E-R 모델과 자동생성기를 이용한 응용시스템의 구축 과정에 관한 연구

정 일 주*

Research on the Process of Constructing Application Systems Using the E-R Model and an Automated Application Generator

Ilchoo Chung*

Abstract

This paper makes an attempt to suggest a process of automatically generating application software based on the Entity-Relationship model : 1. The designer develops an E-R model of an real-world system. 2. The designer inputs the entity and relationship types, and attributes shown in the E-R model, and also the basic operations of the application system to the software generator. 3. The application generator produces database schema and link information between application programs, and then automatically generates a stereo-type application system. In order for the automated application generator to build the application system in a systematic way, four basic program generation rules have been suggested. A set of computer programs have been developed in order to show the applicability of the automated software generation process suggested in this paper. By following each rule with the generator, the designer can build an application in an efficient manner compared with traditional (manual programming) approaches. It has been demonstrated from the case study that the idea of applying an automated generator in systems development based upon the E-R model is feasible.

Keywords : Entity-Relationship Model, Automatic Generation of Application Software, Application Generator, Software Construction

논문접수일 : 2004년 12월 16일 논문게재확정일 : 2005년 12월 6일

※ 이 논문은 2004년도 홍익대학교 학술연구 조성비에 의하여 연구되었음.

* 홍익대학교 상경대학 상경학부 교수, (339-700)충남 연기군 조치원읍 339-700
Tel : 041-860-2362, Fax : 041-862-2974, e-mail : chungic@hongik.ac.kr

1. 연구의 배경과 목적

개체관계(E-R) 모델은 1970년대 중반 P. Chen에 의해 제안된 이래 시스템과 데이터의 논리적 구조를 표현하는 유용한 방식으로 광범위하게 이용되어 왔다[Chen, 1976 ; Francett, 1994 ; McFadden, 1994]. E-R 모델은 관계형 데이터베이스, 정보 모형, 요구분석 등 다양한 논리적 체계의 표현 도구로서 활용되고 있고, 최근에는 CASE 도구, 추론적 모델링, 전문가 시스템 및 객체지향 모델 등의 분야로 연구와 적용 범위가 확대되고 있다[정일주, 1999 ; McFadden, 1994 ; Gane, 1990].

E-R 모델이 표현하는 시스템의 정적(靜的)인 특성과 시스템의 행태를 표현하는 프로그램 사이에는 체계적인 연관성이 존재한다는 사실은 전자로부터 후자를 생성하는 일관성 있는 원칙과 절차에 의해 설득력을 얻고 있다[정일주, 2002]. 이러한 연관성에 입각하여 E-R 모델을 통하여 소프트웨어의 생성 과정을 자동화 해 보려는 시도는 다양한 각도에서 접근되어 왔다[Francett, 1994]. Siau 등은 일반화, 추상화 등을 지원하는 EERD(Enhanced E-R Diagrammer)를 개발하였다[Siau et al., 1992].

E-R 모델에 기초한 대표적인 CASE 도구들 중의 하나인 Logic Works의 ERwin은 초기의 데이터 구조의 설계에서 데이터베이스의 스키마, 뷰, 트리거, 저장프로시저 등을 자동으로 생성하고, Microsoft의 Visual Basic을 이용한 클라이언트 화면 생성, Powerbuilder, Clipper, FoxPro 등과의 인터페이스 제공 등의 방향으로 진화하고 있다[Stoughton, 1997]. 그러나 ERWin은 직접 응용 시스템을 생성하기 보다는, 이러한 응용시스템 개발도구에 스키마 정보를 자동으로 연계하여 데이터베이스를 응용 시스템 개발 프로세스에 통합시켜 소프트웨어 개발 생산성을 높이는데 중점을 두고 있다.

한편 InfoModeler에서는 ORM(Object Role Model)과 E-R 모델 사이의 자동변환을 지원하고 동시에 기존의 데이터베이스에서 E-R 모델을 도출하는 리버스 엔지니어링 기능으로 확장하고 있다[Shelly et al., 1995]. 지금까지의 E-R 모델과 CASE를 연계한 연구와 개발 동향을 종합해 보면 모델링과 데이터베이스 설계를 연계하면서 DFD 등 다른 분석 기법과 연결하는 방향이 주종을 이루고 있는 가운데 E-R 모델링과 Powerbuilder 등 Lower CASE 도구와의 연계와 통합을 시도하고 있다.

국내에서도 웹 어플리케이션의 개발에 있어서 인터페이스 테스트 자동화 기법이 연구되고, CBD 환경을 위한 웹 어플리케이션 개발, S/W 컴포넌트를 통한 조립 방법론 연구 등이 이루어지고 있다[권영호, 2003 ; 박병형, 2003 ; 정연대 et al., 2003].

이러한 배경에서 본 논문에서는 두 가지 방향에서 CASE의 핵심 주제인 소프트웨어 자동 생성을 E-R 모델을 중심으로 접근한다. 첫째, E-R 모델이 기능적으로 소프트웨어 자동 생성을 전제로 한 응용시스템의 모델링 역할을 할 수 있는지 그 가능성을 점검하고, 둘째, CASE 분야의 중요한 과제가 되어 온, Upper CASE와 Lower CASE로 분할되어 있는 CASE 도구들을 연결하는 과정을 생략하고 소프트웨어 생성을 보다 단순화 하는 방안을 탐색한다.

본 연구의 목적은 앞의 연구에서 제시된 E-R 모델에 기반을 둔 소프트웨어 자동생성 절차를 객체지향의 개념과 이를 실현하는 컴퓨터 언어를 사용하여 보다 구체화 하는 것이다.

특히 응용 시스템 중에서 데이터 조작 시스템과 관련된 기능을 중심으로 자동생성 과정을 구현하고, E-R 모델로 표현된 시스템 설계 데이터를 객체지향 개념으로 체계적으로 조직화하

는 데에 주안점을 둔다. 본 논문의 2장에서는 E-R 모델과 응용시스템 사이의 관계에 기초하여 응용시스템을 체계적으로 생성할 수 있는 자동 생성 원칙을 제시하고 3장에서는 자동 생성 시스템의 설계 그리고 4장에서는 자동 생성 시스템의 구현 결과를 설명한다.

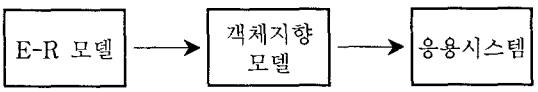
2. E-R 모델과 응용시스템 자동 생성의 원칙

2.1 접근 방법

E-R 모델은 실제 세계의 의미를 상호 연결된 개체/관계 타입, 속성 및 값타입 등의 개념 체계를 통해 표현하고 있다. E-R 모델은 실제 세계의 의미를 잘 정립된 개념 체계를 통해 제공함으로써 그것을 소프트웨어 생성 과정에 조직적으로 적용할 수 있는 가능성을 높여 주고 있다. E-R 모델이 제공하는 의미적 풍부함을 응용시스템의 정적 요소인 데이터베이스와 이를 동적 요소와 결합시켜주는 컴퓨터 프로그램에 조직적으로 반영하고자 하는 데에서 소프트웨어 자동 생성의 단서를 찾을 수 있다.

E-R 모델에 존재하는 각각의 개체/관계 타입은 반복되는 하나의 "프로그램 패턴" 이라는 관점으로 이해할 수 있다. 그렇게 보면 어떤 대상을 표현하는 E-R 모델 전체는 특별한 규칙에 의해 표현된 하나의 응용시스템 패턴이다. 따라서 E-R 모델의 상호 연결된 개체/관계 타입에 일련의 행동적 요소를 결합시키고 이를 일정한 언어적 방식으로 표현함으로써 우리는 컴퓨터 프로그램의 기본적 두 요소인 데이터와 그 데이터의 행동적 특성을 나타낼 수 있다.

한편, 객체지향 모델은 현재 응용시스템의 실현을 위한 가장 강력한 표현 방식으로 보편화되어 있다. 객체지향 모델로 잘 표현된 현실세계는 역시 객체지향 개념에 기반을 둔 언어에 의해 응용시스템으로 효율적으로 변환될 수 있다. 연구 결과에 의하면 E-R 모델과 객체지향 모델 사이에는 밀접한 연관성이 존재한다. E-R 모델은 일관성 있게 객체지향의 개념으로 표현될 수 있다[Chan et al., 1998]. 따라서 기본적으로 객체지향 방식이 제공하는, 예컨대, UML과 같은 풍부한 설계 언어 및 도구들 그리고 C++ 및 Visual Basic.Net 등 기존 컴퓨터 언어를 활용할 수 있도록 접근 방향을 설정한다. 즉, <그림 1>과 같이 E-R 모델을 먼저 객체지향 모델로 변환한 다음 이를 응용시스템의 소프트웨어로 변환하는 두 단계의 접근 방식에 기초하여 보다 구체적으로 응용시스템의 자동 생성 과정을 설계해 나아간다.



<그림 1> E-R 모델을 이용한 응용시스템 생성 접근방식

2.2 E-R 모델을 이용한 응용시스템 생성 원칙

E-R 모델로부터 응용시스템의 업무 프로그램이 자동적으로 생성되기 위해서 응용 프로그램의 특성과 E-R 개념 사이의 연계성을 설정하는 일정한 규칙이 필요하다. 먼저 가장 기본적인 원칙으로서 개체/관계 타입을 별도의 프로그램으로 보는 관점에서 <규칙 1>이 제안된다.

<규칙 1> : E-R 모델에 존재하는 각각의 개체타입과 관계타입은 클래스로 변환되고 이를 독립된 응용 프로그램으로 실현한다.

개체타입은 객체지향 개념의 "클래스(Class)"와 그리고 개체는 "객체(Object)"와 거의 일-대-일로 매핑이 된다. 그러나 관계타입은 E-R 다이어그램에서 표현된 개체타입과 관계타입 사이에 존재하는 선(Arc)을 어떻게 보는가에 따

라 하나 이상의 매핑이 가능하다. 예를 들어, 선을 독립된 클래스로 표현할 수도 있지만 관계타입 클래스의 속성으로 매핑 할 수 있다.

개체/관계 타입의 모든 속성은 클래스의 속성이 되고 따라서 해당 응용 프로그램의 입출력 대상이 된다. 개체타입과 관계타입이 "클래스"로 매핑된 다음 "응용 프로그램"과 다시 매핑된다. 여기에는 기본적인 UI(User Interface)가 필요하며, 사용자가 개체와 관계들을 어떻게 활용할 것인가에 따라 다양한 실현이 가능하다.

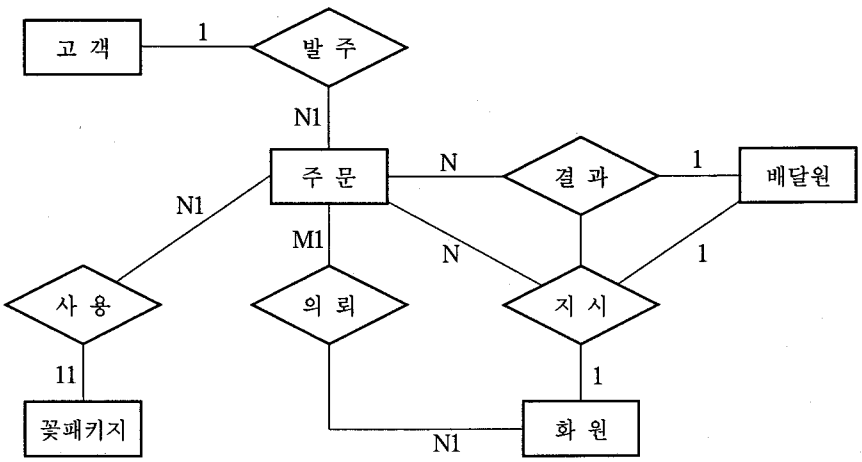
개체/관계 타입 클래스와 응용 프로그램과의 매핑에서 핵심적인 부분으로서 각각의 개체/관계 타입으로부터 클래스를 통해 매핑된 응용 프로그램의 "행동적 특성"이 규정되어야 한다. E-R 모델을 이미 주어진 것으로 본다면, 이에 적합한 그리고 필수적인 행동적 요소가 무엇인가 규정하여야 한다. E-R 모델이 표현하는 개체/관계 타입에 속한 개체/관계에 대한 가장 기본적인 행동 요소는 개체/관계의 생성과 소멸이다. 그리고 수정과 조회 역시 기본적인 행동 요소에 포함된다. 따라서 생성, 소멸, 수정, 조회의 네 가지 작업은 필수적인 행동 요소이다. 이들 요소 위에 다양한 비즈니스 로직이 존재할 수 있다. 다음의 규칙 2는 이러한 행동적 요소를 반영하기 위해 필요하다.

다음으로 각각의 개체/관계 타입으로부터 변환된 응용 프로그램의 행동적 특성이 규정되어야 한다. E-R 모델을 이미 주어진 것으로 본다면, 이에 적합한 그리고 필수적인 행동적 요소가 무엇인가 규정하여야 한다. E-R 모델이 표현하는 개체/관계 타입에 속한 개체/관계에 대한 기본적인 행동 요소는 생성과 소멸이다. 그리고 이에 추가하여 수정과 조회와 같은 작업이 필요하다. 이 네 가지 작업은 필수적인 행동 요소이다.

<규칙 2> : 개체/관계 타입으로부터 생성되는 업무 프로그램은 등록, 삭제, 수정 및 조회의 네 가지 작업을 실현할 수 있도록 한다.

위의 두 규칙의 예를 들어 보자. <그림 2>는 가상의 꽃배달 업무를 E-R 모델로 표현한 것이다.

위의 <규칙 1>과 <규칙 2>를 적용하면 <표 1>과 같이 10 개의 응용 프로그램 그룹이 생성된다. 그러나 개체와 관계타입 10개에 대하여 각각 조회(I), 추가(A), 삭제(D) 그리고 수정(M) 프로그램으로 세분화하면 응용 프로그램의 수는 모두 40개가 된다.

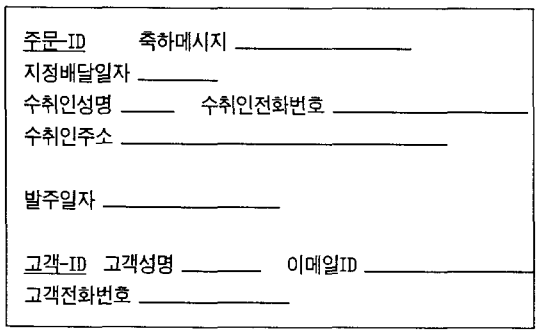


〈그림 2〉 꽃배달 시스템의 E-R 모델

〈표 1〉 개체/관계 타입으로부터 생성되는 프로그램

E-R 타입 프로그램	업무 프로그램 기능			
	조회	추가	삭제	수정
고 객	고객-I	고객-A	고객-D	고객-M
주 문	주문-I	주문-A	주문-D	주문-M
꽃패키지	꽃패키지-I	꽃패키지-A	꽃패키지-D	꽃패키지-M
화 원	화원-I	화원-A	화원-D	화원-M
배 달 원	배달원-I	배달원-A	배달원-D	배달원-M
발 주	발주-I	발주-A	발주-D	발주-M
사 용	사용-I	사용-A	사용-D	사용-M
의 뢰	의뢰-I	의뢰-A	의뢰-D	의뢰-M
지 시	지시-I	지시-A	지시-D	지시-M
결 과	결과-I	결과-A	결과-D	결과-M

위의 프로그램 목록에서 관계타입 "발주"로부터 생성된 프로그램 "발주-I"는 <그림 3>과 같은 사용자 화면을 가질 수 있다. 이 화면에서 관계타입 "발주"의 속성인 주문-ID, 발주일자 그리고 고객-ID는 수정, 삭제, 추가의 대상이 되지만 이 관계타입과 연관된 두 개체타입의 속성들은 오직 "조회" 목적으로만 제시된다.



〈그림 3〉 개체/관계 타입으로부터 생성되는 응용 프로그램

위의 두 규칙에 의해서 E-R 모델의 모든 개체/관계 타입들이 응용시스템에 일-대-일로 포함된다. 그러나 이들 사이의 상호작용 즉, E-R 다이어그램에서 개체타입과 관계타입을 연결하는 선에 대한, 즉 개체/관계 타입 사이의 운행에 관한 정보는 규칙 1과 2에 의해 변환되는 응용 시스템에는 아직 포함되지 못하고 있다. 다시 말해, 응용 시스템은 E-R 다이어그램에 있는 관계타입의 실체인 "관계"들을 통해 네트워크로 연결된 개체들 사이의 연결 관계가 유지되고 관리될 수 있는 기능을 제공해야 한다.

<규칙 3> : 응용시스템은 한 개체타입의 개체로부터 관계를 통해 E-R 다이어그램의 연결된 다른 개체로 접근할 수 있는 기능을 제공한다.

E-R 모델에서 표현하는 개체/관계들 사이의 접근을 실현하기 위한 설계에는 매우 다양한 가능성이 존재한다. 예를 들면, 한 가지 대안은 <규칙 1>에서 언급한 각각의 개체/관계 타입 응용 프로그램에서 운행을 위한 인터페이스 정보를 제공하고 개체/관계들 사이의 접근을 관리하는 별도의 응용 프로그램을 생성하는 것이다.

위의 세 가지 원칙에 의해 도출되는 응용시스템은 최소한의 기본적인 기능을 가진 것으로서 그 위에 다양한 비즈니스 기능이 탑재될 수 있다. 각 개체/관계 타입의 개별적 특성에 의존하고 있는 비즈니스 기능들은 자동 생성의 또 다른 차원의 고려와 접근이 필요하며 본 연구의 범위를 넘어서는 것이다. 다음으로 E-R 모델의

개체/관계 타입과 데이터베이스 객체의 관계를 정의한다.

<규칙 4> : E-R 모델에 있는 각각의 개체/관계 타입은 데이터베이스 요소로 정의된다.

만약 실제의 응용시스템 환경이 관계형 데이터베이스라면 개체/관계 타입 각각은 클래스로서 정의된 다음 관계형 데이터베이스의 테이블로 그리고 개체/관계 타입의 속성들은 각각 해당 테이블의 칼럼으로 생성된다.

3. 자동생성 시스템의 설계

3.1 응용 시스템의 범위

하나의 E-R 모델에 대해서 실현할 수 있는 기능은 실제 세계의 복잡성만큼 다양하다. 이들 기능들을 자동생성 시스템이 모두 만들어 낼 수 있는 것은 아니며 또 그럴 필요도 없다. 실용적인 관점에서 볼 때 어떤 E-R 모델에서 출발하여 자동 생성된 응용 시스템이 담당할 수 있는 기능의 범위는 크게 두 가지로 구분된다. 하나는 E-R 모델에 존재하는 개체/관계 타입들을 개별적인 것으로 보고 이들 각각에 대해 조회, 추가, 수정 및 삭제의 기본적인 처리를 위한 기능이 제공되어야 한다. 다른 하나는 E-R 모델에 존재하는 개체/관계 타입 상호간의 연결이 가진 의미를 응용 시스템이 구현할 수 있어야 하는 요구이다. E-R 모델의 연결된 모든 개체/관계 타입들 간에는 그 타입에 속한 개체/관계들의 상호 교류가 존재한다. 이들 간에 예컨대, A라는 사람이 구매한 B라는 꽃 패키지를 전달하는 C라는 직원이 존재한다. 이러한 개체/관계들 사이의 교류를 실현하기 위한 어떤 형태의 운행(Navigation) 체계가 존재해야 하며 이 기능을 응용시스템이 실현해야 한다.

3.2 개체타입과 클래스 설계

자동생성 시스템이 소프트웨어의 생성에 사용할 데이터는 기본적으로 E-R 모델로 표현한 시스템 설계 데이터인 대상 시스템의 E-R 모델이다. E-R 모델로 표현된 시스템 설계 데이터를 조직화하기 위해서는 E-R 모델의 메타모델이 필요하다. 이 메타모델을 표현하는 방식으로 객체지향 언어를 선택한다.

객체지향 시스템에서 E-R 모델은 기본적으로 개체/관계 타입 클래스들의 집합으로 모델링 될 수 있다. 아울러 속성(Attribute) 및 값타입(Value Type)에 대한 클래스가 고려될 수 있다. 이러한 접근은 E-R 모델의 기본이 되는 개념들이 객체지향의 핵심 개념들로 거의 일-대-일로 변환이 가능하다는 점에서 단순하다는 장점이 있다.

하나의 E-R 모델에 포함된 모든 개체타입은 각각 EntityType 클래스의 EntityType 객체

```
Public Class EntityType
        Inherits SerializableData
          Public EntityTypeName As String
          Public EntityID As Attribute()
          Public AttributeList As Attribute_Collection()
          Public RelatedRelationshipTypes As RelatedRelationshipType_Collection()
          Public Methods As ArrayList()
End Class
```

(Object)로 표현된다. 마찬가지로, E-R 모델의 연관타입은 RelationshipType 클래스의 객체로 모델링 할 수 있다. 다음의 EntityType 클래스의 예를 검토해 보자.

위의 클래스 선언에서 개체타입의 증명속성은 EntityType 클래스의 EntityID 속성이 된다. E-R 모델에 포함된 한 개체타입의 속성들은 EntityType 객체(Object)의 속성 집합(Attribute Collection)으로 변환된다. 각각의 속성은 다시 Attribute 클래스로 표현된다.

위의 EntityType 클래스 정의에서 Related-RelationshipTypes 항목은 이 개체타입과 연관된 연관타입을 의미한다. 한 개체타입은 복수의 연관타입과 연결될 수 있으므로 컬렉션(Collection) 타입으로 정의된다. E-R 모델을 클래스 모델로 변환하는 과정에서 연관된 연관타입을 어떻게 취급할 것인가에 대해서는 다양한 방법이 존재할 수 있다. 그러나 이를 EntityType 클래스에 포함시킴으로써 보다 효율적으로 응용시스템 프로그램을 생성할 수 있게 된다.

위의 예에서 EntityType 객체는(즉, E-R 모델의 개체타입은) ArrayList 타입의 Methods 속성을 갖는다는 사실에 주목할 필요가 있다. Methods 속성은 E-R 모델에는 포함되지 않은 것으로서 EntityType 객체의 생성과 유지 및 소멸 등 응용 프로그램의 동적 특성을 나타낸다. 시스템 설계자는 개체/관계 타입에 대해 Method 특성을 명기할 수 있다. 예를 들면, 꽃배달 시스템의 고객 UI(User Interface) 프로그램은 고객의 등록, 조회, 고객 정보 수정 등의 메소드를 필요로 한다.

EntityType 클래스의 메소드에는 프로그램 자동생성 과정을 지원하는 기능도 포함된다. 즉, 개체타입이 XML 파일로 처리되도록 Serializable-Data로부터 필요한 메소드를 상속으며 동시에 개체타입을 나타내는 객체의 XML 파일을 로드하고 저장하기 위한 메소드 그리고 개체타입 객체 전체를 가져오는 메소드들이 필요하다. 요약하면 다음과 같은 메소드들이 필요하다. (부록에 EntityType 클래스의 모든 속성과 메소드의 예가 주어져 있다.)

- 개체의 생성, 수정, 조회 및 삭제
- 개체/관계 타입 객체를 XML 형식으로 저장 및 탑재
- 한 개체/관계 타입에 속한 모든 개체/관계 객체의 탑재

부수적으로 개체타입 클래스 객체들의 메모리 안의 저장소로서 EntityType_Collection과 같은 컬렉션 클래스가 필요하다. 이 컬렉션에는 각각의 개체타입을 조회, 추가 혹은 삭제하는 메소드들이 정의된다.

3.3 연관타입과 속성의 클래스 설계

동일한 방식으로 연관타입과 연관타입 컬렉션에 대한 클래스 정의가 주어진다. 하나의 연관타입에는 복수의 개체타입이 참여한다. 양방향 연관타입에서는 두 개의, 그리고 3-방향 연관타입에서는 세 개의 개체타입이 참여하므로 이들 역시 컬렉션으로 정의되어야 한다. 연관타입 객체도 메소드 컬렉션을 갖는다.

관계타입 클래스 선언에서 주목할 부분은 ReadOnly Property로 선언된 RelationshipTypeID 즉, 관계 ID이다. 여기서 관계 ID는 해당 관계타입에 참여하는 개체타입의 증명속성들의 동적인 연결로 정의하고 있다. 마지막으로 다음과 같이 속성을 위한 클래스가 정의된다. 각각의 속성은 이름과 값타입을 가진다. 그리고 일련의 속성들은 역시 컬렉션으로 저장된다.

```
Public Class RelationshipType
    Inherits SerializableData
    Public RelationshipTypeName As String
    Public ReadOnly Property RelationshipTypeID() As String
        Get
            Dim eType As New RelatedEntityType()
            Dim r_ID As String = ""
            For Each eType In RelatedEntityTypeList
                r_ID += eType.anEntityType.EntityID.AttributeName & "_"
            Next
            r_ID = r_ID.TrimEnd("_")
            Return r_ID
        End Get
    End Property
    Public AttributeList As New Attribute_Collection()
    Public RelatedEntityTypeList As New RelatedEntityType_Collection()
    Public Methods As New ArrayList() . . .
End Class
```

```
Public Class Attribute
    Public AttributeName As String
    Public ValueType As String
    Public Visibility As String
    Public SingleMultiValued As String
End Class
```

위에서 Visibility 속성은 "Public", "Private", "Protected"와 같은 Attribute의 가시성을 나타낸다. ValueType 속성은 이 Attribute의 기본 데이터 타입을 나타낸다. SingleMultiValued 속성은 E-R 다이어그램에서 말하는 "값"의 Single/MultiValued 즉, 세트 여부와 관련된 것으로서 Set, Array, Collection 등으로 표현된다. ValueType과 SingleMultiValued 속성은 하나의 일반적인 타입으로 통합될 수도 있다.

속성 클래스에도 다양한 메소드가 선언될 수 있다. 예컨대 "주소"를 일정한 형식에 맞게 인쇄하는 메소드들을 포함할 수 있다. 아울러 ValueType 역시 클래스로서 정의될 수 있으나 그러나 여기서는 불필요한 복잡성을 피하기 위해 생략한다.

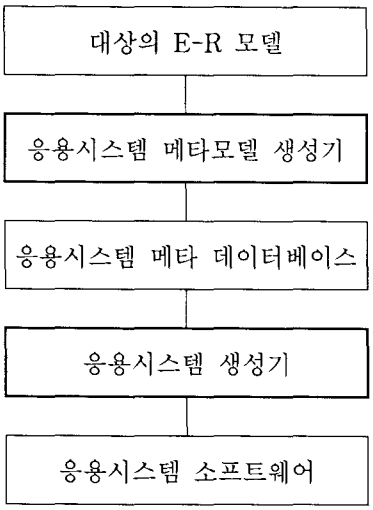
지금까지 설명한 개체/관계 타입 클래스는 응용시스템 메타 데이터베이스의 스키마가 된다. 구체적으로 이 클래스들은 생성 과정에서 XML 파일로서 실현된다.

3.4 응용 시스템의 자동 생성 단계

응용시스템의 자동 생성 단계는 <그림 4>와 같이 요약될 수 있다. 먼저 시스템 분석가는 시스템 구축의 대상이 되는 현실세계의 일부분에 대한 E-R 모델을 작성한다. 이때에는 E-R 모델의 기본 개념 요소들을 가지고 다이어그램을 그리고 텍스트로도 작성한다.

시스템 분석가는 그가 작성한 대상의 E-R 모델을 가지고 예를 들면, <그림 6>, <그림 7> 등에서 제시된 화면을 이용해서 메타데이터를

입력한다. 우리가 통상 E-R 다이어그램에서 표현하지 않는 "메소드" 부분을 추가로 입력한다. 그러면 "응용시스템 메타 모델 생성기"는 방금 전 설명한 EntityType, RelationshipType 및 Attribute 클래스로 표현된 E-R 메타 모델을 기반으로 하여 응용시스템의 메타 데이터베이스를 "자동적으로" 생성한다.



〈그림 4〉 응용 시스템 자동 생성 과정

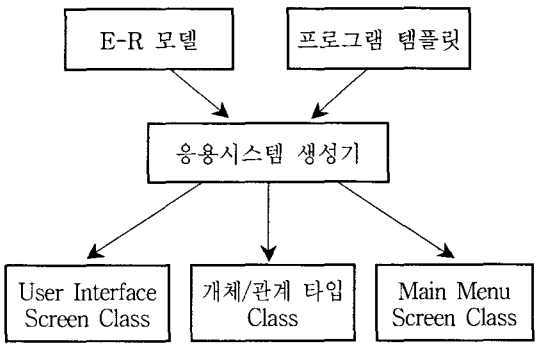
앞에서 생성된 "응용시스템 메타데이터베이스"는 XML 파일들로 구성된다. 여기에는 Entity type, Relationship type, Attributes, Methods 등을 구분하여 자동생성이 용이하게 진행되도록 구성된 파일들이 존재한다. 이러한 XML 파일들은 다음 단계에서 "응용시스템 생성기"에 입력된다.

마지막으로 "응용시스템 생성기"는 클래스로 표현된 응용시스템 메타 데이터베이스와 응용 프로그램 템플릿 등 다른 패러미터들을 사용하여 응용시스템 소프트웨어를 생성한다. 여기서 중요한 역할을 하는 것은 템플릿(Template)로서, E-R 개념을 표현한 XML 파일들과 세부적인 프로그램 코드들의 중간에서 자동 생성 로직을 구체적으로 지원한다.

3.5 응용시스템 생성기

대상에 대한 E-R 모델을 객체지향 개념의 클래스로서 표현한 응용시스템 메타 데이터베이스를 가지고 응용시스템 생성기는 응용시스템의 소프트웨어 즉, 프로그램들을 생성한다. 메타 데이터베이스는 일정한 소스 프로그램 템플릿(Template)과 함께 응용시스템 자동생성기에 입력된다. 자동생성기는 세 종류의 응용 프로그램을 생성한다. 객체지향 시스템에서는 각각의 응용 프로그램이 곧 클래스이기 때문에 실제로는 위의 각각에 상응하는 클래스들이 생성된다. 첫째, E-R 모델의 모든 개체타입과 관계타입 각각에 대해서 한 쌍의 사용자 인터페이스(User Interface) 화면 클래스, 둘째, 개체/관계 타입 클래스, 그리고 셋째, 응용시스템 전체에 대한 인터페이스를 제공하는 메인 메뉴 화면 클래스가 생성된다.

<그림 5>가 이러한 프로그램 자동 생성기의 입력과 출력 관계를 보여주고 있다. 여기서 프로그램 템플릿의 역할은 주어진 언어를 사용하여(여기서는 Visual Basic.Net을 사용함) 컴파일 되고 실행될 수 있는 프로그램의 세부 코드를 치환 가능한 형태 즉, 템플릿 형태로 제공하여 E-R 모델에서 제공하는 개체/연관 타입들을 변환할 수 있도록 한다.



〈그림 5〉 응용 시스템 소스코드 생성 작업

4. 자동생성 시스템의 구현

4.1 꽃배달 시스템

그러면 E-R 모델을 통하여 응용시스템을 설계하고 생성하는 시스템의 사례를 검토하기로 한다. 여기서 다룰 대상은 꽃배달 시스템이다. 이 시스템에 대한 E-R 모델은 <그림 2>에서 이미 완성된 것으로 가정하고 그 다음 단계에서부터 출발한다. <표 2>는 개체타입과 관계타입들의 속성 목록을 보여 주고 있다. 속성들 중에 밑줄이 그어진 것은 증명속성 즉, 그 타입에서 각각의 개체 혹은 관계를 유일하게 증명하는 속성을 의미한다.

4.2 응용시스템 메타 데이터베이스의 생성

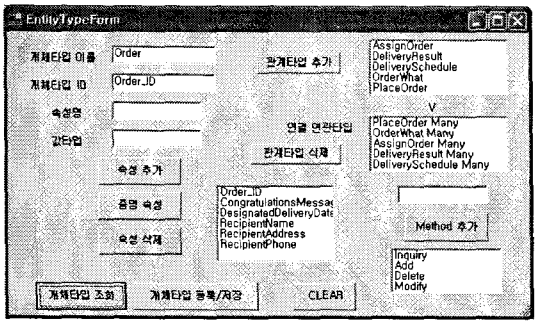
지금까지 설명한 응용시스템 자동 생성 시스템은 Visual Basic.Net 언어를 통해 구현된다. 먼저 대상에 대한 E-R 모델을 입력하는 단계로서 응용시스템 메타 데이터베이스 생성기가 실행된다. 이 프로그램에 의해 생성되는 결과는 앞서 제시한 메타 E-R 모델을 표현하는 개체/관계 타입 클래스에 정의된 구조로 생성되어 XML 파일로서 저장된다.

<그림 6>은 응용시스템 메타 데이터베이스 생성기 화면들 중에서 개체타입 입력 화면을 보여 주고 있다. 개체타입 입력 기능은 개체타입이 개체 클래스로서 정의되고 XML 파일로 생성될 수 있도록 해준다. 개체타입의 속성은 반복해서 입력할 수 있다. 개체 증명속성은 주어진 속성의 목록에서 선택한다. 개체타입과 연관된 연관타입들을 복수로 입력하기 위하여 E-R 모델에 포함된 모든 연관타입을 리스트로 제공하고 설계자는 목록에서 선택하도록 한다. 메소드

〈표 2〉 개체/관계 타입의 속성 목록

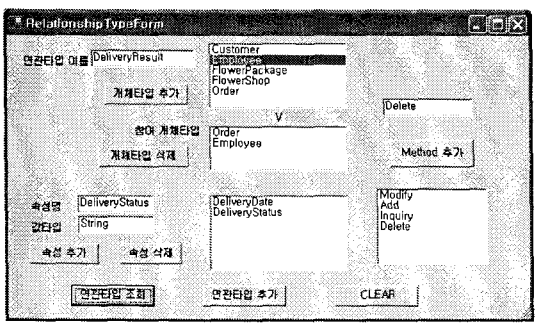
개체타입	속 성	값 타 입	관계타입	속 성	값타입
주 문	주문-ID 축하메시지 지정배달일자 수취인성명 수취인주소 수취인전화	문자 문자 일자 문자 문자 전화번호	발주	주문-ID 고객-ID 발주일자	문자 문자 일자
고 객	고객-ID 고객성명 이메일ID 고객전화번호	문자 문자 문자 전화번호	사용	주문-ID 꽃패키지-ID	문자 문자
꽃패키지	꽃패키지-ID 패키지이름 가격	문자 문자 숫자	의뢰	주문-ID 화원-ID 의뢰일자	문자 문자 일자
화 원	화원-ID 화원이름	문자 문자	지시	주문-ID 화원-ID 배달원-ID 배달지시일자	문자 문자
배 달 원	배달원-ID 배달원성명	문자 문자	결과	주문-ID 배달원-ID 배달결과 배달일자	문자 문자 문자 일자

역시 반복해서 입력할 수 있도록 하고 있다. 앞서 설명한 바와 같이 제시된 메소드는 E-R 모델에서 표현된 개체타입의 개체들의 추가, 조회, 수정 및 삭제와 함께 다른 개체/관계 타입으로 운행하는 절차들이다.



〈그림 6〉 개체타입 입력 화면

다음으로 <그림 7>에 주어진 바와 같이 연관타입을 입력하는 프로그램이 필요하다. 연관타입의 정의를 위해서는 시스템에 존재하는 모든 개체타입의 목록을 가지고 있어야 한다. 따라서 개체타입의 정의가 선행되어야 한다. 이 클래스는 연관 객체 각각을 표현하는 것이므로 여기서는 기수성을 나타내지 않는다. 참여하는 개체타입과 속성을 가질 뿐이다. 연관타입의 이름은 개체연관도의 마름모 부호 안에 있는 이름을 말한다.



〈그림 7〉 연관타입 입력 화면

마지막으로 개체/연관 타입의 연결을 나타내는 설계 정보가 입력되어야 한다. 이 정보는 개체연관도에서 보면 개체타입 부호인 직사각형과 마름모 사이의 연결선 즉, 아크(Arc)를 의미한다. 기수성은 이 화면에서 입력된다.

4.3 프로그램 소스 코드의 생성

응용시스템 소프트웨어의 자동생성의 마지막 단계로 소스 프로그램을 생성하는 작업을 시작한다. 앞서 제시한 생성 규칙들을 적용하여 시스템 구축의 단계를 구체적으로 진행한다. <규칙 2>는 시스템의 모든 개체/연관 타입에 대해 조회, 수정, 추가 및 삭제의 네 작업을 규정하고 있다. 물론 각 개체/연관 타입의 특성에 따라 이들 중에서 선택할 수 있다. <규칙 1>과 2는 각각의 개체타입에 대해 앞의 4 작업을 수행하는 프로그램 4 개를 생성하도록 규정하고 있다. 각각의 개체타입은 3 종류의 소스코드로 생성된다. (1) 사용자 인터페이스(UI) 화면 디자인, (2) 사용자 인터페이스(UI) 프로그램의 메소드 실행 소스코드, 그리고 (3) 클래스의 정의가 그것이다. 부록에 Customer에 대한 UI 프로그램과 클래스 정의 소스 코드 전체가 주어져 있다.

4.3.1 사용자 인터페이스 화면 프로그램 소스코드

각각의 개체타입에 대해서 하나씩 사용자 인터페이스(UI)를 위한 화면이 제공되는 프로그램이 필요하며 이를 위한 소스코드가 생성된다. UI 프로그램 소스코드는 기본이 되는 템플릿을 가지고 생성하는 방법을 택한다.

- UI 화면 역시 클래스이므로 클래스 정의로 시작된다. UI 화면의 명칭 즉, 클래스 명칭은 개체타입 명칭에 일반적으로 UI 화면을 의미하는 "Form"을 연결한 것으로 한다.
- 개체타입에 정의된 각각의 속성에 대해

TextBox와 Label 컨트롤 한 쌍이 정의된다. 컨트롤의 위치 소스코드에서 역시 지정하지만 나중에 변경할 수 있다.

4.3.2 UI 프로그램 메소드 실행을 위한 소스코드

해당 개체타입에 대해서 정의된 메소드에 대해서 하나씩 이벤트 핸들러가 생성된다. 본 연구에서는 <규칙 2>에 언급한 바와 같이 조회, 추가, 수정 및 삭제의 4 메소드 중에서 생성될 수 있다.

4.3.3 클래스 정의 소스코드의 생성

각각의 개체타입에 대해서 하나씩 클래스 정의 소스코드가 생성된다. 클래스 정의에는 해당 클래스의 속성과 함께 필요에 따라 다음과 같은 메소드들이 자동으로 생성될 수 있다.

- 해당 클래스의 객체 하나를 영구저장소에서 로드(Load)하는 메소드
- 해당 클래스의 객체 하나를 영구저장소에 저장하는 메소드
- 해당 클래스의 모드 객체를 메모리에 로드하는 메소드 등

4.4 데이터베이스의 구현

본 논문의 목적은 소프트웨어 자동 생성 시스템의 실현 가능성을 연구하는 데에 있기 때문에 논의를 단순하게 하기 위해 실제 DBMS를 사용하지 않고 PC의 디렉토리에 XML 파일로서 응용시스템 메타 데이터베이스와 응용시스템의 비즈니스 데이터베이스를 구현하는 방식을 택한다. MS 사의 닷넷(.NET) 환경에서는 이러한 설계가 ADO라는 컴포넌트를 통해 DBMS로 연결하는 방식으로 간단히 데이터베이스 연결 방식이 변환될 수 있다. 아울러 자동생성 절차의 단순성을 위해 <그림 2>의 E-R 모델에 있는 개체/관계 타입과 속성들의 명칭을 모두 영문으로 표현하였다.

다음은 <그림 2>에서 보여준 E-R 모델의 설계 중에서 Order(즉, 주문) 개체타입을 클래스로서 구현한 결과를 보여주고 있다. Order 개체타입의 구현을 위해서는 기본적으로 Order 클래스가 존재해야 하며, Order 객체들은 메모리(RAM)에서는 컬렉션에 저장되고 사용되지만, 영구적으로는 Order라는 디렉토리에 XML 파일로서 저장된다.

```
Public Class Order
    Inherits SerializableData
    Public Order_ID As String
    Public CongratulationsMessage As String
    Public DesignatedDeliveryDate As Date
    Public RecipientName As String
    Public RecipientAddress As String
    Public RecipientPhone As String
    Public RelaredRelationshipTypes As Arraylist
End Class
```

Order 클래스에 대해 "Inherits Serializable-Data" 구절이 주어진 것은 이 클래스를 XML 파일로 저장하고 가져오기 위해서이다. Order 클래스 정의에서 특히 유의할 부분은 Arraylist로서 선언된 연결된 관계타입 즉, Relared-RelationshipTypes 속성이다. 개체 Order와 연결된 모든 관계(Relationships)들은 이곳에 그 객체 ID에 해당되는 정보가 저장된다. 관계 하나는 XML 파일로 저장되며 이 XML 파일의 Path가 객체 ID를 포함하므로 실질적으로 객체 ID 즉, Object ID(OID)의 역할을 한다.

연관타입 PlaceOrder(발주)의 구현도 위와 비슷하다. 연관타입의 구현에는 예컨대 <그림 7>의 설계에 있는 Customer_ID와 Order_ID를 하나의 단순 속성으로 보는가 아니면(클래스의) 객체로 볼 것인가에 따라 두 가지 방법이 존재한다. 여기

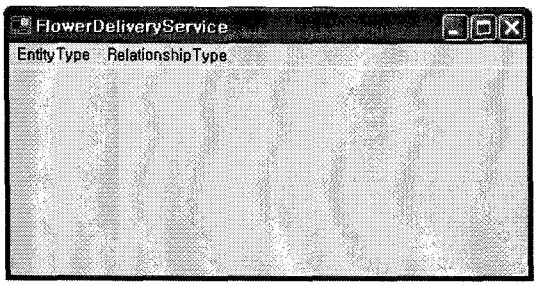
서는 후자를 선택하는 바, 객체지향 시스템의 이상적 형태인 모든 객체가 고유의 객체 ID(즉, OID)를 가지고 있는 상황에서는 이 방식이 더 직접적이기 때문이다. PlaceOrder 타입의 객체들 역시 별도의 폴더에 XML 파일로서 저장된다.

```
Public Class PlaceOrder
    Public aCustomer As Customer
    Public anOrder As Order
    Public OrderDate As Date
    Public RelatedEntityTypes As Arraylist
End Class
```

5. 응용 시스템의 실행

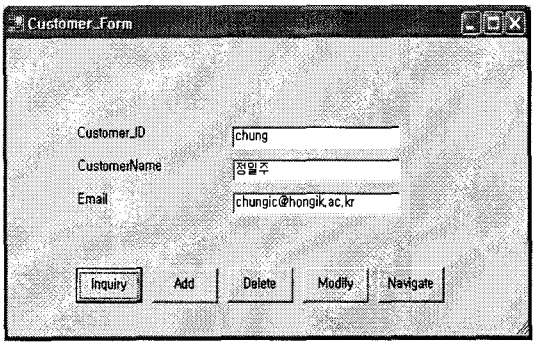
5.1 주 메뉴 화면과 개체/관계 타입 화면

이제 자동 생성된 응용 시스템을 시험한다. 먼저 메인 메뉴 프로그램이 실행되면 <그림 8>과 같은 화면이 사용자에게 주어진다. EntityType 메뉴를 선택하면 세부 메뉴 항목이 나타난다.



〈그림 8〉 메인 메뉴 화면

<그림 9>는 EntityType 메뉴의 세부 메뉴 항목들 중에서 Customer를 선택한 다음 나타나는 꽃배달 시스템의 고객(Customer) 화면을 보여주고 있다. 화면의 하단에는 Inquiry, Add, Navigate 등 다섯 가지 작업을 실행할 수 있도록 버튼을 제공하고 있다.



〈그림 9〉 Customer 처리 화면

개체타입 화면에서 한 개체를 처리 절차는 비교적 단순하다. 예를 들면 개체 하나를 추가하는 단계는 다음과 같다.

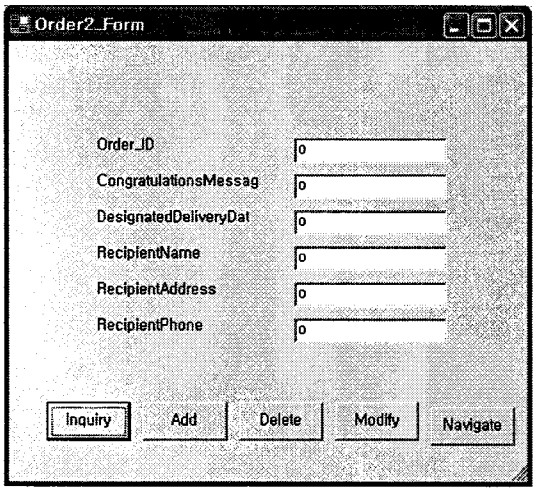
- 개체 E의 속성 값을 획득한다.
- 개체 E의 ID를 생성한다.
- 개체 E를 저장한다.

그러나 연관타입 화면의 경우는 참여하는 모든 개체들을 등록하고 처리해야하므로 개체타입 화면에 비해 절차가 복잡하다. 다음은 연관 하나의 등록 절차이다.

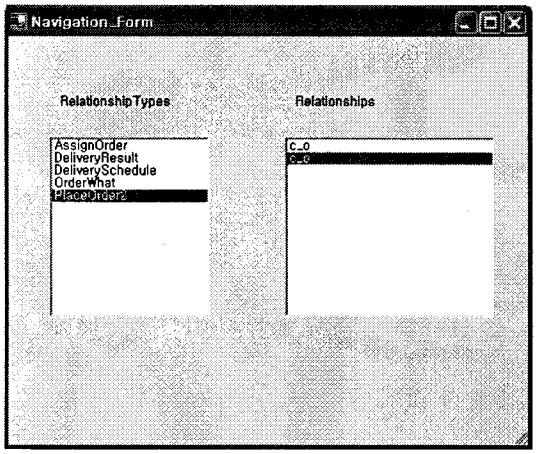
- 연관 R에 참여할 개체 1 즉, 이 연관에 참여하는 첫 번째 개체를 조회한다.
- 개체 1의 객체 ID를 획득한다.
- 연관 R에 참여할 개체 2 즉, 이 연관에 참여하는 두 번째 개체를 조회한다.
- 개체 2의 객체 ID를 획득한다.
  * n-방향 이상의 연관타입의 경우에는 개체 n까지 위 단계를 반복한다.
- 연관 R의 비 증명 속성을 지정한다.
- 연관 R의 ID 즉, 증명 속성을 생성한다.
- 연관 R을 등록한다.
- 개체 1과 2의 연관 R Collection에 R ID를 저장한다.

5.2 E-R 모델 상의 개체/관계 타입 공간의 운행

E-R 모델의 운행은 개체 혹은 관계타입 화면 어디서든지 시작할 수 있다. <그림 10>과 같이 하나의 개체를 검색한 다음 Navigate 버튼을 클릭한다.



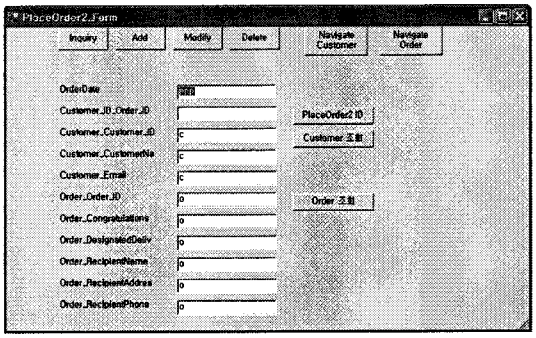
〈그림 10〉 Order 화면



〈그림 11〉 Customer 처리 화면

그러면 <그림 11>과 같이 Navigator 화면이 나타난다. 즉, 응용시스템은 이 Order 개체에 연결된 모든 관계를 보여준다. 여기서 사용자는 먼저 그가 계속 운행을 원하는 "관계타입"을 선택한다. Navigator는 선택된 관계타입에 속한 모든 관계들을 우측의 리스트에 보여준다. 사용자는 우측의 리스트에서 운행할 관계를 선택한다.

그러면 <그림 12>와 같이 PlaceOrder 관계를 보여준다. 여기서 다시 운행을 계속하려면 상단 우측의 Navigate 버튼을 클릭한다.



〈그림 12〉 Customer 처리 화면

6. 요약 및 결론

지금까지 본 논문에서는 E-R 모델로부터 출발하여 시스템 설계 데이터를 객체지향 모델로 표현하고 이를 프로그램으로 생성하는 과정을 제시하였다. 제안된 자동생성의 원리가 실제로 작동할 수 있는지 그 실현 가능성을 점검하기 위하여 가상의 꽃배달 시스템을 대상으로 하여 자동생성 과정의 실현 가능성을 점검해 보았다. 그 결과 응용시스템은 E-R 모델과 기존의 언어와 객체지향 방식을 통하여 상당히 효율적으로 생성될 수 있음을 알 수 있었다.

본 연구에서 제기되는 세 가지 이론적 이슈는 다음과 같다. 첫째, E-R 모델이 보유한 정보만으로 응용 프로그램에서 사용할 객체를 충분히 정의할 수 없으며, 이를 위해서는 E-R 모델에 대한 확장이 필요할 것이라는 사실이다. 다시 말해, UML 등의 객체 모델에서는 데이터의 구

조와 행위가 통합적으로 모델링되므로 기본적인 기능 수준에서의 응용 생성이 자연스럽지만 E-R 모델의 경우 데이터 측면만 모델링되어 이에 대한 보완이 필요하다는 것이다. 특히, 소프트웨어의 구현과 관련된 행동적 측면에서 E-R 모델의 확장이 필요하다고 본다.

둘째, 객체지향에서의 클래스 모델만으로는 응용 시스템에 대한 자동 생성이 불가능하며 시스템 행위에 대한 제어 로직, 비즈니스 로직 등이 포함되어야 한다. E-R 모델을 이용한 소프트웨어 자동 생성의 과정에서 이 부분이 조금 더 체계적으로 개념화되어야 할 것으로 생각된다. 셋째, 객체 모델에 존재하는 클래스들 사이의 관계 예를 들면, Inheritance, Aggregation 등이 E-R 모델을 이용한 소프트웨어 자동 생성 과정에 어떻게 도입될 수 있는가 그리고 확장된 E-R 모델로부터 이러한 관계를 도출할 수 있는지에 대한 연구가 필요하다.

본 연구에서 적용한 방법론에 의해 생성되는 응용 시스템의 기능이 제한적이라는 문제는 있으나 E-R 모델만 설계되면 응용 시스템은 자동으로 생성되는 수준의 상당한 생산성 향상이 가능하였다.

본 연구에서는 실현 가능성을 검증하는 프로토타입의 제시에 주안점을 두었기 때문에 실용적 관점에서 응용시스템의 화면의 설계 등 보다 완성도 높은 응용시스템을 생성할 수 있는 자동화 체계를 제시하는 것은 다음 과제로 남겨두었다. 이용자의 특성과 요구에 부응하기 위한 인터페이스 설계의 범위는 훨씬 더 다양하고 복잡할 수 있을 것으로 예상된다. 본 연구에서 제안한 자동생성의 규칙과 원리를 실제의 소프트웨어로 구현하는 데에는 포함하여 이 밖에도 여러 가지 대안이 존재할 것이다. 또한 자동 생성된 업무 프로그램들이 작동할 플랫폼, 인터넷 웹 시스템과 기존의 클라이언트-서버 아키텍처, 미들웨어 등 다양한 차원의 문제들이 존재할 것이다. 이러한 문제들은 향후의 연구 대상들이다.

참 고 문 헌

[1] 권영호, "웹 어플리케이션의 사용자 인터페이스 테스트 자동화 기법", *한국정보처리학회논문지*, Vol. 10D, No. 002, 2003, pp. 293-300.

[2] 박병형, 양해술, "CBD 환경을 위한 4GL 어플리케이션의 웹 어플리케이션으로의 변화", *한국정보처리학회지*, Vol. 010, No. 003, 2003, pp. 89-96.

[3] 정연대, 임진수, "S/W 컴포넌트 조립도구 기반 조립 방법론 연구", *한국정보처리학회지*, Vol. 010, No. 003, 2003, pp.74-88.

[4] 정일주, *데이터베이스, 모델, 언어 및 설계*, 시그마프레스, 1999.

[5] 정일주, "시스템 구축 과정에서 소프트웨어 자동생성 도구의 사용, *정보기술응용연*", 제3권 제4호, 2002, pp. 63-92.

[6] 최성운, "객체지향 소프트웨어 개발 방법론의 객체지향 모델링", *한국정보처리학회논문지*, Vol. 08D, No. 004, 2001, pp. 401-408.

[7] Batra, Dinesh, Hoffer, and Jeffrey A., "An ER based methodology for modeling user views and detecting derived relationships", *Journal of Database Management*, Winter 1994, Vol. 5, No. 1, pp. 3-16.

[8] Chan, Hock Chuan, Poo, Danny C.C., and Woon, Cheng Peng. "An Object-Oriented Implementation of an Entity Relationship Model", *Journal of Systems & Software*, May 1998, Vol. 41, No. 2, pp. 117-125.

[9] Chen, Peter P. "The Entity-Relationship Model : Towards a Unified View of Data", *ACM Transaction on Database Systems*, Vol. 1, No. 1, March 1976, pp. 9-37.

[10] Chen, Peter P., Ilchoo Chung & Dennis Perry. *A Logical Database Design Framework*,

NBS Publication, NIST, 1982.

[11] Francett, Barbara, "Companies mind their business with integrated modeling tools", *Software Magazine*, Dec 1994, Vol. 14, No. 12, pp. 57-63

[12] McChesney, I., Glass, D., and Hughes, J. G., "CASE Tool Support for Requirements Capture and Analysis", *Microprocessing & Microprogramming*, Aug 1990, Vol. 30, No. 1-5, pp. 281-288.

[13] McFadden Fred R. and Jeffrey A. Hoffer, *Modern Database Management*, The Benjamin Cummings Publishing Company, 1994.

[14] Shelly, Gary B., Thomas J. Cashman, Judy Adamsky & Joseph J. Adamski. *Systems Analysis and Design* (Second Ed.), boyd & fraser publishing company, 1995.

[15] Siau, K.L., Chan, H.C., and Tan, K.P. "A CASE too for conceptual database design", *Information & Software Technology*, Dec 1992, Vol. 34, No. 12, pp. 779-786.

[16] Stoughton, Alan M., "ERwin streamlines GUI and adds versatile features", *InfoWorld*, July 1997, Vol. 19, No. 27, pp. 96-98.

[부 록]

1. 개체타입의 클래스 정의

```
Public Class EntityType
    Inherits SerializableData

  Public EntityTypeName As String
  Public EntityID As New Attribute()
  Public AttributeList As New Attribute_Collection()
  Public RelatedRelationships As New RelatedRelationshipType_Collection()
  Public Methods As New ArrayList()

  Public Overrides Function ToString() As String
      Return EntityTypeName
  End Function

  Public Function IsEmpty() As Boolean
      If EntityTypeName = "" Then
          Return True
      Else
          Return False
      End If
  End Function
```

```
Public Function LoadEntityType(ByVal EntityTypeName As String) As EntityType
    Dim DataFileName As String
    Dim myFilePath As New FilePath()
    Dim aEntityType As New EntityType()
    DataFileName = myFilePath.EntityTypeXMLFilePath(EntityTypeName)
    aEntityType = SerializableData.Load(DataFileName, GetType(EntityType))
    Return aEntityType
End Function

Public Function SaveEntityType(ByVal anEntityType As EntityType) As String
    Dim DataFileName As String
    Dim myFilePath As New FilePath()
    DataFileName = myFilePath.EntityTypeXMLFilePath(anEntityType.EntityTypeName)
    save(DataFileName)
    Return "OK"
End Function

Public Function Delete(ByVal anEntityType As EntityType) As String
    Dim myFilePath As New FilePath()
    Dim DataFileName As String = myFilePath.EntityTypeXMLFilePath _
        (anEntityType.EntityTypeName)
    File.Delete(DataFileName)
End Function

Public Function GetEntityTypeList() As EntityType_Collection
    Dim myArrayList As New ArrayList()
    Dim myFilePath As New FilePath()
    Dim folderName As String = myFilePath.EntityTypeFolderPath()
    Dim di As New DirectoryInfo(folderName)
    Dim fi As FileInfo() = di.GetFiles()
    Console.WriteLine("The following files exist in the current directory:")
    Dim fiTemp As FileInfo
    Dim anEntityType_Collection As New EntityType_Collection()
    For Each fiTemp In fi
        Dim anEntityType As New EntityType()
        Dim anEntityTypeName = fiTemp.Name.Remove(fiTemp.Name.Length - 4, 4)
        anEntityType = anEntityType.LoadEntityType(anEntityTypeName)
        anEntityType_Collection.Add(anEntityType)
```

```
        Next fiTemp
        Return anEntityType_Collection
    End Function
End Class

Public Class EntityType_Collection
            Inherits System.Collections.CollectionBase
    Default Public Property item(ByVal index As Integer) As EntityType
        Get
            Return Me.List.Item(index)
        End Get
        Set(ByVal Value As EntityType)
            Me.List.Item(index) = Value
        End Set
    End Property

    Public Sub Add(ByVal newEntityType As EntityType)
        Me.List.Add(newEntityType)
    End Sub

    Public Sub Remove(ByVal oldEntityType As EntityType)
        Me.List.Remove(oldEntityType)
    End Sub
End Class
```

2. 자동생성된 UI 프로그램 "Customer_Form"의 소스 코드

```
imports system.io
Public Class Customer_Form
    Inherits System.Windows.Forms.Form

#Region " Windows Form 디자이너에서 생성한 코드 "
    Public Sub New()
        MyBase.New()
        InitializeComponent()
    End Sub

    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
```

```
    If disposing Then
        If Not (components Is Nothing) Then
            components.Dispose()
        End If
    End If
    MyBase.Dispose(disposing)
End Sub

Private components As System.ComponentModel.IContainer
Friend WithEvents txtCustomer_ID As System.Windows.Forms.TextBox
Friend WithEvents lblCustomer_ID As System.Windows.Forms.Label
Friend WithEvents txtCustomerName As System.Windows.Forms.TextBox
Friend WithEvents lblCustomerName As System.Windows.Forms.Label
Friend WithEvents txtEmail As System.Windows.Forms.TextBox
Friend WithEvents lblEmail As System.Windows.Forms.Label
Friend WithEvents btnInquiry As System.Windows.Forms.Button
Friend WithEvents btnAdd As System.Windows.Forms.Button
Friend WithEvents btnModify As System.Windows.Forms.Button
Friend WithEvents btnDelete As System.Windows.Forms.Button

<System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
    Me.txtCustomer_ID = New System.Windows.Forms.TextBox()
    Me.lblCustomer_ID = New System.Windows.Forms.Label()
    Me.txtCustomerName = New System.Windows.Forms.TextBox()
    Me.lblCustomerName = New System.Windows.Forms.Label()
    Me.txtEmail = New System.Windows.Forms.TextBox()
    Me.lblEmail = New System.Windows.Forms.Label()

    'lblCustomer_ID
    Me.lblCustomer_ID.Location = New System.Drawing.Point(72, 80)
    Me.lblCustomer_ID.Name = "lblCustomer_ID"
    Me.lblCustomer_ID.Size = New System.Drawing.Size(100, 16)
    Me.lblCustomer_ID.TabIndex = 1
    Me.lblCustomer_ID.Text = "Customer_ID"

    'lblCustomerName
    Me.lblCustomerName.Location = New System.Drawing.Point(72, 110)
    Me.lblCustomerName.Name = "lblCustomerName"
```

```
Me.lblCustomerName.Size = New System.Drawing.Size(100, 16)
Me.lblCustomerName.TabIndex = 1
Me.lblCustomerName.Text = "CustomerName"

'lblEmail
Me.lblEmail.Location = New System.Drawing.Point(72, 140)
Me.lblEmail.Name = "lblEmail"
Me.lblEmail.Size = New System.Drawing.Size(100, 16)
Me.lblEmail.TabIndex = 1
Me.lblEmail.Text = "Email"

'txtCustomer_ID
Me.txtCustomer_ID.Location = New System.Drawing.Point(188, 80)
Me.txtCustomer_ID.Name = "txtCustomer_ID"
Me.txtCustomer_ID.Size = New System.Drawing.Size(128, 21)
Me.txtCustomer_ID.TabIndex = 0
Me.txtCustomer_ID.Text = ""

'txtCustomerName
Me.txtCustomerName.Location = New System.Drawing.Point(188, 110)
Me.txtCustomerName.Name = "txtCustomerName"
Me.txtCustomerName.Size = New System.Drawing.Size(128, 21)
Me.txtCustomerName.TabIndex = 0
Me.txtCustomerName.Text = ""

'txtEmail
Me.txtEmail.Location = New System.Drawing.Point(188, 140)
Me.txtEmail.Name = "txtEmail"
Me.txtEmail.Size = New System.Drawing.Size(128, 21)
Me.txtEmail.TabIndex = 0
Me.txtEmail.Text = ""

Me.btnInquiry = New System.Windows.Forms.Button()
Me.btnAdd = New System.Windows.Forms.Button()
Me.btnModify = New System.Windows.Forms.Button()
Me.btnDelete = New System.Windows.Forms.Button()

'btnInquiry
```

```
Me.btnInquiry.Location = New System.Drawing.Point(72, 300)
Me.btnInquiry.Name = "btnInquiry"
Me.btnInquiry.Size = New System.Drawing.Size(70, 32)
Me.btnInquiry.TabIndex = 1
Me.btnInquiry.Text = "Inquiry"

'btnAdd
Me.btnAdd.Location = New System.Drawing.Point(152, 300)
Me.btnAdd.Name = "btnAdd"
Me.btnAdd.Size = New System.Drawing.Size(70, 32)
Me.btnAdd.TabIndex = 1
Me.btnAdd.Text = "Add"

'btnModify
Me.btnModify.Location = New System.Drawing.Point(232, 300)
Me.btnModify.Name = "btnModify"
Me.btnModify.Size = New System.Drawing.Size(70, 32)
Me.btnModify.TabIndex = 1
Me.btnModify.Text = "Modify"

'btnDelete
Me.btnDelete.Location = New System.Drawing.Point(312, 300)
Me.btnDelete.Name = "btnDelete"
Me.btnDelete.Size = New System.Drawing.Size(70, 32)
Me.btnDelete.TabIndex = 1
Me.btnDelete.Text = "Delete"

Me.SuspendLayout()

'Customer_Form
Me.AutoScaleBaseSize = New System.Drawing.Size(6, 14)
Me.ClientSize = New System.Drawing.Size(432, 366)
Me.Controls.AddRange(New System.Windows.Forms.Control() {Me.txtCustomer_ID,
 Me.lblCustomer_ID, Me.txtCustomerName, Me.lblCustomerName, Me.txtEmail, Me.lblEmail
 Me.btnInquiry, Me.btnAdd, Me.btnModify, Me.btnDelete})
Me.Name = "Customer_Form"
Me.Text = "Customer_Form"
Me.ResumeLayout(False)
```

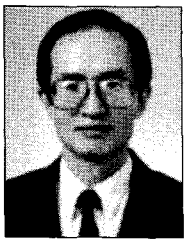
```
    End Sub
#End Region

    Private Sub btnInquiry_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Handles btnInquiry.Click
        Dim myCustomer As New Customer()
        myCustomer = myCustomer.LoadCustomer(txtCustomer_ID.Text)
        txtCustomer_ID.Text = myCustomer.Customer_ID
        txtCustomerName.Text = myCustomer.CustomerName
        txtEmail.Text = myCustomer.Email
        MessageBox.Show("Customer " & " " & myCustomer.Customer_ID & "조회 완료")
    End Sub

    Private Sub btnAdd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Handles btnAdd.Click
        Dim myCustomer As New Customer()
        myCustomer.Customer_ID = txtCustomer_ID.Text
        myCustomer.CustomerName = txtCustomerName.Text
        myCustomer.Email = txtEmail.Text
        myCustomer.SaveCustomer(myCustomer)
        MessageBox.Show("Customer " & " " & myCustomer.Customer_ID & "추가 완료")
    End Sub

    Private Sub btnModify_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Handles btnModify.Click
        Dim myCustomer As New Customer()
        myCustomer.Customer_ID = txtCustomer_ID.Text
        myCustomer.CustomerName = txtCustomerName.Text
        myCustomer.Email = txtEmail.Text
        myCustomer.SaveCustomer(myCustomer)
        MessageBox.Show("Customer " & " " & myCustomer.Customer_ID & "수정 완료")
    End Sub

    Private Sub btnDelete_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Handles btnDelete.Click
        Dim myCustomer As New Customer()
        Dim fileName as string
        fileName = "C:\연구\Data\Application\" & "Customer\" & txtCustomer_ID.Text & ".xml"
```

```
        File.Delete(fileName)
        MessageBox.Show("Customer " & " " & myCustomer.Customer_ID & "삭제 완료")
    End Sub
End Class
```

3. 자동 생성된 Customer 클래스 정의 소스 코드

```
Public Class Customer
    Inherits SerializableData
    Public Customer_ID As String
    Public CustomerName As String
    Public Email As String
    Public PhoneNumber As String
End Class
```

■ 저자소개



정 일 주

저자는 서울대학교 상과대학을 졸업하고 미국 Oregon 대학교에서 MBA 그리고 미국 캘리포니아 대학교(UCLA) 경영대학에서 컴퓨터 및 정보시스템 분야로 경영학 박사를 취득하였다. 이후 캐나다 University of British Columbia 대학의 경영대학과 미국 California State University 경영대학 경영정보학과 교수 그리고 한국전산원 연구위원을 거쳐 현재 홍익대학교 상경대학 경영정보학과 교수로 재직 중이다. 관심 분야는 정보시스템, 데이터베이스, 시스템 분석 및 설계, 웹 응용시스템 등이다.

◈ 이 논문은 2004년 12월 16일 접수하여 4차 수정을 거쳐 2005년 12월 6일 게재확정되었습니다.