韓國컴퓨터情報學會 論文誌
第16卷 第11號, 2011. 11.

2011-16-11-1-4

관계형 데이터베이스 응용시스템을 위한 통합 설계방법론 개발 -객체지향 분석·설계 방법론을 중심으로-

주 경 수*, 조 도 형*

Development of Integrated Design Methodology for Relational Database Application -Focusing on Object-Oriented Analysis and Design Methodology-

Kyung-Soo Joo*, Do-Hyung Jho*

요 약

본 논문에서는 UML(Unified Modeling Language)을 토대로 유스케이스(use case) 중심의 객체지향 분석·설계 방법론을 기반으로 한, 관계형 데이터베이스 통합 설계방법론에 대하여 다룬다. 본 통합 설계방법론에서 사용하는 개념모델은 비즈니스 프로화일(business profile)에 기반을 두고 있으며, 6단계로 구성되어 있다. 첫 번째 단계에서는 비즈니스 유스케이스(business use case)가 식별되어 마크로액티비티 다이어그램(macroactivity diagram)으로 표현되고, 두 번째 단계에서 마크로액티비티 다이어그램은 비즈니스 객체(business object)와 비즈니스 객체흐름(business object flow) 그리고 비즈니스 사용자책임(business worker's responsibilities)을 이용하여 상세 비즈니스 활동다이어그램(detailed business activity diagram)으로 변환된다. 세 번째 단계에서는 상세 비즈니스 활동다이어그램을 기반으로 시스템 전체의 정적 구조를 설명하는 비즈니스 클래스다이어그램(business class diagram)으로 변환된다. 네 번째 단계에서 비즈니스 클래스다이어그램은 대응하는 관계형 데이터베이스 초기 개념모델을 대표하는 클래스다이어그램으로 변환된다. 다섯 번째 단계에서 클래스다이어그램에 일반화와 특수화, 역할과 활동, 클래스 추가 그리고 중복 연관에 따른 추가적인 변환이 이루어지고, 마지막으로 관계형 데이터베이스 스키마로 변환이 이루어지게 된다. 본 논문에서 제시하는 방법론을 적용함으로써, 객체지향 분석·설계 방법론과 관계형 데이터베이스 설계방법론 사이에 유기적 연결이 이루어지게 되어, 객체지향 분석·설계 방법론

• 제1저자 : 주경수 • 교신저자 : 주경수
• 투고일 : 2011. 06. 30, 심사일 : 2011. 08. 25, 게재확정일 : 2011. 09. 16.
* 순천향대학교 컴퓨터소프트웨어공학과(Dept. of Computer Software Engineering, SoonChunHyang University)
※ 이 논문은 2008년도 순천향대학교 교수 연구년제에 의하여 연구하였음.

과 관계형 데이터베이스 설계 방법론을 통합적으로 다룰 수 있게 된다. 이에 따라 관계형 데이터베이스 기반의 소프트웨어 시스템에 대한 객체지향 방식의 일관된 그리고 통합된 구축방안이 제공된다. 사례 연구로 제안한 통합 설계 방법론을 비자발급시스템에 대하여 적용한다.

▸ Keyword : 객체지향 분석·설계, 관계형 데이터베이스, UML

Abstract

In this paper we present an integrated design methodology for relational database based on object-oriented analysis and design. The integrated design methodology is based on business profile and has six phases. In the first phase, business use cases are identified and described by macroactivity diagrams and then the macroactivity diagrams are transformed to detailed business activity diagrams by using objects, object flows and business worker's responsibilities. In the third phase, the detailed business activity diagrams are transformed to business class diagrams that describe the static structure of the entire business system based on detailed business activity diagrams. In the four phase, the business class diagrams are transformed to class diagrams that represent the initial conceptual model of the target relational database. In the five phase, we add additional transformations on the class diagrams with generalization and specialization of associations, roles, activities, additional classes and redundant associations. Eventually, the final class diagrams are transformed to relational database schema. The methodology presented in this paper by applying that proposal for organic connection between object-oriented analysis and design methodology and relational database design methodology. And it will be able to deal with integration management. By the integrated design methodology, we can make more easily software systems based on relational database. In the case study, proposal integrated design methodology applied for a visa issuing system.

▸ Keyword : Object-Oriented Analysis·Design, Relational Database, UML

I. 서 론

UML은 소프트웨어 시스템의 시각적 모델링 언어이다. 이것은 선도적인 모델링 언어로서 소프트웨어 모델링의 산업 표준이다. UML이 소프트웨어 시스템 모델링에 폭 넓게 적용되는 이유는 독립적이고 풍부한 표기법을 사용하고 개념에 대한 확장이 가능하기 때문이다[1].

정보시스템 개발에 있어서 비즈니스 모델링을 첫 번째 단계로 하는 두 가지 이유가 존재한다. 첫째로 비즈니스 모델링은 정보시스템 설계의 기초로서 시스템의 기능과 비기능 요구사항을 식별할 수 있다. 둘째로 비즈니스 모델은 대응하는 정보시스템의 소프트웨어 모델에 맞게 변환될 수 있다.

기존의 유스케이스 중심의 대표적인 객체지향 분석·설계 방법론 중에는 RUP(Rational Unified Process)가 존재하는데 이 방법론의 특징은 유스케이스 기반, 아키텍쳐 중심, 반복 및 점증적이며, 도메인 모델과 유스케이스 모델과 분석 모델과 설계 모델 그리고 구현 모델 등으로 작성 된다[2]. 또한 이 객체지향 분석·설계 방법론을 통해서 도출된 개념적 모델인 클래스다이어그램을 바탕으로 객체지향 프로그래밍 코드를 생성할 수 있다. 그러나 RUP는 관계형 데이터베이스 설계 방법론에 대한 일관되고 통합된 방안을 제시하지 못하고 있으며, 다만 관련된 케이스 툴을 사용하여 E-R다이어그램을 도출한다.

대표적인 관계형 데이터베이스 설계 방법론 중에는 정보공학 방법론(Information Engineering)이 있다. 이 방법론은 비즈니스 시스템의 성장과 소프트웨어 공학(Software engineering)의 발전에 따라 나타나게 되었으며, 시스템의 계획, 분석, 설계 및 구축 하는 데이터 중심의 방법론이다. 정보공학 방법론은 업무 영역에서 가장 안정적인 요소를 데이터 중심으로 시스템을 구축하고, 일관성 있고 통일된 정보시스템

구축이 가능하며 데이터 중심의 업무절차 및 환경변화에 유연한 장점을 가지고 있고, 시스템 구축 방법의 주를 이루고 있는 안정된 방법론 중의 하나이다. 그러나 이 정보공학 방법론 또한 널리 사용되고 있는 객체지향 분석·설계 방법론과 관련하여 어떤 연관도 제시하고 있지 않다.

그리고 객체지향 분석·설계 방법론을 기반으로 한 관계형 데이터베이스 설계 방법론을 다루고는 있으나, 관계형 데이터베이스 스키마 변환작업과 관련한 부분은 다루지 않고 있다[3].

본 논문에서는 비즈니스 프로화일을 기반으로 한 6단계 변환 방법을 제시함으로써, 객체지향 분석·설계 방법론을 기반으로 하여 관계형 데이터베이스 구축을 위한 일관되고 통합된 설계 방법론을 개발했다.

본 논문의 구성은 다음과 같다. 2절에서 기존의 설계 방법론에 대하여 설명하며 3절에서는 6단계 변환 방법 중 4단계까지의 방법인 클래스다이어그램 도출 과정을 설명한다. 그리고 4절에서는 3절에서 도출된 클래스다이어그램에 추가적인 변환작업에 대해 다루며, 5절에서 클래스다이어그램을 관계형 데이터베이스 스키마 변환 과정에 대하여 다룬다. 마지막으로 6절에서는 결론을 요약한다.

II. 관련 연구

소프트웨어 개발방법은 소프트웨어 공학의 대가에 의해 소프트웨어 개발 방법론이라는 학문형태로 자리를 잡고 있다. 따라서 방법이라는 것은 일반적으로 개발 방법론이라 볼 수 있다. 세계적인 소프트웨어 개발방법론은 크게 구조적 분석 및 설계(structured analysis and design), 정보공학(information engineering), 그리고 최근에 각광받고 있는 객체지향 방법론(object-oriented methodology)이 있다. 실제적으로 보면 현재 기업이나 조직에서 사용하는 방법론은 이들을 적절하게 혼합하여 사용하고 있다. 구조적 방법론의 프로세스 모델링과 정보공학의 데이터 모델링을 동시에 사용하고 있다. 이는 조직의 환경이나 개발환경에 맞게 적절하게 기본적인 방법론을 변형시켜 적용시키는 것으로 볼 수 있다. 구조적 방법이 기능중심이고, 정보공학 방법이 데이터 중심이라면, 객체지향 방법론은 데이터적 요소와 기능적 요소를 하나의 관점으로 표현한 객체 중심이다. 다양한 표현 기법을 통해 쉽게 객체를 추출하고, 추상화 수준을 높여 정확하고 단순하게 실세계의 것들을 표현함으로써 복잡성을 줄이고, 재사용을 현실화시키고 있다. 특히 객체지향 방법론은 새롭게 만들어진 것이 아니라, 구조적 방법과 정보공학의 장점을 그대로 수용함으로써 상호 보완적인 방법을 추구하고 있다.

UML은 객체지향 분석(analysis)과 설계(design)를 위한 모델링 언어이다. UML은 소프트웨어를 시각화하고, 기술하고, 구축하며 또한 산출물들을 문서화하는데 사용되는 모델링 언어를 말한다. UML은 소프트웨어 개발에 사용하기 위한 표기법들을 제시하여 여러 다이어그램들을 정의하고 있으며, 이런 다이어그램에 대한 의미를 정의하고 있다. UML은 여러 다이어그램들을 통해 소프트웨어 개발과정의 산출물들을 비주얼하게 자공하고, 개발자들과 고객 또는 개발자들 간의 의사소통을 원활하게 할 수 있도록 하고 있다. UML은 시스템을 모델링 뿐만 아니라 모델링한 결과를 쉽게 파악할 수 있게 된다. 또한 산업계 표준으로 채택되었기 때문에 UML을 적용한 시스템은 신뢰성 있는 시스템으로 평가받을 수 있다. 기존의 유스케이스 중심의 대표적인 객체지향 분석·설계 방법론 중에는 RUP(Rational Unified Process)가 존재하는데 이 방법론을 통해서 도출된 개념적 모델인 클래스다이어그램을 바탕으로 객체지향 프로그래밍 코드를 생성할 수 있다. 그러나 RUP는 관계형 데이터베이스 설계 방법론에 대한 일관되고 통합된 방안을 제시하지 못하고 있으며, 다만 관련된 케이스 툴을 사용하여 E-R다이어그램을 도출한다.

따라서 이러한 특징들을 통하여 본 논문에서는 비즈니스 프로화일을 기반으로 한 6단계 변환 방법 제시와 객체지향 분석·설계 방법론을 기반으로 하여 관계형 데이터베이스 구축을 위한 일관되고 통합된 설계를 가능하게 한다. 다음 절에서 6단계 변환 방법 중 4단계까지의 방법인 클래스다이어그램 도출 과정에 대하여 소개한다.

III. 초기 개념적 모델

앞에서 언급한 바와 같이, 유스케이스는 시스템 기능 확인을 위한 적절한 개념과 시스템 요구사항 명세를 위한 기초 자료를 제공한다. 유스케이스는 시스템이 외부 시스템에 제공하는 외부 표시 기능의 일관된 단위이며, 내부 구조를 다루지 않고 소프트웨어 시스템 외부에서 보이는 행동을 설명한다[4]. 전체 시스템은 액터(actor)와 상호 작용을 한다. 따라서 유스케이스는 구체적인 가치를 생성하는 액터에 의해 전체 소프트웨어 시스템 시작 이벤트나 트랜잭션(transaction) 작업의 순서이다.

소프트웨어와 비즈니스 시스템 사이의 유사한 기능에 관하여 제공하는 외부 시스템은 비즈니스 유스케이스 중심의 비즈니스 모델링 접근법을 위한 기초가 된다[5]. 비즈니스 모델은 외부 비즈니스 모델과 내부 비즈니스 모델 두 개의 관점에서 만들어 진다. 외부 모델은 비즈니스 유스케이스에 의한 시스

템과 인터페이스를 보여준다. 비즈니스 유스케이스는 비즈니스 시스템이 외부 시스템 즉, 액터에게 제공하는 특정한 기능이라 할 수 있다. 다시 말해 특정 비즈니스 액터를 위하여 구체적으로 인식할 수 있는 가치의 결과가 되는 활동의 순서이다[6]. 비즈니스 액터는 공급자, 구매자 등 다른 사람일 수 있다.

초기 비즈니스 모델은 비즈니스 시스템의 내부 관점을 대표하고, 사용자의 행동(worker's behavior), 사용된 자원 그리고 그들 간의 관계의 전체적인 흐름을 보여준다. 표준 UML 비즈니스 프로화일은 비즈니스 모델이 외부 모델인 비즈니스 유스케이스 모델과 내부 모델인 비즈니스 객체 모델(business object model)로 구성된다[7]. 비즈니스 유스케이스 모델은 비즈니스 유스케이스 다이어그램, 텍스트 설명(textual descriptions) 그리고 유스케이스를 위한 마크로 액티비티 다이어그램으로 구성되고, 비즈니스 객체 모델은 상세 비즈니스 활동다이어그램, 비즈니스 순차다이어그램(business sequence diagram), 비즈니스 클래스다이어그램을 포함한다[8]. 초기 개념적 데이터베이스 설계를 위한 기초로 그림1의 비즈니스 모델링 순서로 작업한다.

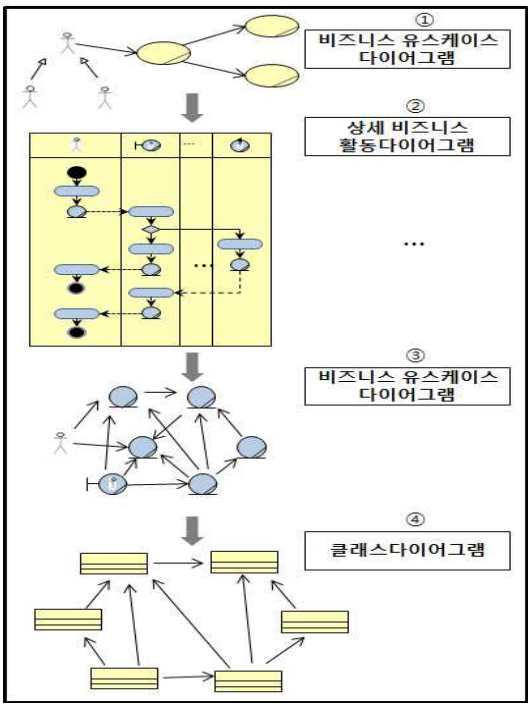

그림 1. 관계형 데이터베이스의 초기 개념 설계의 과정
Fig. 1. Process of the initial conceptual design of the relational database

1. 1단계 : 비즈니스 유스케이스 다이어그램 도출

1단계에서 비즈니스 유스케이스와 비즈니스 액터에 기반을 둔 비즈니스 유스케이스 다이어그램을 도출한다. 비즈니스 액터와 비즈니스 시스템 사이의 관계는 UML 연관으로 설계한다. 비즈니스 유스케이스 사이는 서로 다른 본질을 가질 수 있다. 기본 유스케이스의 행위가 다른 유스케이스의 행위에 의하여 확장이 가능한데 <<extend>> 키워드를 사용하여 확장할 수 있다. 또한 다른 유스케이스 행위를 끌어오는 것도 가능한데 <<include>> 키워드를 사용하여 포함할 수 있다. 비즈니스 액터는 다양한 역할과 관련된다.

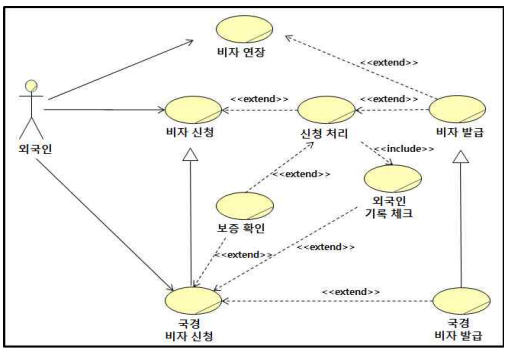

그림 2. 비지니스 유스케이스 다이어그램의 예
(비자발급시스템)
Fig. 2. An example of business use case diagram
(visa issuing system)

2. 2단계 : 상세 비즈니스 활동다이어그램 도출

비즈니스 유스케이스 다이어그램을 도출한 후에 확인된 모든 비즈니스 유스케이스를 문서화해야 하며, 활동의 순서에 관한 정의는 활동다이어그램을 사용한다[8].

그림3과 같이, 활동다이어그램은 사용자의 책임, 비즈니스 객체 사용 등에 관한 세부사항 없이 중요한 활동의 순서로 표현된다. 마크로액티비티 다이어그램은 비즈니스 객체, 비즈니스 객체흐름, 비즈니스 사용자책임을 이용하여 상세 비즈니스 활동다이어그램으로 변환된다.

스윔레인(swimlanes)은 활동의 중심을 지정한다. 각각의 스윔레인은 다이어그램 상에서 고유한 이름을 가지며, 궁극적으로 하나 이상의 클래스로 구현된다. 스윔레인으로 분할된 활동다이어그램에서 각각의 활동은 정확하게 하나의 스윔레인에 포함되지만, 변환은 여러 개의 스윔레인에 걸쳐 존재한다. 마크로액티비티 다이어그램과 활동다이어그램에 의한 비즈니스 유스케이스의 문서화 절차는 모든 비즈니스 유스케이스 확인을 위해 따라야 한다.

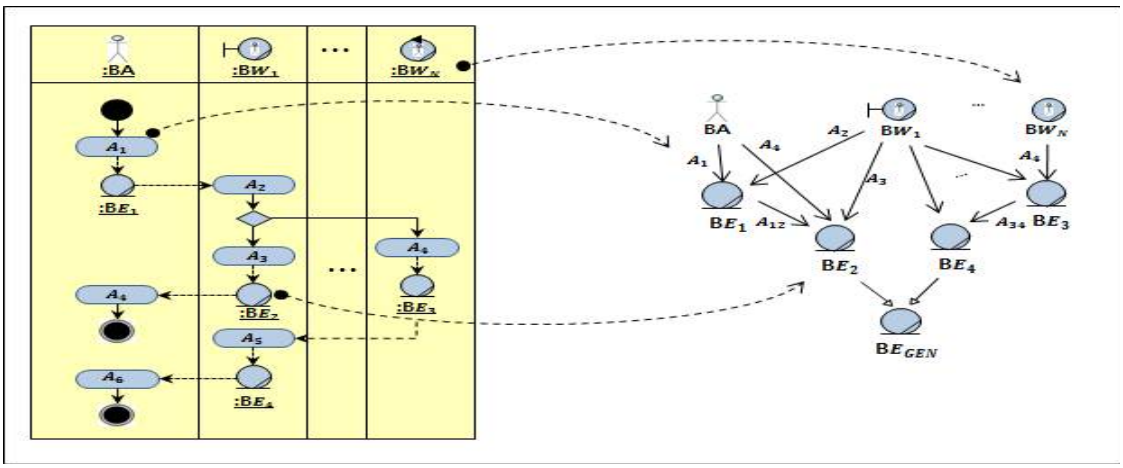

그림 4. 상세 비즈니스 활동다이어그램을
비즈니스 클래스다이어그램으로 변환
Fig. 4. Mapping of detailed business activity diagram into business class diagram

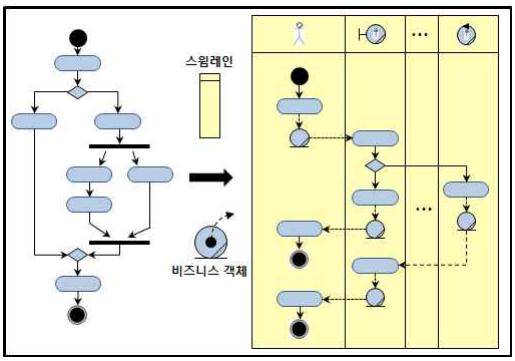

그림 3. 마크로액티비티 다이어그램을
상세 비즈니스 활동다이어그램으로 변환
Fig. 3. Migration of a macroactivity diagram into
detailed business activity diagram

3. 3단계 : 비즈니스 클래스다이어그램 도출

이번 단계에서는 상세 비즈니스 활동다이어그램은 비즈니스 클래스다이어그램으로 변환된다. 그래서 상세 활동다이어그램처럼 부분적으로 많은 클래스다이어그램을 가진다. 다시 말하면, 각각의 비즈니스 유스케이스를 위한 적절한 비즈니스 클래스다이어그램을 가진다.

그리고 다음 단계에서 전체 시스템의 정적 구조를 보여주는 통합 비즈니스 클래스다이어그램으로 변환된다. 상세 비즈니스 활동다이어그램은 초기 개념적 모델로서 비즈니스 클래스다이어그램 설계에 적합한 기준이다. 왜냐하면 비즈니스 객체, 객체흐름, 비즈니스 활동과 사용자책임 등, 확인된 유스케이스를 문서화했기 때문이다.

비즈니스 액터와 같은 세부적인 비즈니스 사용자와 비즈니스 객체는 비즈니스 클래스다이어그램을 위한 기초가 된다. 그림4와 같이, 확인된 모든 비즈니스 활동(business activites)은 적절한 다중성을 갖고 비즈니스 클래스로 직접 변환된다. 활동다이어그램이 클래스로 변환된 후에 객체들 사이의 연관을 확립해 주어야 한다. 연관은 첨부(attachment)뿐만 아니라 다른 유사한 비즈니스 엔티티에 적용된다.

그림5는 비즈니스 클래스다이어그램의 예를 보여준다. 일부 비즈니스 클래스는 상세 활동다이어그램으로부터 직접 변환된 결과이다. 비즈니스 액터는 '외국인'이고, 비즈니스 사용자는 '비자 발행인'이고, 비즈니스 엔티티는 '신청', '승인', '첨부'이다. 일부 비즈니스 엔티티와 관련하여 추가적인 연관(예를 들어, 신청 ↔ 첨부, 신청 ↔ 승인)이 필요하다.

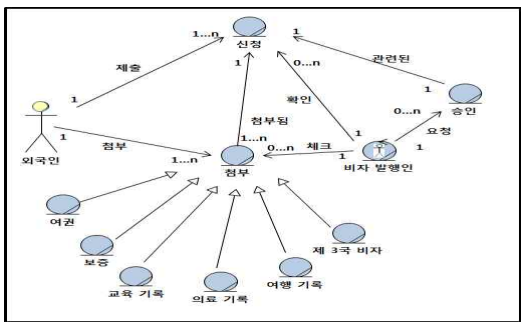

그림5. 비즈니스 클래스다이어그램의 예
(비자발급시스템)
Fig. 5. An example of business class diagram (visa
issuing system)

4. 4단계 : 클래스다이어그램 도출

비즈니스 클래스다이어그램은 대상이 되는 관계형 데이터베이스의 개념적 설계를 대표하는 클래스다이어그램으로 변환된다. 이 변환된 비즈니스 클래스는 <<비지니스 액터>>, <<비지니스 사용자>>, <<비지니스 엔티티>>와 같은 UML 클래스의 스테리오타입(stereotype)들이다. 그림6은 비자발급 시스템의 개념적 모델을 설명한다.

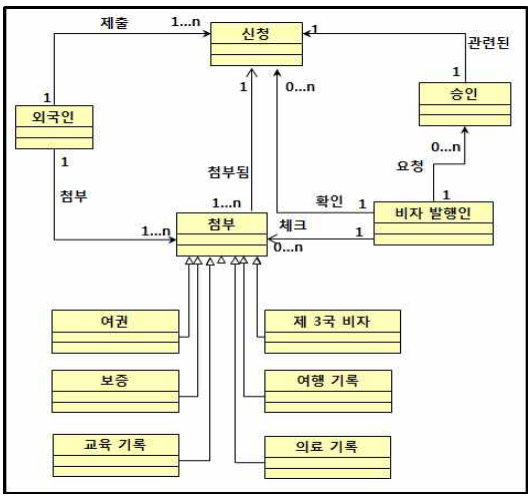

그림 6. 초기 개념모델의 예
(비자발급시스템)
Fig. 6. An example of initial conceptual model (visa issuing system)

IV. 개념모델의 추가적인 변환

초기 개념모델과 관련하여 초기 개념모델이 관계형 데이터베이스 설계를 위해 적합하지 않아 수정 작업이 이루어진다. 일반화의 목적과 데이터 검색 최적화 등을 위해 일부는 나누어지거나 통합된다. 연관도 비슷한 방식으로 이루어지게 된다.

초기 개념모델의 이러한 변환 작업은 일반화와 특수화와 연관된 수많은 집합의 존재, 비즈니스 역할과 활동과 관련하여 수많은 집합과 관련된 문제를 해결해야 한다.

1. 일반화와 특수화

일반화와 특수화 연관은 초기 개념모델의 첫 번째 변환 작업으로 이루어진다. 여기서 일반화 연관은 두 가지 근원을 강조하는데 첫째로, 일부 비즈니스 엔티티의 본성과 유사한 비즈니스 엔티티들의 집합과 관련된 것이다. 초기 개념 설계의 비슷한 성격과 비즈니스 엔티티의 모든 집합들은 원칙에 따라서 적합한 비즈니스 클래스다이어그램으로 변환된다. 이 경우에서 확인된 모든 비즈니스 클래스는 하위 종류 또는 일반화된 클래스의 특별한 유형이라 할 수 있다. 둘째로, 비즈니스 엔티티는 특정 비즈니스 엔티티와 관련된 수많은 첨부와 관련된 것이다. 예를 들어 비자 신청을 위한 첨부는 비즈니스 엔티티 일반화를 통해 설명된다.

첨부와 관련된 것들은 적절한 비즈니스 클래스로 설계된다. 일반화된 비즈니스 클래스는 원칙에 따라서 모든 첨부와 관련된 사이의 연관을 확립해야 한다(그림7의 ①번 참조). 일반화는 첨부와 관련된 총 수를 감소시키고 하나의 첨부 연관을 남긴다(그림7의 ②번 참조).

특정 시스템에 있어서 비즈니스 엔티티를 상세히 분석한 후에, 고객의 요구사항들을 그룹화 하여 전문화된 엔티티로 분류가 가능하다(그림7의 ③번 참조). 예를 들어, 첨부와 관련된 모든 것들은 전형적인 세 클래스(보증, 여행 목적, 다른 나라 여행 기록)에 의해 분류가 이루어진다. 첨부의 유형을 대표하는 클래스를 추가 생성할 수 있다.

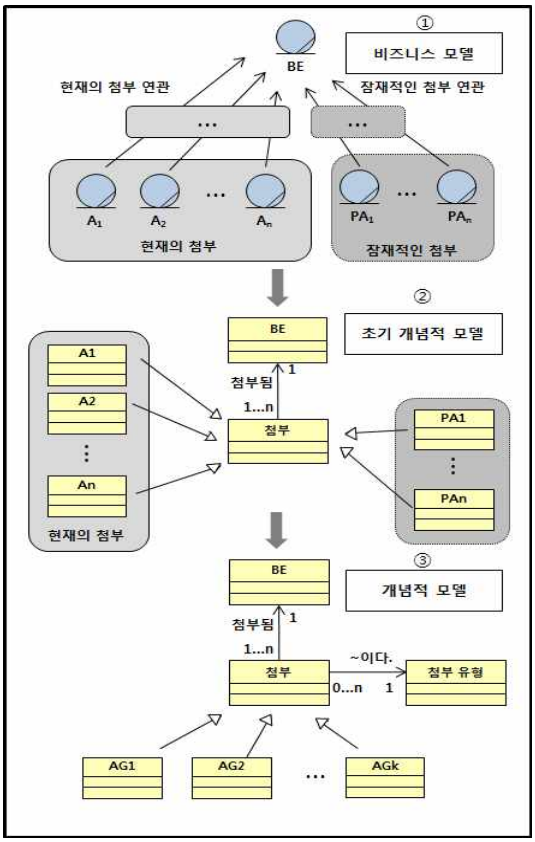

그림 7. 첨부 집합 일반화의 단순화
Fig. 7. Simplification of attachment set generalization

2. 역할과 활동

역할은 개념모델링의 기본 개념 중의 하나로 특정 사람 또는 위치의 행동과 관련된 목록으로서 데이터 모델링 분야에서 중요한 개념이다[9]. 객체지향 설계의 많은 클래스는 실제 클래스가 아닌 역할이라 할 수 있다(예를 들어, 승객과 구매자와 같이 특정 역할을 하는 사람). 그리고 동시에 여러 가지 역할을 가질 수도 있다. 가능한 행동은 기본 클래스의 인터페이스 추가를 통해 각 역할을 고려할 수 있다.

UML 비즈니스 프로화일은 스테리오타입 클래스 <<busines s worker>>로서 UML 메타모델에서 비즈니스 역할을 정의한다. 전체 시스템에서 실제 역할은 메타클래스(metaclass)로 설계된다.

예를 들어 비자 발행인은 비즈니스 역할을 대표한다. 비즈니스 시스템에서 동일한 사용자뿐만 아니라 다른 사용자로 지정될 수 있는 다른 비즈니스 역할의 집합이 있다. 따라서 이런 역할에 관하여 새로운 비즈니스 역할의 집합과 다른 사용자에게 재 할당 할 수 있다. 각 사용자는 '할당된 역할' 클래스에 의해 다른 역할을 가질 수 있다. '할당된 역할'과 '사용자' 클래스는 이중 연관이다.

'활동' 클래스는 특정 엔티티에 제공하는 비즈니스 활동의 추상적인 개념을 대표하는 집합이다. '완료된 행동' 클래스에 의해 설계되는 각 활동 수행은 '엔티티 상태' 클래스에 의해 설계된 객체의 상태를 변경시킬 수 있다. 이런 방식으로 비즈니스 프로세서 처리 과정에서 어떤 사용자에 의해 실행되는

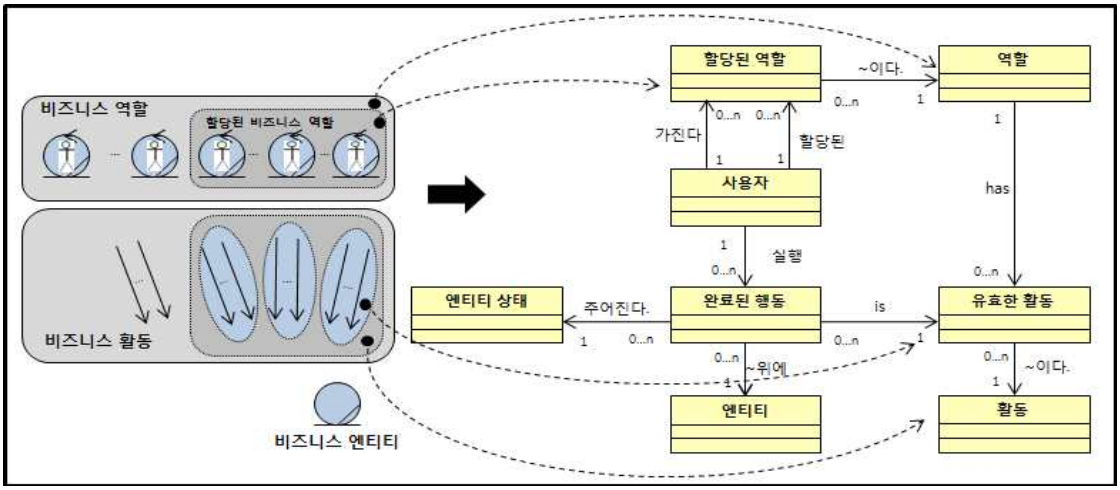

그림 8. 비즈니스 역할 다른 종류 집합의 단순화와 비즈니스 활동
Fig. 8. Simplification of heterogeneous sets of business roles and business activities

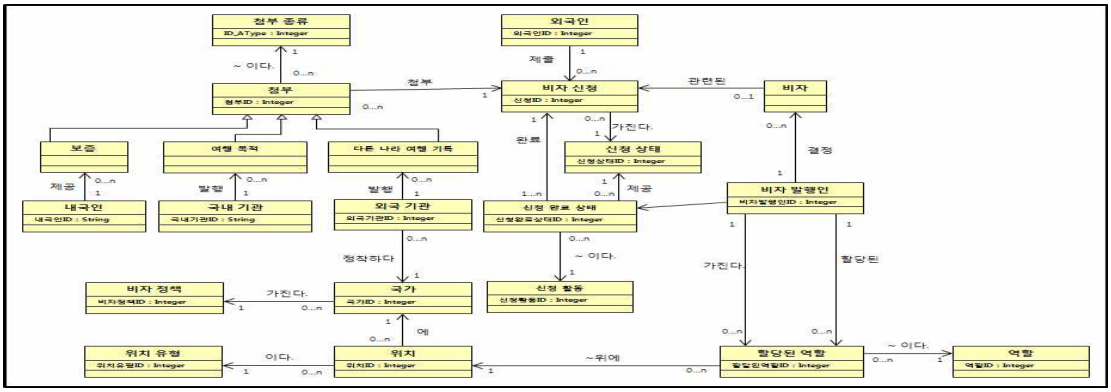

그림 9. 대응하는 관계형 데이터베이스의 개념적 모델(비자발급시스템)
Fig. 9. Target conceptual model of relational database (visa issuing system)

활동의 상태를 확인 할 수 있다. 특정 사용자에 의해 수행되는 활동의 제한을 만들려면 클래스 '역할'과 '활동' 사이의 연결을 대표하는 새로운 클래스 '유효한 활동'을 생성한다. 이런 것들이 비자발급시스템에 적용되어 그림8과 같이 표시된다.

3. 클래스 추가

이전 단계에서 초기 클래스다이어그램이 재구성 되었다. 초기 클래스다이어그램이 일부 존재하지만 특정 모델링 패턴의 유연성과 데이터 모델링을 위해 새로운 클래스를 생성한다. 이 변환은 속성의 확인과 데이터 모델 표준화에 연관되어 있다. 예를 들어 '국가', '위치 유형', '비자 정책' 클래스이다. 대상이 되는 데이터베이스가 일부 다른 데이터베이스와 연결되어 있을 경우 데이터베이스 연결 지점을 대표하는 클래스를 설명해야 할 것이다. 예를 들어 '외국 기관', '내국인', '국내 기관' 클래스이며 그림9에서 확인할 수 있다.

4. 중복 연관

마지막으로 클래스다이어그램 추가 변환 작업의 중복 연관이 데이터베이스 검색과 일부 엔티티 상태 정보를 빠르게 확인하기 위한 내용도 포함되어 있다. 중복 클래스는 데이터베이스정규화를 방해하지만 지금과 같은 경우에는 전체적으로 데이터베이스 성능을 향상시키게 된다. 그림9는 초기 클래스다이어그램 적용 원칙의 결과를 설명한다.

V. 클래스다이어그램을 관계형 데이터베이스 스키마로의 변환

이전 단계에서 도출된 클래스다이어그램을 통해서 관계형 데이터 모델링으로 변환이 가능하다. 데이터베이스 스키마 변환 과정에서 정보 구조로부터 논리적 개념을 이용하여 논리적 데이터 구조로 표현하는 것이 필요한데 이 변환 과정을 데이터 모델링이라 한다[10]. 표1을 바탕으로 UML 클래스다이어그램을 이용하여 관계형 데이터베이스 스키마 변환 작업을 수행 할 수 있다[11].

표 1. 관계형 데이터베이스 스키마의 변환 방법
Table 1. Method for transformed to Relational database schema

① 클래스는 테이블이 됨. ② 클래스의 속성(attribute)은 테이블의 열(column)이 됨. ③ 클래스의 속성 타입은 테이블의 열 타입이 됨. ④ 일반화가 없는 클래스를 위해서는 integer 기본키를 생성하고, {oid}를 위해서는 기본키 제약 조건에 {oid} 태그 열을 추가함. ⑤ 자식 클래스(Subclasses)들은, 각 부모 클래스의 키를 기본키와 외부키 제약 조건에 추가함. ⑥ 속성에 {nullable} 태그가 있으면 테이블 속성에 NULL 또는 NOT NULL을 추가함. ⑦ 속성이 초기 값을 가지면, 열에 DEFAULT 문을 추가함. ⑧ 연관 클래스들은, 각 역할-실행 테이블에 대한 기본키를 기본키와 외부키 제약 조건에 추가함. ⑨ 만일 {alternate oid = <n>} 태그이면, UNIQUE 제약 조건에 대한 열을 추가함. ⑩ 각 명시된 제약에 대해 CHECK를 추가함. ⑪ 0..1, 1..1 규칙의 연관 관계에서 참조하는 테이블에 외부키를 생성함. ⑫ 집합 테이블(CASCASE와 같이)의 외부키를 갖는 복합 집합을 위해서 기본키를 생성한다 : 기본키를 위해 추가적인 열(column)을 추가함. ⑬ 이진 연관 클래스를 적당한 "N"쪽 테이블로 이동함으로써 최적화 함. ⑭ 연관 클래스가 아닌 3원 연관은 N : N에 대한 테이블로 생성함. ⑮ N : N, 3원 연관에서 역할-실행 테이블의 키로부터 기본키와 외부키 제약 조건을 생성함. ⑯ 연관 클래스 없는 다대다(many-to-many) 연관을 위해 기본키와 외부키를 생성함.① 클래스는 테이블이 됨. ② 클래스의 속성(attribute)은 테이블의 열(column)이 됨. ③ 클래스의 속성 타입은 테이블의 열 타입이 됨.

그림9에 의한 '외국인' 객체는 표1의 관계형 데이터베이스 스키마 변환 방법 ①과 ②번의 성질에 따라 '외국인ID' 객체 타입 속성을 저장하는 '외국인' 테이블로 테이블 스키마는 다음의 표2와 같다.

표2. 외국인 테이블 스키마
Table 2. Foreigner table schema

```
CREATE TABLE 외국인(
외국인ID Integer PRIMARY KEY );
```

그림9에 의한 '비자' 객체는 표1의 관계형 데이터베이스 스키마 변환 방법 ①과 ② 및 ④번의 성질에 따라 '비자ID' 객체 타입 속성을 저장하며 변환 방법 ⑪에 의해서 '비자' 테이블 스키마는 다음의 표3과 같다.

표3. 비자 테이블 스키마
Table 3. Visa table schema

```
CREATE TABLE 비자(
비자ID Integer PRIMARY KEY
신청ID Integer REFERENCE 비자 신청
비자발행인ID Integer REFERENCE 비자 발행인
CONSTRAINT 비자_PK PRIMARY KEY(비자ID, 신청ID, 비자발행인ID) );
```

그림9에 의한 '보증' 객체는 표1의 관계형 데이터베이스 스키마 변환 방법 ①과 ② 및 ④번의 성질에 따라 '보증ID' 객체 타입 속성을 저장하며 변환 방법 ⑤와 ⑪에 의해서 '보증' 테이블 스키마는 다음의 표4와 같다.

표4. 보증 테이블 스키마
Table 4. Guarantee table schema

```
CREATE TABLE 보증(
보증ID Integer PRIMARY KEY
첨부ID REFERENCE 첨부
내국인ID REFERENCE 내국인
CONSTRAINT 보증_PK PRIMARY KEY(보증ID, 첨부ID, 내국인ID) );
```

변환 방법과 정규화 과정을 통하여 변환된 관계형 데이터베이스 스키마의 나머지는 다음의 표5와 같다.

표 5. 관계형 데이터베이스 스키마
Table 5. Relational database schema

```
CREATE TABLE 비자 신청(
신청ID Integer PRIMARY KEY
외국인ID Integer REFERENCE 외국인
첨부ID Integer REFERENCE 첨부
비자ID Integer REFERENCE 비자
신청완료상태ID Integer REFERENCE 신청 완료 상태
신청상태ID Integer REFERENCE 신청 상태
CONSTRAINT 비자신청_PK PRIMARY KEY(신청ID,첨부ID, 비자ID, 외국인ID, 신청완료상태ID, 신청상태ID) );

CREATE TABLE 신청 상태(
신청상태ID Integer PRIMARY KEY
신청ID Integer REFERENCE 비자 신청
신청완료상태ID Integer REFERENCE 신청 완료 상태
CONSTRAINT 신청상태_PK PRIMARY KEY(신청상태ID, 신청ID, 신청완료상태ID) );
...
CREATE TABLE 신청 완료 상태(
신청완료상태ID Integer PRIMARY KEY
신청ID Integer REFERENCE 비자 신청
신청상태ID Integer REFERENCE 신청 상태
비자발행인ID Integer REFERENCE 비자 발행인
신청활동ID Integer REFERENCE 신청 활동
CONSTRAINT 신청완료상태_PK PRIMARY KEY(신청완료상태ID, 신청ID, 신청상태ID, 비자발행인ID, 신청활동ID) );
```

```
CREATE TABLE 위치(
위치ID Integer PRIMARY KEY
위치유형ID REFERENCE 위치유형
CONSTRAINT 위치_PK PRIMARY KEY(위치ID, 위치유형ID) );
```

VI. 결 론

본 논문에서는 UML을 토대로 유스케이스 중심의 객체지향 분석·설계를 통한 초기 개념모델 설계, 개념모델의 추가적인 변환, 그리고 클래스다이어그램을 관계형 데이터베이스 스키마로 변환까지, 객체지향 분석·설계 방법론을 기반으로 한, 관계형 데이터베이스 설계 방법론을 통합적으로 다루었다.

본 통합 모델에서 사용한 개념모델은 비즈니스 유스케이스를 기반으로 비즈니스 활동다이어그램으로 변환되고, 비즈니스 활동다이어그램을 기반으로 비즈니스 클래스다이어그램, 그리고 초기 개념모델을 대표하는 클래스다이어그램으로 변환하였다. 도출된 클래스다이어그램에 일반화와 특수화, 역할과 활동, 클래스 추가 그리고 중복 연관에 따른 추가적인 변환 작업을 진행하였고, 최종적으로 관계형 데이터베이스 스키마 변환 작업까지 진행하였다.

본 논문에서 제안한 방법론을 적용함으로써 객체지향 분석·설계 방법론과 관계형 데이터베이스 설계 방법론을 통합적으로 다룰 수 있게 되었다. 이에 따라 대다수의 관계형 데이터베이스 기반의 정보시스템을 보다 용이하게 구축할 수 있다.

참고문헌

[1] Kim Yeong-gyu and Yang Hae-sul and Choi Hyeong-jin, "Framework Model for Software Productivity Enhancement In Object-Oriented Environment", Journal of The Korea Academia-Industrial Cooperation Society, 9권 6호, pp.1678-1689, 2008.

[2] Cho Wan-su, "UML 2 & UP Object-Oriented Analysis&design", pp.189-205, Hongrung Publishing Company, Seoul, 2005.

[3] Drazen Brdjanin and Slavko Maric, "An Example of Use-Case-driven Conceptual Design of Relational

Database", EUROCON 2007 The International Conference on "Computer as a Tool", Warsaw, September 9-12, 2007.
[4] J. Rumbaugh and I. Jacobson and G. Booch, "The Unified Modeling Language Reference Manual", Addison-Wesley, 1999.
[5] I. Jacobson and M. Ericsson and A. Jacobson, "The Object Advantage - Business Process Reengineering with Object Technology", ACM Press, Addison-Wesley, 2011.
[6] Gung Sang-hwan, "UML Based Software Design Methodology for Software Project Class", KAIS Fall Conference 2008, pp.250-253, 2008.
[7] OMG, UML 1.4 Specification, http://www.uml.org, 2001.
[8] D. Brdjanin and S. Maric, "UML-business profile-based Business Modeling in Iterative-Incremental Software Development", Proc. of EUROCON 2005 - Int. Conf. on "Computer as a tool", pp. 1263-1266, 2005.
[9] C. Bachman and M. Daya, "The role concept in data models", Proc. of 3rd Int. Conf. on VLDB, pp. 464-476, 1977.
[10] Cho Jeong-gil, "An Efficient Transformation Technique from Relational Schema to Redundancy Free XML Schema", Korean Society for Internet Information 11권 6호, pp.123-133, 2010.
[11] Bang Seung-yun and Joo Kyung-soo, "Design Methodology for XML Schema Application based on UML.", Soonchunhyang Univ, pp.71-75, 2003.
[12] Choi Ji-Woong and Kim Myung-ho, "OWL/Relational Mapping Rules to Use Relational Databases as OWL 2 Web Ontologies", Korea Society of Computer Information, Vol. 16, No. 7, pp.35-47, 2011.
[13] Lee Soon-mi, "Design of Relational Storage Schema and Query Processing for Semantic Web Documents", Korea Society of Computer Information, Vol. 14, No. 1, pp.35-45, 2009.

저 자 소 개

주 경 수
1993 : 고려대학교 전식학과 공학박사.
현 재 : 순천향대학교 컴퓨터소프트웨어공학과 교수
관심분야 : Database System, XML, System Integration, Object Oriented System
Email : gsoojoo@sch.ac.kr

조 도 형
2010 : 순천향대학교 컴퓨터학과 공학사.
현 재 : 순천향대학교 컴퓨터소프트웨어공학과 석사과정
관심분야 : Database System, Object Oriented System, UML
Email : jhodohyung@sch.ac.kr