XML DTD의 객체-관계형 데이터베이스 스키마로의 변환을 위한 Component 설계

이 정 수* · 주 경 수**

Designing Components for mapping from XML DTD to ORDB Schema

Jung-Soo Lee* · Kyung-Soo Joo**

Abstract

As the application area of XML extends to the database from simple contents, the technology for the storage and management of XML also grows with the XML. One of the main issues is how to efficiently manage the XML by using the previous DBMS. Many studies have been performed for the efficient management based on the XML application and database interface. However, the XML data is based on the object and has the hierarchical structure. Accordingly, a modeling plan needs to store the XML data on the variety type of database.

In this paper, for easier connection between XML application and database system, we have designed components for the transformation from XML DTD to ORDB schema. The method of the design for the transformation from XML DTD to ORDB schema is chosen the method of CBD.

※ 본 연구는 정보통신부의 ITRC 사업에 의해 수행된 것임.
* 순천향대학교 전산학과 석사과정
** 순천향대학교 정보기술공학부 교수

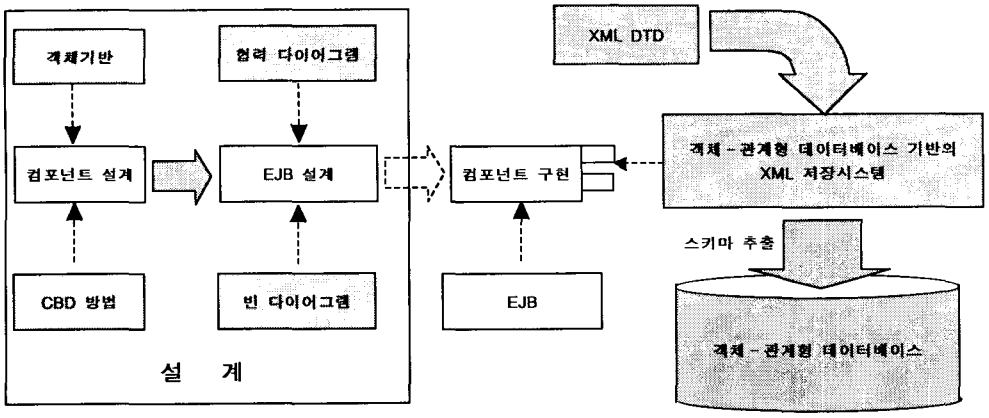
1. 서 론

XML(eXtensible Markup Language)은 웹에서 데이터를 교환하기 위한 보편적인 표준을 정의하고 있다. 또한 XML은 데이터가 모호하지 않게 쓰여지고 해석될 수 있도록 데이터에 대한 고유한 구조를 표현하기 위해 면밀한 텍스트-기반에 따른 방식을 사용하고 있다. XML의 간단한 태그 접근방식은 개발자들에게 HTML을 사용하던 때와 같이 간단하게 사용할 수 있도록 해주며, 또한 고유한 구조를 표현할 수 있도록 함으로써 고도로 구조화된 데이터베이스 레코드에서부터 비정형의 문서에 이르기까지 "디지털 자산" 전반을 처리할 수 있는 유연성과 확장성이 있는 메커니즘을 제공한다[21].

현재 많은 연구들은 XML DTD의 객체-관계형 데이터베이스 스키마로서 변환 방법들을 제시하였으나, 이를 컴포넌트로 설계하는 연구는 없었다. 본 논문에서는 객체 기반의 변환 방법을 제안하여 이를 CBD 방법론에 따라 컴포넌트를 설계하였다. 또한 CBD 방법론에 따라 설계된 컴포넌트를 EJB로 구현하기 위한 설계 방법까지 제시하였다. (그림 1)에서는 본 논문에 대한 구성을 표현한다.

다시 말하면, 본 논문에서는 XML 응용과 객체-관계형 데이터베이스 시스템간의 원활한 연계를 위해서, XML DTD를 객체-관계형 데이터베이스 스키마로의 변환 해 주는 컴포넌트를 설계한다. (그림 1)에서 제시한 설계는 두 부분으로 나누어 설계하였다. 첫 번째로, 컴포넌트 설계는 현재 소프트웨어 공학분야에서 각광을 받고 있는 CBD 방법론[20]에 설계하였고, XML DTD를 객체-관계형 데이터베이스 스키마로 변환하는 과정은 객체기반[17]에 의해 이루어졌다. 두 번째는 설계된 컴포넌트를 EJB기반으로 구현하기 위해 EJB 스펙에 따라 빈 다이어그램과 협력 다이어그램을 설계하였다[22]. 따라서, 본 논문에서 제시한 XML DTD의 객체-관계형 데이터베이스 스키마로의 변환을 위한 컴포넌트 설계는 EJB 컴포넌트 구축이 된다면 향후 XML 응용과 객체-관계형 데이터베이스간의 XML 저장시스템에 효율적으로 적용할 수 있다는 장점을 가지고 있다.

또한 이미 관계형 데이터베이스 기반으로 한 XML DTD를 관계형 데이터베이스 스키마로 변환을 위한 컴포넌트 설계 및 구현을 하였고, 이를 더 확장하기 위해 본 논문에서는 객체-관계형 데이터베이스 기반의 XML DTD를 객체-



(그림 1) 구 성

관계형 데이터베이스 스키마로 변환을 위한 컴포넌트를 설계하였다.

본 논문에서는 이 XML DTD에 정의한 엘리먼트와 어트리뷰트들을 객체화하여 객체-관계형 데이터베이스 스키마로 변환하기 위한 컴포넌트를 설계하였다. 따라서 제2절에서는 관련연구들을 기술하고, 제3절에서는 객체 기반 변환 방법[18, 19]에 대하여 기술하며, 제4절에서는 CBD 방법론에 의해 설계된 내용을 설명하며, 마지막으로 결론을 기술한다.

2. 관련 연구

XML이 단순한 컨텐츠에서 데이터베이스로까지 그 적용 분야가 확장되면서 XML로 표현된 정보들을 어떻게 효율적으로 저장하고 관리할 것인지에 대한 기술도 XML과 함께 발전되어 가고 있다. 가장 큰 이슈 둥의 하나는 기존의 DBMS로도 XML을 효율적으로 관리할 수 있는가이다.

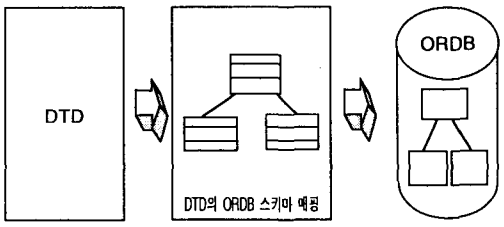
많은 논문들이 복잡한 XML 파일을 관계형 데이터베이스에 저장하는데 초점을 맞추어 데이터베이스와 XML 문서의 연결에 대하여 발표되었다[7, 10]. 발표된 논문들의 내용을 분류하면 주어진 XML 파일을 관계형 데이터베이스에 어떻게 저장할 것인가 하는 제안들을 하고 있으며, 주어진 XML 파일이나 DTD로부터 정보의 의미에 대하여 데이터베이스 제약사항들을 다루는 방법을 제안한다[4, 5, 6, 9].

또한 XML 문서가 객체-관계 시스템으로 구조화된 데이터 타입을 저장하는 방법이 제안되고 있다[3, 6]. 다른 한편으로는 객체-관계 데이터베이스로부터 XML로 변환하는 내용을 다루고 있으며, 그 내용은 어떻게 XML 파일을 객체-관계 데이터베이스에 저장하여 빠르게 질의할 것인가를 다루고 있다[2]. 또한 객체-관계 데이터베이스 시스템은 XML 문서들을 서로 연결하기 위한 것이며, XML의 중첩된 문서들을 객체-관계 모델로 변환하는 내용을 다루고 있다.

객체지향 데이터베이스 시스템의 풍부한 데이터 모델링 때문에 전자문서의 저장이 잘 이루어진다[1, 9]. 그러나 객체지향 데이터베이스 시스템은 효과적인 방법으로 대량의 데이터를 다루기에는 적합하지 않다[8].

3. 객체 기반 변환

XML DTD를 데이터베이스 스키마로 변환하고자 할 때 단순한 XML 문서는 직접 테이블로 변환이 가능하나 복잡한 XML 문서를 다룰 때는 객체 기반 변환 방법으로 처리되어야 한다. 따라서 테이블로 직접 변환하는 방법으로는 복잡한 XML 문서를 변환하기가 불가능하므로 이 단점을 개선시키기 위해 중간에 객체를 이용하여 (그림 2)와 같이 객체-관계형 데이터베이스 스키마로 변환한다[17-19].



(그림 2) 객체 기반 매핑

3.1 XML DTD의 객체 기반 매핑

3.1.1 엘리먼트

엘리먼트는 하나의 루트 엘리먼트만 가져야 하며, 다른 태그는 모두 엘리먼트 안에 확실하게 중첩되어야 한다. 엘리먼트 타입은 두 개의 타입으로 분류하는데 PCDATA만 가진 엘리먼트의 타입을 단순 엘리먼트 타입이라고 하며,

PCDATA을 제외한 모든 엘리먼트의 타입을 복합 엘리먼트 타입이라고 한다. 즉, 복합 엘리먼트 타입은 엘리먼트의 자식 엘리먼트, 혼합 엘리먼트 그리고 엘리먼트의 어트리뷰트들이다. 단순 엘리먼트 타입 경우 루트엘리먼트 이름은 클래스 이름으로 변환하고, PCDATA를 가진 엘리먼트들은 클래스 속성으로 변환한다. 복합 엘리먼트 타입일 경우에는 별도의 자식 클래스로 분리할 수 있다.

루트 엘리먼트는 부모 클래스로 변환하고, 루트 엘리먼트 안에 있는 엘리먼트가 또 다른 자식 엘리먼트들을 가질 경우 서브 엘리먼트라고 말한다. 서브 엘리먼트는 별도의 자식 클래스로 변환하고, 부모 클래스에서 자식 클래스를 참조형으로 연결해 준다.

3.1.2 어트리뷰트

모든 엘리먼트는 어트리뷰트를 가질 수 있다. 어트리뷰트는 이름 또는 값의 형태를 가진다. 어트리뷰트는 엘리먼트에 포함되며, 어트리뷰트에 부여된 값을 통해서 그 엘리먼트에 어떠한 특징을 제공하게 된다.

어트리뷰트는 XML 파서가 애플리케이션에게 보내는 값들을 뜻하는데, 엘리먼트의 내용과는 구별되는 값으로, 단일 값을 가진 어트리뷰트와 다중 값을 가진 어트리뷰트으로 분류된다. 단일 값을 가진 어트리뷰트는 문자열 어트리뷰트(CDATA), 토큰 어트리뷰트(ID, IDREF, NMTOKEN, ENTITY, NOTATION), 열거형 어트리뷰트이며 객체로 변환될 때 클래스의 속성으로 변환된다.

3.1.3 복합 내용 모델 변환

복합 내용 모델이란 엘리먼트들간의 순차적으로 이루어진 엘리먼트, 엘리먼트 선택, 반복되는 엘리먼트, 서브그룹일 경우를 말한다. 순차일 경우 DTD에 있는 엘리먼트와 엘리먼트에 속하는 자식 엘리먼트들을 순차적으로 객체로 변환하면, 자식 클래스와 부모 클래스간의 연결을 참조형으로 변환할 수 있다. 두 개 이상의 엘리먼트가 선택일 경우 DTD 엘리먼트 안에 자식 엘리먼트들이 있는데 그 중에 하나만 선택할 경우에도 객체로 변환될 수 있고, 변환된 객체는 다시 객체-관계형 데이터베이스 스키마로 변환된다. 엘리먼트 안에 자식 엘리먼트가 중복된 경우 객체화로 변환할 때 엘리먼트 안에 중복된 같은 엘리먼트들이 클래스 속성으로 변환하는 경우 배열 형태로 변환된다. 이렇게 변환된 클래스 속성은 별도의 테이블로 변환된다. 하나 이상의 엘리먼트들이 존재한다는 의미로 클래스 속성의 데이터형을 String[]으로 변환한다. 즉 작성자가 알 수 없는 만큼 엘리먼트가 존재한다는 의미로 별도의 객체-관계형 데이터베이스 스키마로 변환된다. 서브그룹이란 <!ELEMENT A (B, (C | D))>와 같은 경우를 말하며, 이 엘리먼트는 <!ELEMENT A (B, C)>와 <!ELEMENT A (B, D)> 엘리먼트로 분류된다. 따라서 두 개의 객체로 변환할 수 있다.

3.1.4 혼합 내용 모델 변환

텍스트나 엘리먼트, 그 둘을 모두 포함할 수 있는 엘리먼트들을 혼합 내용 모델이라고 하며 XML 프로세서는 공백, 탭 등의 PCDATA와 엘리먼트 내용간의 구분이 어렵다. 그 이유는 끝 태그와 다음의 시작 태그 사이에 공백이 있으면 불분명하게 된다. 혼합 내용 모델에서 선택이 가능한 그룹은 하나이고, #PCDATA로 시작하며, 그 뒤에는 혼합 내용의 연산자수를 나타내는 타입 순서로 되며, 각각은 한 번만 선언된다. #PCDATA만 유일한 옵션이며, "*"는 반드시 괄호를 닫은 바로 뒤에 와야 한다.

3.2 객체로부터 객체-관계형 데이터베이스 스키마로 변환

3.2.1 변환

XML DTD의 엘리먼트들을 객체를 이용하여 객체 - 관계형 데이터베이스 스키마로 변환한다. 먼저 XML DTD의 엘리먼트들을 객체클래스로 변환하고, 이들은 객체-관계형 데이터베이스 스키마 객체, 즉 객체 타입(Object Type), 객체 테이블(Object Table), 참조 대 포함(Referenced vs Embedded), 컬렉션(Collections)으로 변환된다.

3.2.2 변환 과정

다음은 XML DTD로부터 변환된 객체클래스가 객체-관계형 데이터베이스 스키마로 변환하는 과정이다.

(1) 타입, 테이블, 보조적인 객체를 생성하기 위해서는 객체-관계형 데이터베이스 시스템으로 확장되어 사용한다.
(2) 객체 클래스는 구조화된 객체 타입을 생성하며, 테이블과 연결하여 타입을 하나 또는 여러 개의 테이블에 객체의 원소가 되도록 한다.
(3) 객체 타입은 사용자 정의 타입이거나, 속성을 가진 객체타입이다.
(4) 클래스에서의 객체 속성은 객체 타입에서의 속성이다.
(5) 타입이 테이블이나 객체 또는 특정 타입으로 변환할 때 클래스에서의 객체 속성 타입은 객체 타입에서의 속성 타입이다.
(6) 객체 속성이 NULL 제약조건을 가지면 NOT NULL 제약조건도 가진다.
(7) 객체 속성이 초기값을 가지면 컬럼에 DEFAULT 값을 더한다.
(8) 독립적인 클래스나 함축적인 성질을 가진 클래들을 위해 기본키로 정수를 생성한다. {oid}를 위해 태그된 컬럼에 기본키 제약조건에 따라 {oid}를 더한다.
(9) 부 클래스를 위해 기본키 제약조건과 외래키 제약조건에 따라 각각의 부모 클래스의 키를 더한다.
(10) 클래스들의 결합을 위해 객체 타입을 생성하고 기본키 제약조건과 외래키 제약조건에 따른 역할 테이블로부터 기본키를 추가한다.
(11) 교환 oid가 <n>인 경우 UNIQUE 제약조건에 따른 컬럼을 추가한다.
(12) 각각의 확실한 제약조건을 위해 CHECK를 실시한다.
(13) 결합하는데 있어서 각각의 0..1, 1..1 역할을 위한 참조 테이블에서 외래키 컬럼을 생성한다. 교대로 단일 객체들이나 컬럭션을 위한 그 자신 안에서 속성으로 객체를 선언하기 위하여 객체 타입에 참조를 사용하거나 또는 다중관계 객체를 위한 참조의 배열로 객체를 선언하기 위하여 객체 타입에 참조를 사용한다.
(14) 집합 테이블에 외래키를 가진 다중 집합을 위해 기본키를 생성하고, 기본키를 위한 컬럼을 추가하며, 테이블 안에서 그 집합을 저장하기 위한 객체 타입을 사용한다. 또한 단일 객체를 위한 객체 속성 또는 컬렉션을 통하여 배열이나 네스티드 테이블을 사용한다.
(15) 너무 많이 이동하게 되는 이진 결합 클래스들을 최적화 한다.
(16) 외래키를 사용하여 결합이 안된 클래스와 함께 다대다 결합의 관계 테이블을 생성한다.
(17) 다대다 결합에서 역할 테이블의 키로부터 기본키, 외래키의 제약조건을 생성한다.

(18) 객체 클래스의 전달 작용을 위한 객체 타입에 메소드를 생성한다.

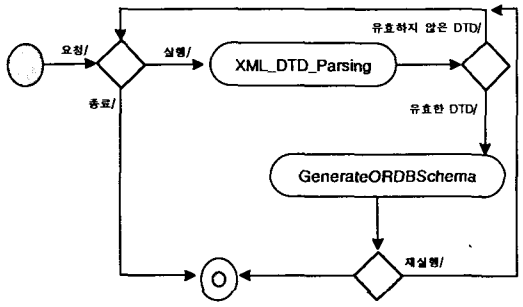
4. 컴포넌트 설계

본 논문에서는 UML 모델링 기법을 이용하여 소프트웨어 개발방법론인 CBD 방법론에 따라 컴포넌트를 설계하였다.

4.1 요구사항

4.1.1 비즈니스 프로세스

(그림 3)에서 비즈니스 프로세스는 비즈니스 모델링하기 위해 전체적인 흐름을 보여준다.

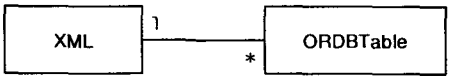


(그림 3) 비즈니스 프로세스

4.1.2 비즈니스 개념 모델

비즈니스 프로세스를 기술하는 동안 명확히 정의해야 할 필요가 있는 업무 용어를 산출하여야 한다. 산출된 업무 용어들은 전체적으로 도식화되어야 하는데, 이것을 비즈니스 개념 모델이라고 한다. 따라서 비즈니스 개념 모델은 실제 업무를 지원하기 위해 시스템이 관리하는 정보들을 표현한다.

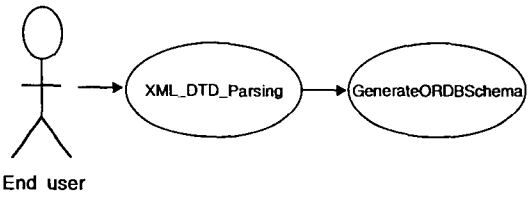
(그림 4)에서는 하나의 XML를 여러 개의 객체-관계형 데이터베이스 테이블로 변환할 수 있다. XML DTD를 여러 개의 객체로 분리하여, 분리된 객체마다 각각 하나의 객체-관계형 데이터베이스 테이블에 표현할 수 있다.



(그림 4) 비즈니스 개념 모델 다이어그램

4.1.3 유즈케이스

유즈케이스는 시스템의 동적인 면을 모델링하기 위한 것이며, 비즈니스 프로세스 모델에서 프로세스 단계별로 책임을 할당하는 초기 작업을 진행한다. 유즈케이스 모델은 시스템과 상호 작용하는 액터(actor)를 포함하며, 액터는 또 다른 시스템을 나타내기도 한다. 또한 유즈케이스 모델 내의 액터는 하나 이상의 유즈케이스에 관여할 수 있다. (그림 5)에서 보여주고 있는 유즈케이스 다이어그램은 액터가 End user이며, XML_DTD_Parsing 유즈케이스 기능이 XML DTD를 파싱하여 유효한지를 결정해주는 것을 보여 준다. GenerateORDBSchema는 파싱한 결과가 유효한 XML DTD일 경우, 그 XML DTD를 객체-관계형 스키마로 변환해주는 역할을 한다.



(그림 5) 유즈케이스 다이어그램

4.1.4 유즈케이스 명세서

유즈케이스 명세서는 유즈케이스 다이어그램에서 유즈케이스, 액터, 그리고 그들 사이의 연관관계를 표시한다. 그 다음에 유즈케이스의 사건 흐름을 기술함으로써 유즈케이스 명세화하였다. 사건 흐름은 액터의 요청에 대해서 액터

와 유즈케이스 사이에 어떤 상호작용이 발생하는지 보여준다.

4.2 컴포넌트 식별

컴포넌트 식별 단계는 요구사항 정의에서 나타나는 워크플로우의 산출물인 비즈니스 개념 모델과 유즈케이스 모델을 입력받아 작업을 진행한다. 컴포넌트 식별 단계의 목표는 초기 인터페이스와 컴포넌트 명세를 생성하여 컴포넌트 아키텍처의 초안을 만드는 것이다.

4.2.1 시스템 인터페이스 식별

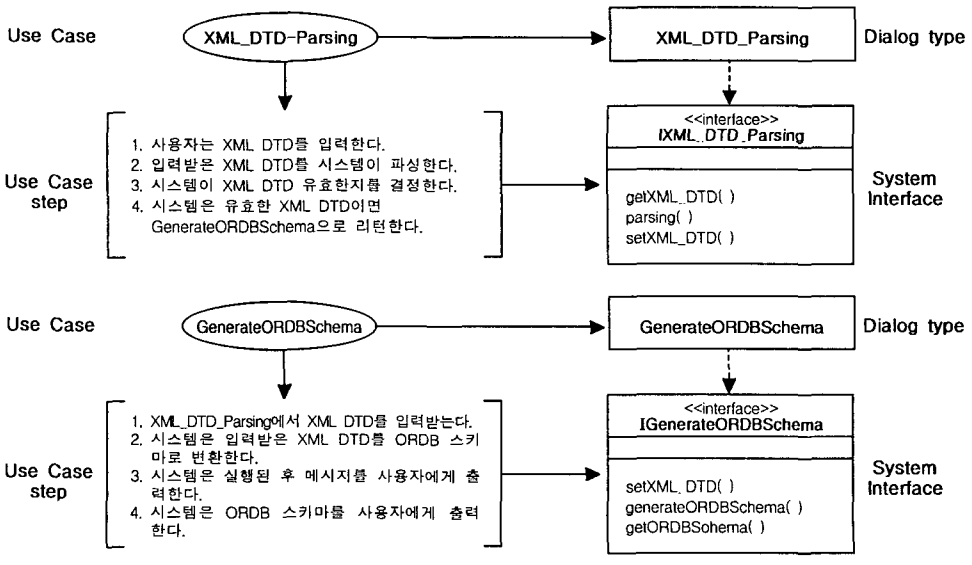
각각의 유즈케이스들은 하나의 인터페이스와 하나의 다이얼로그 타입을 정의한다. 그리고 각각의 유즈케이스 스텝들을 하나씩 살펴보고, 각각의 단계들을 만족시키기 위해 어떤 기능을 제공해야할 책임이 있는가를 판단하여 인터페이스를 찾는다. (그림 7)은 시스템 인터페이스를 추출하는 과정을 보여준다. (그림 6)에 있는 유즈케이스 명세서에서 보여주는 각각의 단계에

이 름 : XML_DTD_Parsing
구동자 : 사용자
목 표 : XML DTD를 파싱한다.
주요시나리오
1. 사용자는 XML DTD를 입력한다.
2. 입력받은 XML DTD를 시스템이 파싱한다.
3. 시스템이 XML DTD 유효한지를 결정한다.
4. 시스템은 유효한 XML DTD이면 GenerateORDBSchema으로 리턴한다.

확 장
1.1 XML DTD가 아닌 다른 문서일 경우 사용자에게 에러 메시지를 출력한다.
3.1 파싱된 XML DTD가 유효하지 않은 경우 사용자에게 에러 메시지를 출력한다.

이 름 : GenerateORDBSchema
구동자 : XML_DTD_Parsing
목 표 : XML_DTD_Parsing에서 입력받은 XML DTD를 ORDB 스키마로 변환한다.
주요시나리오
1. XML_DTD_Parsing에서 XML DTD를 입력받는다.
2. 시스템은 입력받은 XML DTD를 ORDB 스키마로 변환한다.
3. 시스템은 실행된 후 메시지를 사용자에게 출력한다.
4. 시스템은 ORDB 스키마를 사용자에게 출력한다.

(그림 6) 유즈케이스 명세서

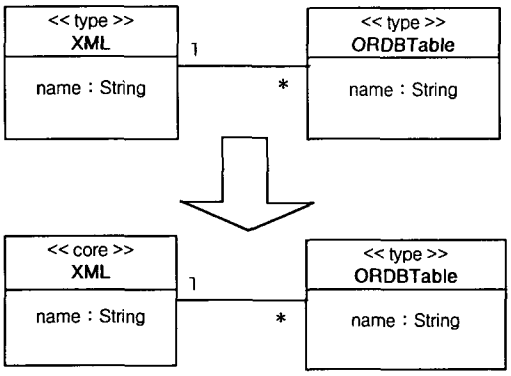


(그림 7) 시스템 인터페이스와 유즈케이스의 매핑

따라 인터페이스를 추출하는 것을 보여준다.

4.2.2 비즈니스 인터페이스 식별

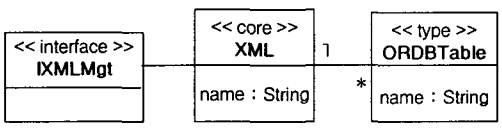
시스템에 의해서 관리되어야 하는 정보들이 추상화시켜 보여준다. 비즈니스 인터페이스를 식별하기 위해서는 비즈니스 개념 모델에서 비즈니스 타입 모델로 생성되고, 타입들 중에서 독립적으로 관리될 수 있는 모델을 핵심(core) 비즈니스 타입으로 지정해 준다.



(그림 8) 핵심 비즈니스 타입 모델 다이어그램

4.2.3 비즈니스 인터페이스 생성과 책임 할당

어느 정보가 어느 인터페이스에 의해서 관리되어야 하느냐 하는 것을 명세화 한다. 또한 인터페이스간의 의존성이 있는지를 명세화 하는데, 이를 위해서 상세 타입의 소유권을 적절한 인터페이스에 할당한다. 비즈니스 인터페이스 생성은 핵심(core) 비즈니스 타입마다 하나의 인터페이스가 생성될 수 있다. (그림 9)에서 보여주는 것과 같이 XML이 핵심 비즈니스 타입이므로 하나의 인터페이스를 생성하고 있다.



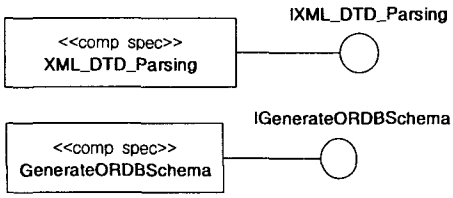
(그림 9) 인터페이스 책임 다이어그램

4.3 컴포넌트 명세 아키텍쳐

초기 컴포넌트 명세를 생성하고 그것들이 어떻게 서로 어울리게 원리에 대한 것을 명세화 할 수 있다. 컴포넌트는 배포의 단위가 되며, 컴포넌트 시스템의 교체의 단위이다. 또한 우리가 만들거나 구매해야하는 공급의 단위, 즉 실체화(realization)의 단위이기도 하다. 따라서 애플리케이션 개발자들은 선택적으로 컴포넌트를 교체하여 시스템을 개발할 수 있다.

4.3.1 시스템 컴포넌트 명세

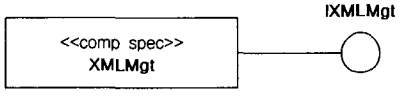
유즈케이스를 통해 도출된 시스템 인터페이스들은 중첩적이며 동일한 생명주기를 가지는 업무상의 사상들을 관리하고 있다. 이러한 인터페이스는 시스템 인터페이스들을 제공하는 하나의 컴포넌트 명세를 제안할 수 있다.



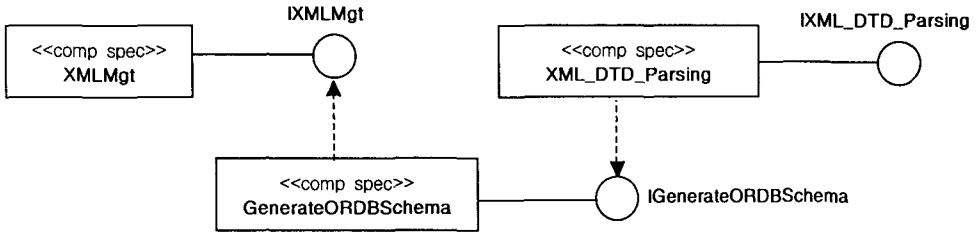
(그림 10) 시스템 컴포넌트 다이어그램

4.3.2 비즈니스 컴포넌트 명세

식별된 비즈니스 인터페이스들은 각각 한 개의 컴포넌트 명세를 작성할 수 있다. 관리자 인터페이스가 핵심 비즈니스 타입과 그 타입에 연관된 상세 타입의 인스턴트들을 관리하게 되므로 이들 인터페이스들은 독립적으로 관리되어지는 정보와 관련이 있다.



(그림 11) 비즈니스 컴포넌트 다이어그램



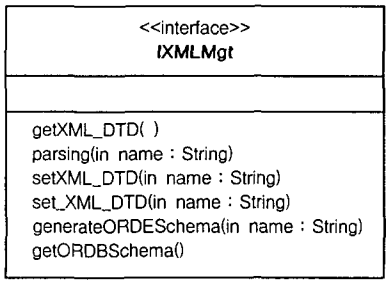
(그림 12) 컴포넌트 명세 아키텍쳐

4.3.3 컴포넌트 명세

(그림 12)에서는 상호 작용하는 컴포넌트간에 전달되는 메시지들을 통해 그 연관성을 분석 할 수 있다. 따라서 컴포넌트 명세 아키텍쳐는 시스템의 정적인 부분을 표현하면서도 내부적으로 동적인 부분을 표현한다. (그림 12)을 통해 시스템의 전체적인 구성을 살펴볼 수 있다.

4.4 컴포넌트 상호작용

컴포넌트들을 식별하기 위해 초기 인터페이스와 컴포넌트들이 제공된다. 그리고 컴포넌트들이 요구사항에 기술된 기능들을 충족시키기 위해서 어떻게 함께 동작 할 것인가를 결정한다. 이러한 결정은 시스템 내부적으로 요구되는 여러 가지 다양한 상호작용을 정의하기 위한 것이며, 기존의 인터페이스가 사용되는 방법을 찾아내기 위한 것이고, 또한 새로운 인터페이스와 오퍼레이션들을 발견하기 위한 것이다



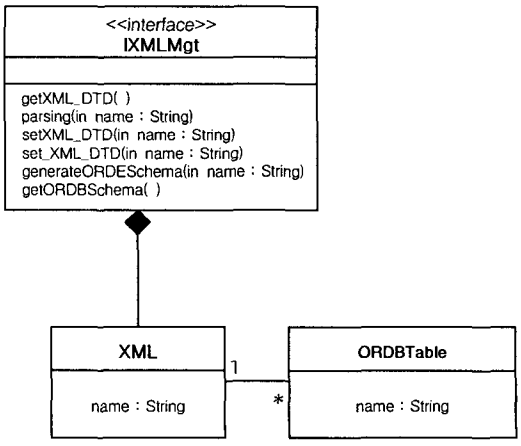
(그림 13) 비즈니스 인터페이스 식별

4.5 컴포넌트 명세서

컴포넌트 시스템에는 사용 계약과 실체화 계약이 있다. 사용 계약은 인터페이스 명세를 정의하고 실체화 계약은 컴포넌트 명세를 정의한다. 컴포넌트 명세는 기본적으로 인터페이스들을 묶어 놓은 것이기 때문에 인터페이스 명세는 이 두 가지 계약의 핵심이라고 할 수 있다.

4.5.1 인터페이스 명세서

인터페이스는 오퍼레이션들의 집합이며, 각 오퍼레이션에는 클라이언트를 위해서 컴포넌트 객체가 수행하게 될 어떤 서비스나 기능이 정의되어 있다. 그러므로 오퍼레이션은 클라이언트와 컴포넌트 객체들 사이에 작은 단위의 계약을 의미한다.



(그림 14) 인터페이스 명세 다이어그램

5.2.2 컴포넌트 명세화

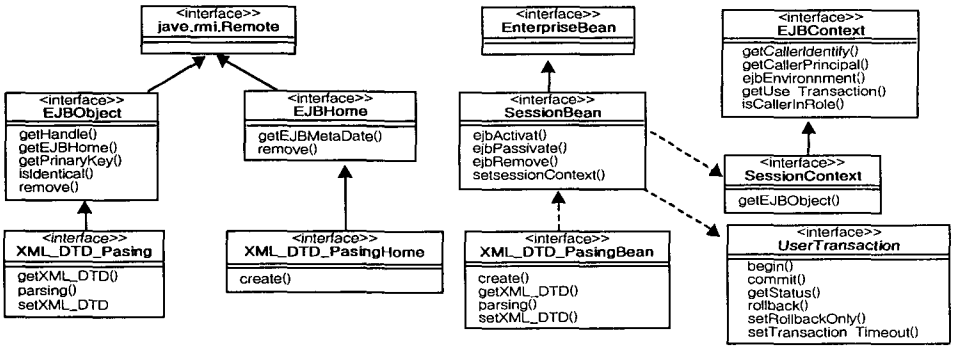
컴포넌트 구현자와 조립자가 알아야 할 추가적인 명세 정보, 특히 컴포넌트와 인터페이스 사이의 의존성에 대해서 고려하게 된다. 이러한 정보들이 컴포넌트 명세를 형성하게 되며, 실체화에 적용되는 제약조건들이 명시될 필요가 있는 경우에 해당 컴포넌트 명세를 정의할 때 정의된다.

4.6 빈 다이어그램

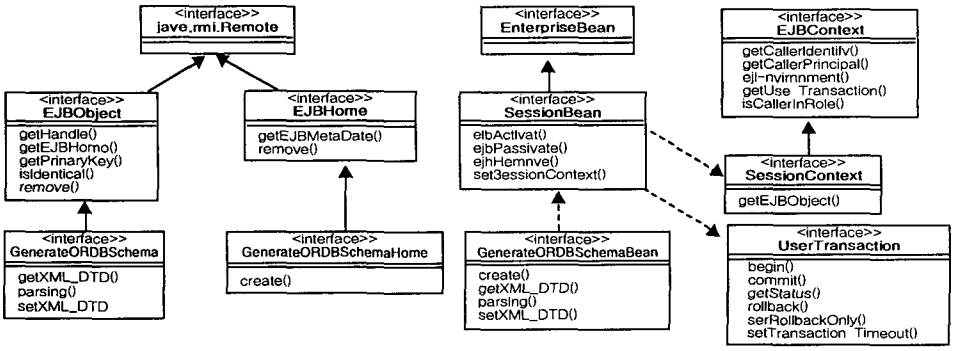
선(SUN)사에 의해 제안된 EJB(Enterprise JavaBeans) 아키텍처는 컴포넌트 기반의 분산 업무 애플리케이션의 개발과 배치를 위한 컴포넌트 아키텍처이다. EJB를 사용함으로써, 복잡한 분산 객체 프레임워크에 대한 작성 없이 확장성과 신뢰성이 높고, 안전한 애플리케이션의 작성이 가능하게 된다[11].

EJB는 분산 트랜잭션 지향 엔터프라이즈 애플리케이션의 컴포넌트로서 한 클래스가 하나의 빈이 된다. 모든 EJB 생성과 관련된 기능을 가지고 있는 홈(Home)과 비즈니스 로직을 가지고 있는 원격(Remote) 인터페이스들을 가지며, 클래스가 지원하는 기능과 상태, 자료의 특징에 따라 세션빈(Session Bean)과 엔티티 빈(Entity Bean)으로 나뉘게 된다.

(그림 15)에서 실전 화살표는 상속(extends)을 의미하고 점선 화살표는 인터페이스의 구현(implement)을 의미한다. XML_DTD_Parsing 빈 클래스 역할은 사용자가 XML DTD를 입력하며 빈 클래스에서는 XML DTD를 파싱하여 XML DTD가 유효한지를 결정한다.



(그림 15) XML_DTD_Parsing 빈 다이어그램



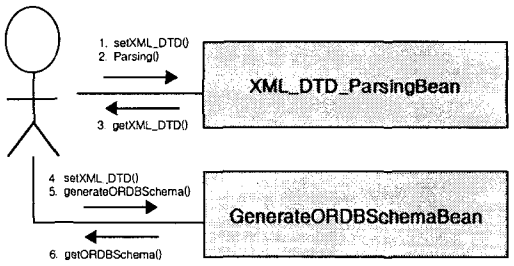
(그림 16) GenerateORDBSchema 빈 다이어그램

유효한 XML DTD이면, (그림 16)에 정의한 GenerateORDBSchema 빈 클래스로 리턴한다.

(그림 16)에서 실전 화살표는 상속(extends)을 의미하고 점선 화살표는 인터페이스의 구현(implement)을 의미한다. GenerateORDBSchema 빈 클래스는 파싱된 XML DTD를 입력받아 객체-관계형 데이터베이스 스키마를 추출한다.

4.7 컴포넌트 협력 다이어그램

사용자는 시스템의 동적인 뷰(View)를 설명하기 위하여 협력 다이어그램을 사용한다. 협력 다이어그램은 객체와 객체(또는 컴포넌트와 컴포넌트)들 사이의 메시지를 주고받는 객체(또는 컴포넌트)의 구조적인 구성을 강조하는 다이어그램이다.



(그림 17) 컴포넌트 협력 다이어그램

(그림 18)에서는 두 개의 컴포넌트간의 메시지를 주고받는 관계가 다음과 같다.

(1) setXML_DTD() : 사용자가 XML DTD를 입력한다.

(2) parsing() : 입력된 XML DTD를 유효한 DTD인지를 결정한다.

(3) getXML_DTD() : 유효한 DTD이면 사용자에게 이 XML DTD를 리턴한다.

(4) setXML_DTD() : 이 인터페이스는 (1)과는 무관한 인터페이스이므로 사용자는 유효한 XML DTD를 입력해 준다.

(5) generateORDBSchema() : 본격적으로 유효한 XML DTD를 객체-관계형 데이터베이스 스키마로 변환해 주는 인터페이스이다.

(6) getORDBSchema() : 이 인터페이스는 최종적으로 사용자가 변환된 객체-관계형 데이터베이스 스키마를 사용자에게 출력해 준다.

5. 결 론

현재 많은 연구들은 XML DTD의 객체-관계형 데이터베이스 스키마로서 변환 방법들을 제시하였으나, 이를 컴포넌트로 설계하는 연구는 없었다. 본 논문에서는 객체-기반의 변환 방법을 제안하여 이를 CBD 방법론에 따라 컴포넌트를 설계하였다. 또한 CBD 방법론에 따라 설계된 컴포넌트를 EJB로 구현하기 위한 설계 방법까지 제시하였다. 따라서 본 논문에서 제시한 EJB 컴포넌트는 향후 XML 응용과 객체-관계형 데이터베이스간의 XML 저장시스템 구축 시에 효율적으로 적용할 수 있다는 장점을 가지고 있다.

참 고 문 헌

[1] Boem, K., K. Aberer, : HyperStorM-Administering Structured Documents Using Object-Oriented Database Technology. Proc. of the ACM SIGMOD Int. Conf. on Management of Data, Montreal, Canada, June 1996.

[2] Carey, M., D. Florescu, Z. Ives, Y. Lu, J. Shanmugasundaram, E. Shekita, S. Subramanian, : XPERANTO : Publishing Object Relational Data as XML. In Suciu, D., Vossen(eds.), G., Proceedings of the Third International Workshop on the Web and Databases, WebDB 2000, Dallas, Texas, USA, May 18-19, 2000, 105-110.

[3] Cheng, J., Xu, J., : XML and DB2, In Sixteenth International Conference on Data Engineering(ICDE'00), 28 February-3 March 2000, San Diego, IEEE Computer Society Press, Los Alamitos, CA, 2000, pp.569-576.

[4] Florescu, D., D. Kossmann, : Storing and Querying XML Data using an RDBMS. Data Engineering 22 : 3 (1999), pp.27-34.

[5] Kanne, C.-C., Moerkotte, G. : Efficient St orage of XML Data. In Sixteenth International Conference on Data Engineering(ICDE'00), 28 February-3 March 2000, San Diego, IEEE Computer Society Press, Los Alamitos, CA, 2000, 198.

[6] Klettke, M., Meyer, H., : XML and Object-Relational Database Systems-Enhancing Structural Mappings Based on Statistics. In Suciu, D., Vossen(eds.), G., Proceedings of the Third International Workshop on the Web and Databases, WebDB 2000, Dallas, Texas, USA, May 18-19, 2000, pp.63-68.

[7] James Rumbaugh, Object-oriented modeling and design, Englewood Cliffs, N.J, Prentice Hall, 1991.

[8] Joo Kyung-soo, "A Design of Middleware Components for the Connection between XML and RDB," 2001 IEEE International Symposium on Industrial Electronics Proceedings, Pusan, Korea, June, 2001.

[9] Lee, D., W. W. Chu, : Constraints-Preserving Transformation from XML DTD to Relational Schema. In Laender, A., Liddle, S., Storey(eds.), V., Conceptual Modeling, Salt Lake City, Utah, USA, October, 9-12, 2000, LNCS 1920, Springer-Verlag, Berlin, 2000, pp.323-338.

[10] Michel Goosens, Janne Saarela, A Practical Introduction to SGML, Vol.16(3), In TUG boat, 1995.

[11] 김수동, "Enterprise JavaBeans(EJB) 기반의 컴포넌트 프로그래밍", 정보처리 제7권, 제4호, 2000, pp.40-45.

[12] 배두환, "e-Business를 위한 컴포넌트 소프트웨어 개발", 정보처리 제7권, 제4호, 2000, pp.27-32.

[13] 서순모, 양해술, "동적 컴퓨터 환경에서의 전자상거래 컴포넌트의 도입 방안", 한국정보처리학회 소프트웨어공학연구회지, 제3권, 제4호, 2000, pp.82-90.

[14] 온용훈, 강병도, 정영은, "소프트웨어 구조 설계 환경 HappyWork 개발", 한국정보처리학회 소프트웨어공학연구회지, 제3권, 제4호, 2000, pp.45-55.

[15] 이상태, 주경수, "제약조건 유지를 위한 XML DTD의 관계 스키마로 변환 방법", 한국인터넷정보학회, 제1권, 제2호, 2000, pp.189-196.

[16] 이상태, 주경수, "UML 객체모델의 ORDB 객체 매핑", 한국인터넷정보학회, 제2권, 제1호, 2001, pp.187-191.

[17] 이상태, 주경수, "객체모델을 이용한 XML

DTD의 ORDB 스키마로의 변환", 한국데이터베이스학회, 춘계 Conference, 2001, pp. 303-310.

[18] 이상태, 이정수, 주경수, "객체모델을 기반으로 한 XML DTD의 RDB 스키마로의 변환 방법", 대한전자공학회, 하계종합논문대회, 제24권, 제1호, pp.113-116.

[19] 이상태, 이정수, 주경수, "XML DTD를 기반으로 한, RDB 스키마 설계를 위한 Component 구현", 한국정보처리학회 지식 및 데이터공학연구회 제8회 학술발표대회 논문집, 2001, pp.309-316.

[20] 김경주, 조남규, UML Components 컴포넌트 기반 소프트웨어 명세를 위한 실용적인 프로세스, 인터비전, 2001, pp.75-77.

[21] 김명주, XML and Java, 이한출판사, 2000, pp.16-24.

[22] 유재우, 최재영, 최종명, 박준서, 프로그래머를 위한 EJB, FREELEC, 2001.

저자소개

이 정 수

청운대학교를 졸업하고, 현재 순천향대학교 대학원 전산학과 석사과정을 재학중입니다. 주요 관심분야는 XML, 분산 데이터베이스, 전자상거래, B2B 등이다.

주 경 수

고려대학교를 졸업하였으며, 동대학원에서 이학석사, 이학박사 학위를 취득하였고, 현재 순천향대학교 정보기술공학부 정교수로 재직하고 있으며, 주요 관심분야는 Database System, Semi-structured Data and XML, System Integration, Object-oriented System 등이다.