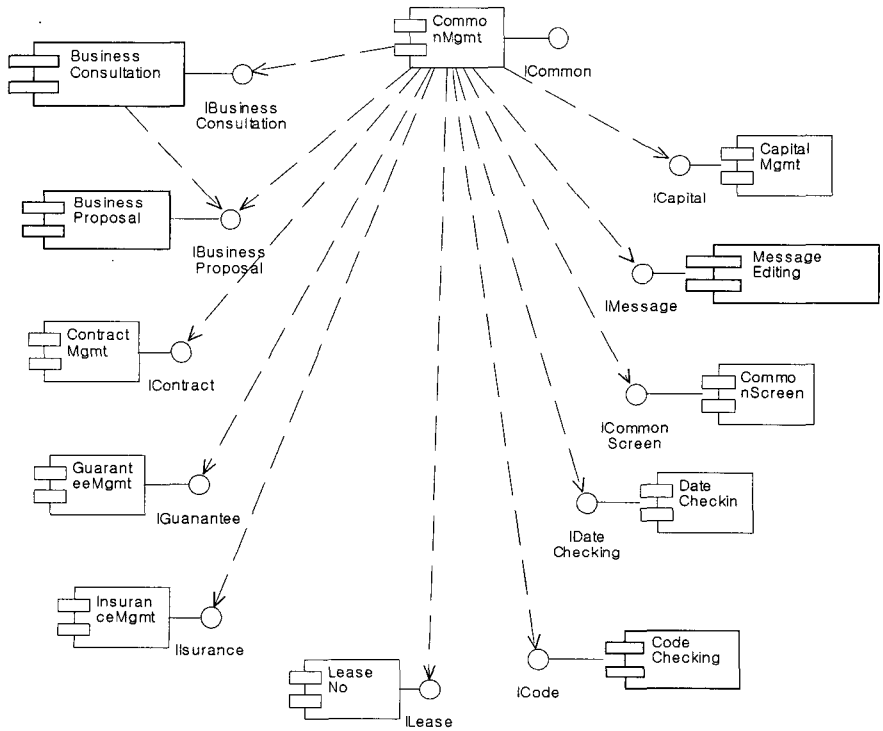

그림 15. 컴포넌트 다이어그램

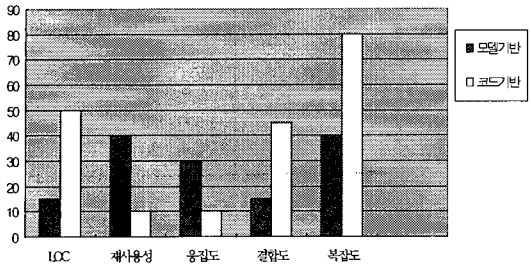

그림 16. 품질 특성에 의한 비교

로운 요구사항을 반영하지 않았을 경우이다. 그리고 재공학 된 시스템의 복잡도 등을 측정하였을 때에도 모델 기반을 적용하여 구축한 경우가 훨씬 낮음을 알 수 있다.

5.3 방법론 측면 비교

여기서는 기존 기법들과 본 연구에서 제시한 기법과의 차이를 방법론 측면에서 어떻게 상이한지에 대하여 비교 평가하고자 한다. 표 1에 나타난 것처럼 기존의 MORALE이나 Butterfly 또는 Sneed의 방법론에서는 메타 모델 기반의 재공학 접근법을 적용하지 않았으나 본 연구에서 메타 모델을 기반의 재공학 접근법을 적용하고 있다.

이러한 재공학 접근법을 적용함으로써 재공학 프로세스에 있어서 모델들 간의 일관성이나 레거시 시스템에 대한 이해와 변환이 쉽게 이루어 질 수 있다. 그리고 기존의 방법론들은 재공학 전단계에 적용되는 모델들을 지원하기 보다는 특정 단계에 적용되는 모델들 만을 지원하기 때문에 재공학에 있어서 자연스러운 모델 변환이 이루어지기 어렵다. 그러나 본 논문에서 제시한 메타 모델은 역공학과 순공학 단계 모두의 모델들을 제시하고 모델들 간의 관련성 또한 제시하기 때문에 재공학 전체 프로세스를 지원할 수 있다.

6. 결 론

레거시 시스템을 컴포넌트 기반 시스템으로 재공학 하기 위해서는 스크린 스크래핑, 래핑 혹은 변환 등과 같은 다양한 접근법들을 적용할 수 있다. 그러나 현재 대부분의 재공학 방법론마다 동일한 표현 정보를 제공하면서도 독단적인 모델을 사용하거나 표기법을 방법론마다 다르게 사용하고 있다. 이렇게 함으로 인해서 재공학 개발자들로 하여금 모델들에 대한 혼란을 야기시킬 뿐만 아니라 모델들의 관계성을 이해하는데 많은 어려움을 주고 있다.

본 논문에서는 이러한 문제점들을 인식하여서 모델들을 계층화시켜서 구분하였고, 또한 소프트웨어 공학 측면에서 3차원 공간 개념을 적용하여 모델들을 메타 모델에 기반을 두고 표현함으로써 단계와 모델 간의 관계, 모델과 모델 간의 관계, 그리고 모델의 묘사 수준 정도를 구별하여 제시하고 있다. 이는 재공학 개발자들로 하여금 재공학 프로젝트 성격에 따라 필요한 모델들을 선택적으로 선정하여 사용할 수 있을 뿐만 아니라 레거시 시스템의 모델이 새로운 컴포넌트 기반 시스템의 모델로 어떻게 대응시켜야 할 것인지에 대해서도 쉽게 접근할 수 있게 해준다.

본 논문에서 제시한 통합 모델을 정형화 된 명세를 통하여 정립하면 자동화 도구로의 개발이 용이하며, 이는 재공학 프로세스에 있어서 많은 생산성의 효과를 가져오리라 기대된다. 앞으로의 향후 연구과제는 메타 모델을 정형화 된 언어로 명세하여, 이 모델에

표 1. 방법론 측면을 통한 비교

	MORALE[13]	Butterfly[14]	Sneed 방법론	제안한 기법
재공학 접근방식	코드 기반	데이터 모델	모델 기반	메타모델 기반
모델 지원 여부	UI 모델 일부	데이터 모델	역공학 일부	역공학 + 순공학
메타모델	-	-	-	O
아키텍처 정보	시스템 아키텍처	데이터 아키텍처	소프트웨어아키텍처	시스템 아키텍처, 소프트웨어 아키텍처
모델의 계층화	-	-	일부	계층적 모델
모델의 추상화 수준	낮음	데이터 모델 국한	역공학 모델 추상화	재공학 모델 추상화

대한 정확한 일관성 검증 기법과 자동화 메커니즘으로의 개발을 하는 것이다.

참 고 문 헌

[1] William Ulrich, *Legacy Systems : Transformation Strategies*, Prentice Hall, 2002.

[2] Bennett, K, "Legacy Systems: Coping With Success", *IEEE Software*, Vol. 12, No. 1, pp. 19-23, 1995.

[3] Sneed, H., Majnar, R., "A Case Study in Software Wrapping", *Int. Conf. in Software Maintenance,* Nov 16-20, Bethesda, Maryland, IEEE Computer Society Press, pp. 86-93, 1998.

[4] Cimitile, A., De Lucia, A., Di Lucca, G., "An Experiment in Identifying Persistent Objects in Large Systems", *International Conference on Software Maintenance,* Nov 16-20, Bethesda, Maryland, *IEEE Computer Society Press*, pp. 122-130, 1998.

[5] De Lucia, A., Di Lucca, G., Fasolino, A., Guerra, P., Petruzzelli, S., "Migrating Legacy Systems Towards Object-Oriented Platforms", *Int. Conf. in Software Maintenance,* pp. 122-129, 1997.

[6] Sneed, H., "Encapsulating Legacy Software for Use in Client-Server Systems", *Working Conference in Reverse Engineering,* IEEE Computer Society Press, pp. 104-119, 1996.

[7] Sneed, H., Majnar, R., "A Case Study in Software Wrapping", *Int. Conf. in Software Maintenance,* Nov 16-20, Bethesda, Maryland, IEEE Computer Society Press, pp. 86-93, 1998.

[8] Abowd G. Goel A. Jerding D.F., McCracken M., Moore M., Murdock J.W., Potts C., Rugaber S., Wills L., "MORALE. Mission ORiented Architectural Legacy Evolution"*International Conference on Software Maintenance*, pp.150-159. 1997.

[9] Nelson Weiderman, Dennis Smith, Scott Tilley, "Approaches to Legacy System Evolution", *CMU/SEI-97-TR-014.*

[10] SEI Reengineering Center "Perspectives on Legacy System Reengineering", 1995.

[11] Dolly M, Neumann, "Evolution Process for Legacy System Transformation", *IEEE Technical Applications Conference*, Northcon/96, pp.57-62, 1996

[12] S.R. Chidamber and C.F. Kemerer, "A Metric Suite for Object-Oriented Design", IEEE Transactions on Software.

[13] Gregory A., et. al., "MORALE: Mission Oriented Architectural Legacy Evolution", Proceedings of International Conference on Software Maintenance'97, Bari, Italy, September 29-October 3, pp.150-159, 1997.

[14] Bing Wu, et. al., "The Butterfly Methodology: A Gateway-free Approach for Migrating Legacy Information Systems" Proceedings of the 3rd IEEE Conference on Engineering of Complex Computer Systems(ICECCS97), Villa Olmo, Como, Italy, September 8-12, pp.200-205, 1997.

조 은 숙

1993년 동의대학교 전산통계학과 이학사

1996년 숭실대학교 컴퓨터학과 공학석사

2000년 숭실대학교 컴퓨터학과 공학박사

1999년~2000년 8월 ㈜객체정보기술 과장

2000년 9월~2004년 2월 동덕여대 정보학부 데이터정보전공 강의전임교수

2004년 3월~현재 동덕여대 정보학부 데이터정보전공 전임강사

2002년 1월~현재 한국전자통신 연구원 초빙연구원

관심분야 : 컴포넌트기반 소프트웨어 개발 방법론, 소프트웨어 아키텍처, 컴포넌트 정형명세, 컴포넌트 메트릭

김 철 진

1996년 경기대학교 전자계산학과 졸업(학사)
1998년 숭실대학교 대학원 컴퓨터학과 졸업(석사)
2004년 숭실대학교 대학원 컴퓨터학과 졸업(박사)
2004년~현재 가톨릭대학교 컴퓨터 정보 공학부 초빙교수
관심분야 : 컴포넌트 기반 소프트웨어 공학, 컴포넌트 커스터마이제이션, 개발 방법론