DOI: 10.3745/KIPSTD.2010.17D.2.129

XML-GDM을 기반으로 한 UML 클래스 다이어그램으로 사상을 위한 XML문서와 질의의 객체 모델링

박 대 현[†] · 김 용 성[††]

요 약

최근 다양한 분야에서 폭넓게 활용되고 있는 XML 문서는 유연하고도 개방적인 특성으로 인해 정보교환이나 전송을 위한 수단으로 널리 이용되고 있다. 한편 XML 문서를 위한 시각적, 직관적 질의 언어인 XML-GL은 질의에 대한 의미와 결과 문서의 구조를 시각적으로 표현할 수 있기 때문에 XML 문서에 대한 구조 검색과 정보의 공유가 용이하다. 그리고 UML은 정해진 표기법과 다양한 다이어그램을 이용하여 객체지향 분석과 설계를 위한 도구로 사용되고 있다. 따라서 본 논문은 XML-GL의 데이터 모델인 XML-GDM을 기반으로 표현된 XML 문서를 UML 클래스 다이어그램으로 사상하기 위한 새로운 객체 모델링 방안을 제안한다. 이를 통해서 XML 문서를 직관적인 방법으로 객체지향데이터로 변환하고 저장/관리할 수 있다. 또한 객체지향 검색방법을 적용하면 보다 효율적으로 XML 문서를 검색할 수가 있다.

키워드 : XML, XML-GL, XML-GDM, UML, 객체 모델링

Object Modeling for Mapping from XML Document and Query to UML Class Diagram based on XML-GDM

Dae-Hyun Park[†] · Yong-Sung Kim[††]

ABSTRACT

Nowadays, XML has been favored by many companies internally and externally as a means of sharing and distributing data. there are many researches and systems for modeling and storing XML documents by an object-oriented method as for the method of saving and managing web-based multimedia document more easily. The representative tool for the object-oriented modeling of XML documents is UML (Unified Modeling Language). UML at the beginning was used as the integrated methodology for software development, but now it is used more frequently as the modeling language of various objects. Currently, UML supports various diagrams for object-oriented analysis and design like class diagram and is widely used as a tool of creating various database schema and object-oriented codes from them. This paper proposes an Efficinet Query Modelling of XML-GL using the UML class diagram and OCL for searching XML document which its application scope is widely extended due to the increased use of WWW and its flexible and open nature. In order to accomplish this, we propose the modeling rules and algorithm that map XML-GL. which has the modeling function for XML document and DTD and the graphical query function about that. In order to describe precisely about the constraint of model component, it is defined by OCL (Object Constraint Language). By using proposed technique creates a query for the XML document of holding various properties of object-oriented model by modeling the XML-GL query from XML document, XML DTD, and XML query while using the class diagram of UML. By converting, saving and managing XML document visually into the object-oriented graphic data model, user can prepare the base that can express the search and query on XML document intuitively and visually. As compared to existing XML-based query languages, it has various object-oriented characteristics and uses the UML notation that is widely used as object modeling tool. Hence, user can construct graphical and intuitive queries on XML-based web document without learning a new query language. By using the same modeling tool, UML class diagram on XML document content, query syntax and semantics, it allows consistently performing all the processes such as searching and saving XML document from/to object-oriented database.

Keywords : XML, XML-GL, XML-GDM, UML, Object Modeling

1. 서 론

웹 문서의 표준으로 폭넓게 활용되고 있는 XML(eXtensible Markup Language)[1]로 작성된 문서가 널리 이용됨에 따라

† 정 회 원 : 한국연구재단 인문사회연구지원단 단장
†† 종신회원 : 전북대학교 전자정보공학부 교수
논문접수 : 2009년 3월 13일
수 정 일 : 1차 2010년 3월 2일
심사완료 : 2010년 3월 17일

XML 문서에 대하여 원하는 정보를 검색하고 추출하기 위한 새로운 언어의 필요성이 증가하고 있다[2]. 이러한 XML 문서에 표현된 정보로부터 원하는 정보를 보다 편리하게 추출하기 위해서는 사용자가 보기 쉽도록 질의와 질의에 의해서 추출된 결과 정보를 시각화하기 위한 스타일시트 문법 등이 필요한데, 이들의 기능을 모두 대신할 수 있는 대표적인 언어가 XML-GL(XML Graphical Language)[2-4]이다. XML-GL은 시각적 구조와 시각적 연산을 기반으로 정의된 질의 문법(Syntax)과 의미(Semantics)를 갖고 있기 때문에, 다른 XML 질의 언어보다 데이터를 구조화하는데 보다 논리적 언어라고 할 수 있다. 따라서 XML-GL을 이용하여 XML 문서에 대한 질의를 수행하기 위해서는 먼저 XML 문서를 시각적으로 표현해야 하는데 이때 사용되는 데이터 모델이 XML-GDM (XML Graphical Data Model)이다. 특히 XML-GL 이외에 대표적인 XML 문서에 대한 질의 언어에는 XML-QL[5], XQuery[6] 등이 있다[8].

한편 최근 급증하는 인터넷의 활용으로 웹 기반의 멀티미디어 문서를 보다 용이하게 저장, 관리하기 위한 방법으로 XML 문서를 객체지향적인 접근법에 의해 모델링하고 저장하기 위한 많은 연구나 시스템들이 있다[9]. 이러한 XML 문서에 대한 객체지향적 모델링을 위한 대표적인 도구가 UML(Unified Modeling Language)[10]이다. UML은 초기에는 소프트웨어 개발을 위한 통합된 방법론으로 활용되었으나 현재는 다양한 객체에 대한 모델링 언어로 보다 많이 활용되고 있다. 즉 현재 UML은 객체지향 분석과 설계를 위하여 클래스 다이어그램을 비롯한 다양한 다이어그램들을 지원하며, 이들로부터 각종 데이터베이스 스키마와 객체지향 코드를 생성해 주는 도구로 널리 이용되고 있다.

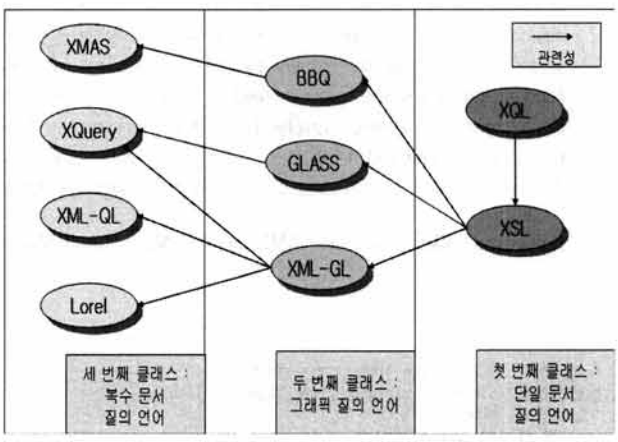
본 논문에서는 웹 기반 정보를 표현하는 표준으로 널리 활용되는 XML에 대한 효율적인 그래픽 질의 언어를 모델링하기 위하여 우선적으로 필요한 XML 문서 모델링 방안을 제안한다. 즉 XML 문서를 시각적으로 표현하여 사용자의 가독성을 높이고 XML 문서에 대한 시각적 질의를 효율적으로 지원할 수 있는 시각적 데이터 모델, 즉 XML 문서에 대한 객체 모델링 방안을 제안하고자 한다. 이를 위하여 XML 문서, XML DTD(Document Type Definition), XML에 대한 질의 등을 UML의 클래스 다이어그램으로 사상시켜 객체지향 모델의 다양한 속성을 갖는 XML 문서를 생성한다. 단순히 XML 문서에 표현된 정보만을 모델링하고자 하는 기존 연구들과는 달리, 제안된 객체 모델링 방법으로 생성된 XML 문서 객체 모델은 XML 문서에 표현된 정보들을 효과적으로 질의할 수 있는 그래픽 XML 질의 언어의 기반이 되는 질의 데이터 모델이 된다. 제안된 방법을 사용하여 XML-GDM으로 표현된 XML 문서를 UML 기반의 객체지향 그래픽 데이터 모델로 시각화하여 변환, 저장, 관리함으로써 사용자가 XML 문서에 대한 검색 질의를 직관적이고 시각적으로 표현할 수 있는 바탕을 마련할 수 있게 된다. 따라서 XML 문서에 대한 기존의 객체 모델링 방법과 비교하면 다양한 객체지향적 특징을 가지고 있으며 객체 모델링 도구로 널리 활용되고 있는 UML의 표기법을 이용하게 되어 새로운 질의 표현 방식을 익히지 않더라도 XML 기반의 웹 문서에 대한 시각적이고 직관적인 질의를 수행할 수 있다. 이를 통해 XML-GDM으로 표현된 XML 문서의 내용, 질의의 문법과 의미를 UML 클래스 다이어그램이라는 동일한 모델링 도구를 적용함으로써 XML 문서를 객체지향 데이터베이스에 저장, 검색하는 등 일련의 모든 과정을 일관적으로 수행할 수 있다.

2. 관련연구

XML 데이터를 대상으로 하는 그래픽 질의 언어에는 XML-GL과 BBQ가 있다. 웹 문서의 다양한 생성 및 활용도에 따라 단일 문서에 대한 질의 보다 복수 문서에 대한 질의의 필요성이 더 부각되고 있다. 따라서 복잡한 문서 구조를 보다 쉽게 이해할 수 있고 접근할 수 있는 시각화 방법으로 XML-GL과 같은 그래픽 사용자 인터페이스가 제안되었다. 그래픽 질의 언어를 대표하는 언어중의 하나인 XML-GL은 XML 그래프에 의해 XML 문서와 DTD를 그래프로 표현하는 방법으로, XML-GL은 그래프 기반의 논리적 언어로 그래프를 패턴으로 취급하고, 질의 결과를 찾기 위해서 그래프 기반으로 매칭시키는 방식에 의해서 처리된다. XML-GL에 직접적으로 영향을 준 언어는 G-log로 논리적인 반응을 그래픽 언어로 복잡한 개체질의 표현이 용이하며 훗날 WG-log로 발전되어 웹기반의 질의 언어로 활용되고 있다.

2.1 XML 질의 언어의 개요

XML 질의 언어는 XSL, XQL과 같은 단일 문서 질의 언어, XML-GL, GLASS, BBQ와 같은 그래픽 질의 언어, 그리고 Lorel, XML-QL, XQuery, XMAS와 같은 복수 문서 질의 언어 등 세 가지 종류의 클래스로 나누어 볼 수 있는데 특징별로 분류하면 (그림 1)과 같다[JEON 2004]. 본 논문에서 제안하는 UML 기반 그래픽 웹 질의 언어는 (그림 1)에서 두 번째 클래스인 그래픽 질의 언어로 분류할 수 있다.



(그림 1) XML 질의 언어의 클래스별 분류

2.1.1 단일 문서 질의 언어

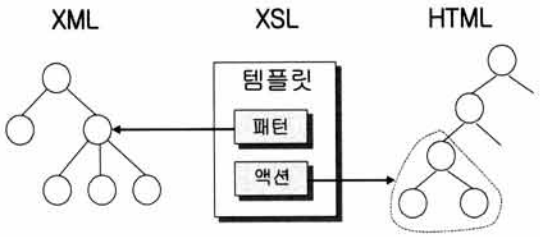
단일 문서 질의 언어는 그 표현력이 간단한 질의를 작성하는 수준으로 한정되어 있으며 주로 한 문서 안에서 질의를 처리하는데 이용한다. 단일 문서 질의 언어에는 XSL과 XQL이 있는데 내용을 요약하면 다음과 같다.

(1) XSL

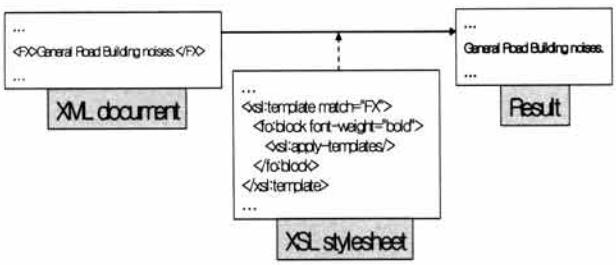
XSL은 (그림 1)에서와 같이 단일 문서 질의 언어로 첫 번째 클래스에 속한다. XSL은 XML 데이터와 문서 포맷팅을 위하여 제안된 포맷팅 언어로 논리적인 구조만 가지는 XML 인스턴스(instance)를 웹이나 문서로 출력하기 위한 표현 형식을 정의하는 역할을 수행한다. XSL은 XML을 변환하기 위한 언어인 XSLT(XML Transformation), XML 문서의 일부분을 참조하거나 접근하기 위하여 XSLT에 의해 사용되는 표현 언어인 XPath(XML Path Language), 그리고 출력형태 즉, 포맷팅의 의미를 명세화를 위한 XML 어휘인 XML-FO(XML Formating Objects)의 세 부분으로 구성된다[W3C 03]. 또한 XSL은 다른 질의 언어의 기초가 되며 패턴과 탬플릿으로 구성된다. 패턴은 원시 트리의 노드와 일치하고 탬플릿은 결과 트리를 만들기 위해 실체화된다[ABIT 97, 문현창 03].

(그림 2)는 XSL의 패턴과 템플릿을 이용하여 XML 문서를 HTML 문서로 변환하는 과정을 표현한 것이다.

(그림 3)에서 패턴은 XML 문서에서 루트 엘리먼트(element) 밑의 <FX> 엘리먼트를 찾도록 해주며, 템플릿은 이 엘리먼트가 "bold" 폰트 속성을 갖도록 해 준다.



(그림 2) XSL의 패턴과 템플릿

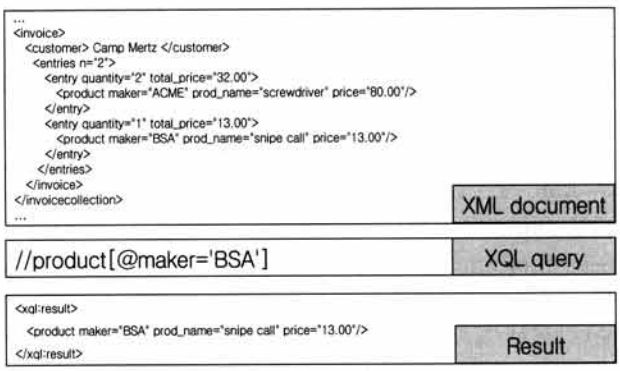


(그림 3) XSL 질의 처리 예

(2) XQL

J. Robie, J. Lapp, D. Schach가 제안한 XQL은 인터넷상의 XML 문서에서 효율적으로 데이터를 추출하고 질의하기 위해 XSL의 패턴 정의 부분을 확장한 질의 언어이다. XSL에 비해 간단하게 문장을 구성할 수 있으며 XML 트리의 각 엘리먼트 노드를 따라 이동하면서 패턴과 일치하는 노드들의 집합을 반환하도록 지원한다. 즉, XQL은 XML 문서의 엘리먼트와 텍스트를 주소화하고 필터링하기 위한 수식이다. 따라서, XQL은 특정 엘리먼트와 특정 문자들을 포함하는 노드들을 찾기 위하여 명확하고 이해 가능한 수식을 제공하며 문법적으로 간단하고 간결하게 설계하는 것을 목표로 한다[김노환 2002].

XQL에서 엘리먼트의 구조와 무관하게 엘리먼트에 접근하려면 (그림 4)의 질의와 같이 자손 연산자 "//"를 사용하는데, 이 경우 구조상에서 위치와 관계없이 <product> 엘리먼트에 접근하게 된다. 또한, XQL은 엘리먼트 뿐만 아니라 엘리먼트 내의 속성들에 대해서도 속성 연산자 "@"을 이용하여 접근할 수 있다. 결국 (그림 4)의 질의는 "maker" 속성이 "BSA"인 <product> 엘리먼트를 검색한다.



(그림 4) XQL 질의 처리의 예

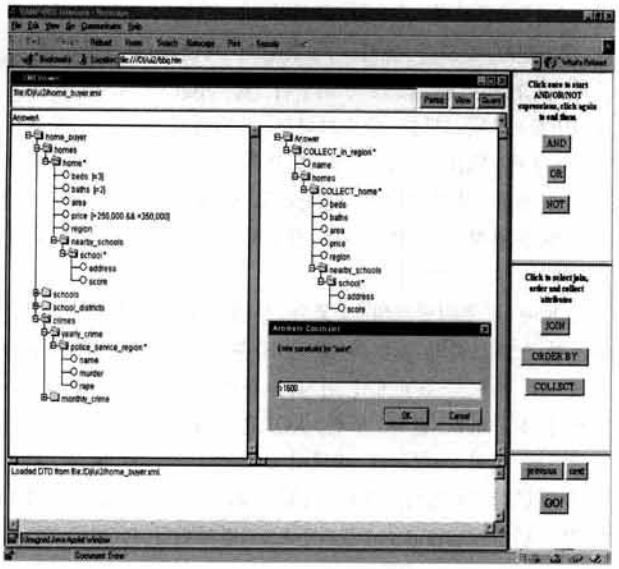
2.1.2 그래픽 질의 언어

XML-QL, XQL, XSL, XQuery 등 지금까지 제안된 대부분의 질의 언어들은 텍스트 기반 질의 언어로 그래픽 인터페이스를 지원하지 않는 특징을 갖고 있다. 이러한 질의 언어들을 사용하려면 복잡한 질의 문법과 데이터베이스 스키마 정보 등의 내용을 기억해야 하기 때문에 일반 사용자들에는 적합하지 않다. 또한 이러한 질의 언어들은 DTD (Document Type Definition) 또는 메타데이터 질의에 대한 사전 지식을 필요로 한다. 따라서 XML-GL이나 GLASS 같은 그래픽 질의 언어들은 이러한 어려움을 극복하기 위해서 복잡한 문서 구조에 좀 더 쉽게 접근할 수 있는 시각화 방법의 일환으로 제안되었다[NI 2003].

(1) BBQ

BBQ는 트리 구조에 기반하여 XML 데이터를 시각화하는 윈도를 사용한다[XMAS 99].

BBQ의 실행화면은 질의 조합을 위한 조건과 응답 창으로 구성되어 있으며 생성된 XMAS 질의의 head와 body 부분을 정의한다. 실제 응답은 별개의 창에 표시되는 mediator를 통해 반환된다. 이런 방식으로 BBQ는 XML 데이터의 검색과 질의를 자연스럽게 통합한다.[BARU 98].

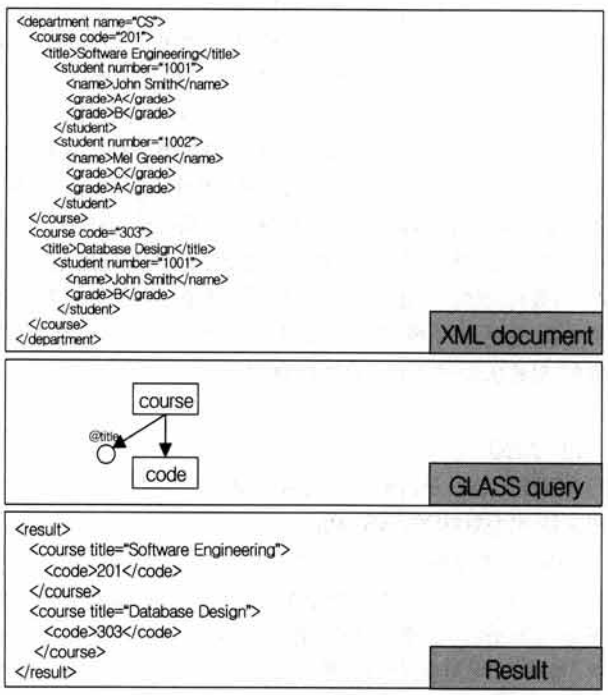


(그림 5) BBQ 실행 화면의 예

(2) GLASS

GLASS는 XML 데이터에 대해 질의를 수행하기 위한 시각화 언어이다. GLASS는 다른 XML 질의 언어와 마찬가지로 XML 문서를 데이터베이스에 저장하기 위한 데이터 모델인 ORA-SS (Object-Relationship- Attribute model for Semi-Structured data)를 갖고 있다[DOBB 2001]. GLASS는 사용자가 직관적인 방법으로 질의를 생성할 수 있게 해준다. 또한 질의의 그래픽 부분은 명확하게, 텍스트 부분은 이해하기 쉽게 조합하였다[NI 2003].

(그림 6)의 질의는 <course> 엘리먼트의 <code> 애트리뷰트(attribute)와 <title> 엘리먼트를 검색하고 재구성하여



(그림 6) GLASS 질의 처리의 예

<title> 애트리뷰트와 <code> 하위 엘리먼트를 갖는 <course> 엘리먼트를 출력한다.

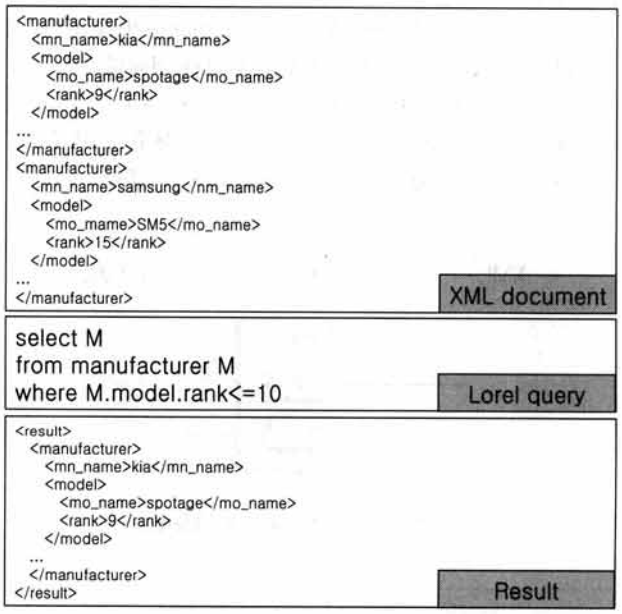
2.1.3 복수 문서 질의 언어

Lorel, XML-QL, XQuery, XMAS는 복수 문서 질의 언어로서 세 번째 클래스에 속한다. 웹 문서의 특징을 고려하면 복수문서 질의 언어가 단순 문서 질의 언어보다 더 유용하다.

(1) Lorel

Lorel은 원래 반구조화된 데이터를 위해 설계되었으나 XML 문서 질의까지 확장된 것이며 SQL/OQL(Object Query Language) 스타일로 사용자가 편리하게 사용할 수 있게 설계된 언어이다. 문서의 구조를 미리 알지 못할 때 매우 유용하고 강력한 경로 표현 능력을 갖는 장점이 있다[MCHU 99].

Lorel 질의는 SQL처럼 조건을 명시하는 WHERE 절, 검색할 문서의 위치를 지정하는 FROM 절, 그리고 선택할 속성 목록을 지정하는 SELECT 절로 구성되어 있다. 결국 (그림 7)의 질의는 <manufacturer> 엘리먼트의 하위 엘리먼트 중 <rank>가 10 이하인 것을 검색한다.

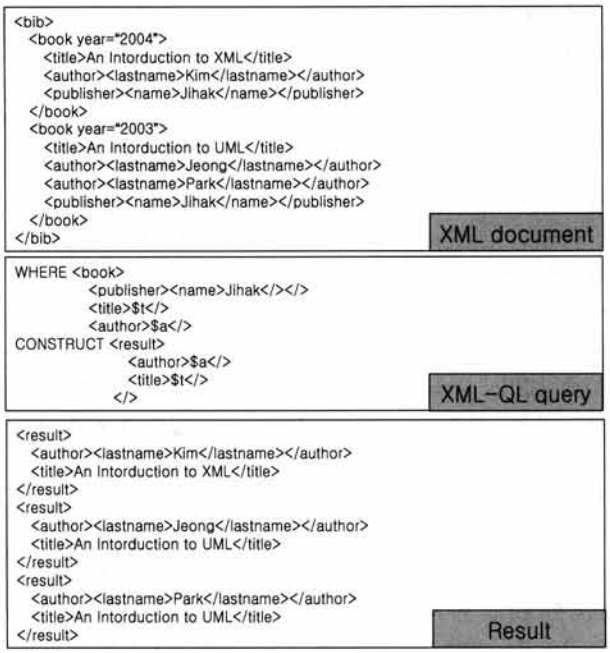


(그림 7) Lorel 질의 처리의 예

(2) XML-QL

XML-QL은 XML 문서에서 효율적으로 데이터를 추출하고 질의하기 위한 선언적 형태의 질의 언어이다. SQL의 SELECT~WHERE 구조와 유사한 구조를 갖고 있는 XML-QL은 XML 패턴을 기술하는 WHERE 절과 질의 결과를 XML로 재구성하기 위한 CONSTRUCT 절로 구성 되어 있다[김노환 2002].

(그림 8)의 질의는 새로운 <result> 엘리먼트를 생성하고, <author>와 <title> 모두를 반환하여 그룹으로 묶어 새로운 XML 문서를 생성한다.



(그림 8) XML-QL 질의 처리의 예

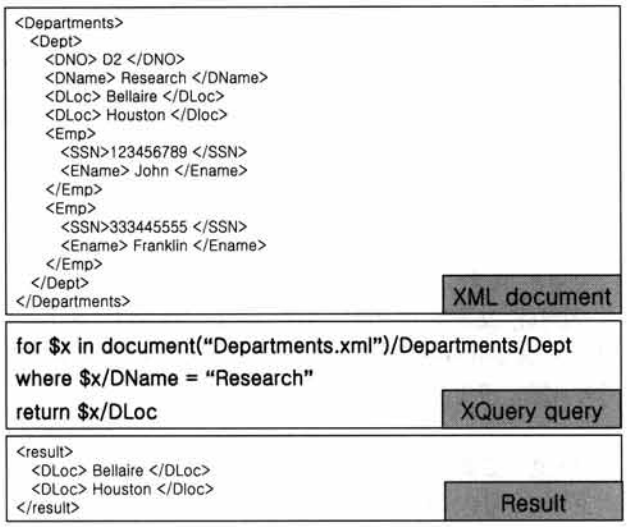
(3) XPath

XPath는 XML 데이터의 특정한 부분을 지정하기 위한 언어로 XSLT와 XPointer등에서 사용하기 위해 디자인 되었다. XML특정 부분의 위치를 표현하기 위해 위치경로를 사용한다. 위치 경로를 통해서 데이터 안의 위치를 단계별로 표현할 수 있다. 그러나 XPath는 SQL의 검색 기능에 해당하는 join, 그룹핑, 정렬, 데이터 타입등을 지원하지 못한다. 이러한 한계를 보안한 질의 언어가 XQuery이다.

(4) XQuery

XQuery는 최초의 XML 질의 언어 중의 하나인 Quilt에서 파생되었으며 XML 문서의 구조를 이용하여 전체가 구조화되거나 부분적으로 구조화된 XML 문서에 대한 질의를 수용할 수 있다. W3C에 의해 XML 질의어의 표준으로 제정되었고, SQL의 SELECT~FROM~WHERE 문과 유사한 기능을 가지는 FLWOR (for-let-where-order by-return) 표현식을 가진다. FLWOR 표현식은 for 절, let 절, where 절, order by 절, 그리고 return 절로 구성되며, 각 절 안에 또 다른 FLWOR 표현식이 중첩될 수 있다. XQuery의 for 절, where 절, return 절은 SQL의 FROM 절, WHERE 절, SELECT 절과 유사한 의미를 가진다. 그리고 order by 절은 순서를 재구성할 때 사용하고 let 절은 표현식을 하나의 변수로 치환하는 의미를 가진다[김서영 2004, BOAG 2005].

(그림 9)의 질의는 FLWOR 표현식을 이용하여 "Research" 부서의 위치를 구한다. for 절은 변수 $x에 주어진 XML 엘리먼트들을 하나씩 바인딩하며, where 절은 $x에 바인딩된 엘리먼트들 중 <DName>이라는 하위 엘리먼트의 값이 "Research"인 것을 선택한다. 끝으로 return 절은 $x에 바인딩된 엘리먼트의 하위 엘리먼트들 중에서 <DLoc> 엘리먼트들을 반환한다.

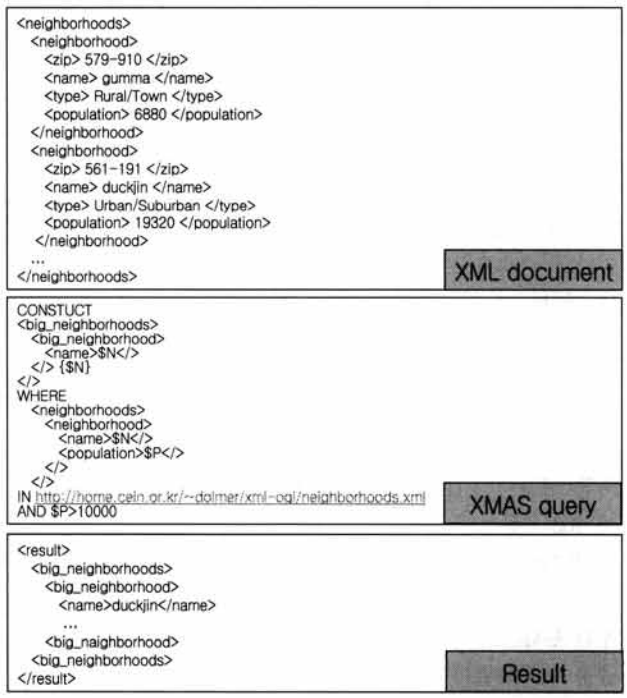


(그림 9) XQuery 질의 처리의 예

(5) XMAS

XMAS도 역시 세 번째 클래스에 속하며 질의를 만들기 위한 그래픽 사용자 인터페이스인 BBQ를 정의한다. XML-QL에서 아이디어를 얻은 XMAS는 선언적인 규칙기반 질의 언어로 CONSTRUCT~WHERE 구조를 갖고 기존에 있는 객체들로부터 새롭게 합성된 XML 객체들을 생성하기 위한 강력한 그룹화와 순서 구성을 특징으로 한다[XMAS 99].

(그림 10)에서 XMAS 질의는 크게 head(CONSTRUCT 절)와 body(WHERE 절)로 나눌 수 있다. CONSTRUCT 절은 SQL의 SELECT 절과 유사하며, WHERE 절은 SQL의 WHERE 절과 비슷한 역할을 수행한다. 결국 (그림 10)의 질의는 인구가 10,000명 이상인 규모가 큰 도시의 이름을 검색한다.



(그림 10) XMAS 질의 처리의 예

2.2 XML 문서 모델링

이기종 시스템들간의 구조적인 문서 교환을 위한 XML 문서[ARBO 98]의 사용이 증가함에 따라 이에 대한 모델링 방법들이 연구되고 있다. 이에 본 절은 XML 문서 모델링 방법에 관한 연구들에 대해 고찰한다.

2.2.1 모델링의 개념 및 단계

모델은 대상이 되는 시스템을 분석하여 정리하고 표현하는 것이다. 모델을 작성하는 작업을 모델링이라고 한다. 모델링은 크게 요구모델링, 분석모델링, 설계모델링, 구현모델링의 4가지 공정으로 나눌 수 있다.

(1) 요구모델링

요구모델링은 사용자의 요구를 파악한다. 프로젝트에 참여하는 모든 이해당사자의 요구사항을 요구사항 명세서(Software Requirement Specification)의 형태로 정형화 하여 프로젝트 완료 후에는 검수 및 인수인계의 기준이 되는 중요한 단계이다. 이 공정에서는 UML Use Case Diagram 등을 사용하여 시스템에 필요한 기능을 담당자 및 유저의 요구사항을 확인하여 시스템의 대상범위를 확정한다.

(2) 분석모델링

분석 모델링은 요구 모델링의 결과를 바탕으로 시스템의 구조를 명확히 한다. 시스템화의 대상에 대한 분석에 중점을 두며 정적인 모델은 Object Diagram, Class Diagram을 사용하고, 동적인 모델은 Sequence Diagram, Activity Diagram을 이용하여 기술한다.

(3) 설계 모델링

설계 모델링은 분석모델링에서 도출된 시스템 대상에 대한 구체적인 설계 방법을 정의한다. 설계 모델링에서 시스템 내부사양을 설계할 때 정적인 모델은 Class Diagram을 동적인 모델은 Sequence Diagram, Activity Diagram을 이용한다.

(4) 구현 모델링

구현 모델링은 시스템의 구성요소를 정의한다. 프로그램이나 리소스파일의 구성은 Component Diagram으로 기술한다. 또한, 프로그램코드수주의 레벨을 Class Diagram이나 Sequence Diagram으로 기술하기도 한다.

2.2.2 UML 다이어그램 모델링

UML 다이어그램을 이용하여 XML 문서를 모델링하기 위한 연구에는 [BERN 2003, 하얀 2003b, BERN 2004, 이돈양 2004] 등이 있다.

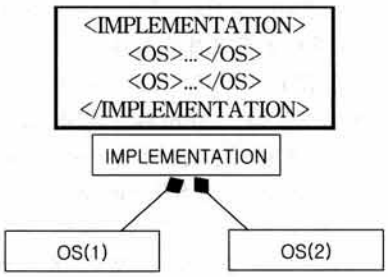
[JENS 2003]은 웹상에서 XML 정보원(information source)에 대해 빠르고 쉬운 그래픽 검색이 가능하며 XML DTD로부터 UML 다이어그램을 자동적으로 구성하기 위한 알고리즘을 제안하였다. XML 데이터의 구조는 XML 정보원을 기술하는 DTD로부터 생성된 UML 다이어그램에 의해 시각화된다. DTD에서 UML로의 자동 변환은 세 개의 단계로 이루어진다. 첫째, 주어진 DTD는 XML 데이터의 필수적 구조는 남아 있지만 DTD를 단순화하는 변환규칙을 적용하여 단순화한다. 둘째, 단순화된 DTD로부터 UML 다이어그램이 자동으로 생성된다. 셋째, 생성된 다이어그램은 후 처리 과정을 거쳐 좀 더 쉽게 이해할 수 있도록 만들어진다.

[VELA 2003, BERN 2004]은 XML 스키마를 대상으로 객체지향 설계 언어인 UML의 클래스 다이어그램으로 모델링하였다. [방승윤 2003]은 모델링된 데이터를 관계 데이터베이스에 저장하는 통합 모델링 방법을 제시하였고 [BERN 2003, BERN 2004]는 XML 스키마 정보의 손실 없이 간결하고 의미 있게 표현하는 UML 프로파일(profile)과 XML 스키마를 UML 프로파일로 변환하는 규칙들을 정의하였다. 또한 XML 스키마의 주요 개념들로 구성된 변환 규칙과 UML 확장을 기술하였다. 한편 [VELA 2003]은 XML 스키마의 타입에 따라 서로 다른 스테레오타입(stereotype)을 선택하여 UML을 확장 기술하였다.

그리고, [하얀 2003b]는 웹 문서에 대한 채널을 정의하는 CDF(Channel Definition Format)와 직접적인 접근을 나타내는 인터페이스를 기술하기 위한 OSD(Open Software Description)에 대한 객체 모델링을 제안하였다. CDF 문서에 대한 객체 모델링은 [김상은 2001]의 객체 모델링 방법을 적용하였으나, OSD의 경우 구현과 의존 관계 등을 위한 새로운 모델링 방법을 적용하였다.

(그림 11)는 OSD 문서를 UML 클래스 다이어그램으로 사상한 결과를 나타낸다.

[이돈양 2004]은 패턴 및 클래스에 대한 부분을 정형화, 표준화하기 위한 방법으로 XMI(XML Metadata Interchange Format)를 사용하였으며 패턴을 정형화하기 위한 방법으로 UML 클래스 다이어그램을 이용하여 시스템을 설계하였다.



(그림 11) OSD 모델링의 예

2.2.3 DOM

DOM은 프로그램이나 스크립트 등을 웹 문서의 구조, 스타일 그리고 내용 등에 대하여 동적 접근과 수정을 할 수 있도록 플랫폼에 독립적이고 언어 중립적인 인터페이스를 제공하는 문서 객체 모델이다. 즉, DOM은 표준화된 API(Application Programming Interface)를 제공하여 웹 문서에 접근하고 조작하기 위한 방법이다[W3C 2004]. HTML 문서에 대한 DOM의 표현은 계층 구조로 표현된다. 즉 HTML이나 XML내의 태그 요소들은 DOM을 통해 객체화되고, 객

체화된 각각의 노드들은 논리적인 구조를 나타내는 트리 형태로 구성된다.

DOM 트리를 이용하여 명세서에 정의된 인터페이스를 기반으로 노드에 접근할 수 있는 다양한 기능들을 구현할 수 있다. 따라서 DOM 트리를 이용하여 정적인 웹 문서의 한계를 넘어설 있다. 이와 같이 일반적인 문서의 내용은 계층 구조를 갖는 객체 기반 구조로 표현할 수 있으며, 표현된 구조는 프로그래밍을 통하여 쉽게 핸들링이 가능하다. 특히 XML의 경우는 구조화된 문서로 작성되므로 문서 일부에 대한 접근, 데이터베이스의 저장, 구조화된 질의 및 검색 등이 용이하게 된다. 그러나 DOM은 문서의 논리적 구조를 정의하는 객체 기반 구조이지만 트리 계층 구조를 가짐으로써 시간의 흐름에 따른 객체 관계 등을 나타낼 수 없고 클래스의 세부적인 정보인 애트리뷰트나 집단화 관계 등을 제대로 표현하지 못하는 단점이 있다[하얀 2003b].

2.3 XML과 UML

본 절에서는 문서와 데이터를 구조적으로 표현할 수 있는 메타언어인 XML의 개념과 구성 그리고 XML 문서 명세화 방법에 대해 고찰하고, UML의 확장 메커니즘과 UML 클래스 다이어그램에 대해 기술한다.

2.3.1 XML

XML은 문서의 개념적인 논리 구조와 내용 구조를 기술할 수 있다. 그리고 복잡한 문서와 데이터를 포함하는 멀티미디어 문서, 하이퍼링크를 포함하는 하이퍼미디어 문서까지도 작성할 수 있는 문서를 표현하고 상호 교환하기 위한 전자 문서 표현 형식이다[BRAY 2004].

XML의 특징을 살펴보면 첫째, XML은 최소화한 처리 명령으로 각 문서를 형식적으로 정의하여 서로 다른 시스템/응용 프로그램 사이의 문서 교환이 가능하도록 해준다.

둘째, XML로 작성된 문서는 XML로 마크업한 태그를 데이터베이스 필드화하여 내용을 기반으로 한 제한 검색이 가능하다.

셋째, XML을 이용하면 화면상에 문서의 내용을 표시할 때, 태그에 따른 조판 및 배치 명령을 사용하여 문서의 외형을 다양하게 표시할 수 있다.

넷째, XML 문서는 ASCII 텍스트형 태그로 문서에 포함되므로 시스템이나 응용 프로그램에 상관없이 문서의 상호 교환이 가능하다는 장점을 가진다.

(1) XML 선언

사용된 XML의 버전을 기술하는 부분으로 사용된 XML의 버전, XML 문서에 사용된 문자의 인코딩(encoding) 방식, 외부 마크업 선언의 존재 유무로 구성된다.

다음은 XML 선언의 예로써, XML 버전 1.0에 의한 XML 문서이며, 문서에서 사용한 문자 집합은 "EUC-KR"이고, 외부 마크업 선언은 존재하지 않는다는 것을 알 수 있다.

```
<?xml version="1.0" encoding='EUC-KR'
 standalone='yes'?>
```

(2) XML 문서형 선언

XML 문서 인스턴스에서 사용할 태그와 이 태그들을 이용한 XML 문서의 논리적/계층적 구조를 정의하는 부분으로 XML 문서의 논리적 구조에 대한 제한을 표현한다. 결국 문서형 선언을 통해 해당 문서 클래스에 대한 문법을 제공하는 마크업 선언이 이루어지며, 해당 문서 클래스에 대하여 정의된 이러한 문법을 문서형 정의(DTD)라고 한다[Bray 2004].

다음은 문서형 선언의 예이다.

```
<!DOCTYPE greeting [ <!ELEMENT greeting
(#PCDATA)> ]>
```

해당 문서 클래스에 대한 문법을 제공하는 문서형 선언은 엘리먼트 형 선언, 속성 리스트 선언, 엔티티 선언, 표기법 선언, 처리명령 선언, 주석 선언 등을 포함한다[Bray04].

① 엘리먼트 형 선언(Element Type Declaration)

엘리먼트가 가질 수 있는 내용에 대한 제한을 정의하며 이를 통해 XML 문서의 논리적 구조(트리 구조)를 파악할 수 있다. 엘리먼트 형 선언은 해당 엘리먼트의 이름(엘리먼트 형)과 해당 엘리먼트가 가질 수 있는 엘리먼트 내용 부분으로 구성되며, 선언 형식은 다음과 같다.

```
<!ELEMENT 엘리먼트_이름 엘리먼트_내용>
```

엘리먼트 내용 부분에 올 수 있는 요소로는 "EMPTY", "ANY", 내용모델(content model), 혼합내용(mixed content) 등이 있다. "EMPTY"는 엘리먼트 내용은 내용을 가지지 않는 빈 엘리먼트를 정의하고 "ANY"는 내용으로 임의의 엘리먼트나 문자 데이터를 가지는 엘리먼트를 정의한다. 그리고 내용모델은 자식 엘리먼트만을 내용으로 포함할 수 있고 엘리먼트를 정의하고 혼합내용은 문자 데이터나 지정된 자식 엘리먼트를 내용으로 가지는 엘리먼트를 정의한다. 내용모델에는 자식 엘리먼트의 순서나 선택을 지정하기 위한 연결자(connector)와 자식 엘리먼트의 발생 횟수를 지정하기 위한 발생지시자를 사용할 수 있는데, 연결자에는 ','과 '|'이 있고 발생지시자에는 '?', '+', '*' 등이 있다.

다음은 엘리먼트 형 선언의 예이다.

```
<!ELEMENT br EMPTY>
<!ELEMENT container ANY>
<!ELEMENT p (#PCDATA|emph)* >
<!ELEMENT %name.para; %content.para; >
```

② 속성 리스트 선언(Attribute-List Declaration)

엘리먼트의 속성들과 이 속성들이 가질 수 있는 값들을 정의하는 속성 리스트의 선언 형식은 다음과 같다.

```
<!ATTLIST 엘리먼트_이름 [속성_이름 속성_형 기본값_선언]* >
```

속성 형은 해당 속성의 값으로 사용할 수 있는 값에 대한 특성을 정의하는 것으로서 문자열 형, 토큰형, 열거형이 있다. 기본값 선언은 해당 속성의 값을 필수적으로 지정해야 하는지의 여부와 해당 속성의 값이 주어지지 않았을 경우에 사용할 기본값(default value)을 정의한다.

다음은 속성 리스트 선언의 예이다.

```
<!ATTLIST termdef
        id      ID      #REQUIRED
        name    CDATA   #IMPLIED>
<!ATTLIST list
        type (bullets|ordered|glossary) "ordered">
<!ATTLIST form
        method  CDATA   #FIXED "POST">
```

(3) XML 문서 인스턴스

실제 XML 문서의 내용을 포함하는 부분인 XML 문서 인스턴스는 DTD에 선언된 엘리먼트와 속성 리스트를 이용하여 태깅된 문서 형태로 작성한다. 한 엘리먼트의 시작태그는 '<엘리먼트_이름>'으로, 끝태그는 '</엘리먼트_이름>'의 형태로 기술한다.

다음은 XML 문서 인스턴스의 예이다.

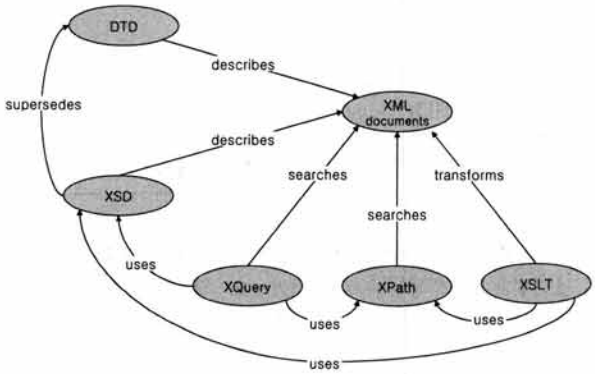
```
<greeting>Hello, world!</greeting>
```

다음은 DTD를 포함하고 있는 유효한 XML 문서에 대한 간단한 예이다.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE greeting [ <!ELEMENT greeting (#PCDATA)> ]>
<greeting>Hello, world!</greeting>
```

(4) XML 문서 명세화

(그림 12)는 XML 문서와 DTD, 그리고 명세언어인 XSD (XML Schema Definition language), 질의 언어인 XQuery, 표현 언어인 XSLT 등의 관계성을 나타낸 것이다[ERL 2004]. XML 문서를 기술하는 명세화 방법에는 DTD와 XSD가 있고 검색에 대한 명세화 방법에는 XQuery, XPath가 있으며 변환과 관련된 명세화 방법에는 XSLT가 있다.



(그림 12) XML 명세화 도구간의 관계성

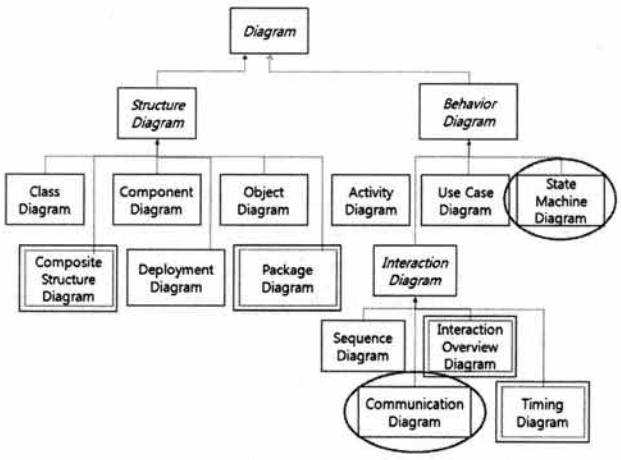
2.3.2 UML

UML은 Rumbaugh의 OMT(Object-Modeling Technique) 방법론, Booch의 Booch 방법론, 그리고 Jacobson의 OOSE (Object-Oriented Software Engineering) 방법론을 통합하여 만든 것으로 현재 많은 CASE 도구들이 이를 지원하고 있다. 그리고 객체지향 시스템 모델을 작성하기 위한 객체지향적 분석과 설계 개념 및 표기법을 제공하는 방법론으로 객체 기술을 연구하는 OMG(Object Management Group)의 공식 승인을 받았다. UML은 시스템의 서로 다른 관점에서 보여주는 뷰들을 정의하는데, 각각의 뷰들에는 그것을 표현할 수 있는 다이어그램이 존재한다[OMG 2003]. 또한 UML 2.0에서는 다양한 용도에 맞게 스테레오 타입으로 정의하고 그룹지은 Profile을 통하여 J2EE/EJB 컴포넌트 Profile, COM 컴포넌트 Profile, .NET 컴포넌트 Profile등을 정의하여 활용한다[OMG 2008].

2.3.3 UML 다이어그램

UML을 사용하여 시스템 상호 작용, 업무 흐름, 객체 간의 메시지 전달, 시스템의 구조, 컴포넌트 관계 등을 표현할 때 다양한 다이어그램이 활용된다. 다음은 UML2.0에 대한 다이어그램이다[OMG 2007a][OMG 2007b]. 그림에서 이탤릭체로 표현된 'Diagram', 'Structure Diagram', 'Behavior Diagram', 'Interaction Diagram'은 실제로 존재하는 다이어그램이 아닌 여러 가지 다이어그램의 상위로 표현되었다.

(그림 13)에서 이탤릭체로 표현된 다이어그램의 이름을 통해 다이어그램들의 특성을 파악해 볼 수 있다. 바로, 구조적인 것과 동적인 것이다. 즉 13개의 다이어그램을 구조적인 관점을 표현하는 여섯 개의 다이어그램과 동적인 관점을 나타내는 일곱 개의 다이어그램으로 분류할 수 있다. 그리고, 동적인 관점을 나타내는 일곱 개의 다이어그램 중에서 네 개의 다이어그램(시퀀스, 인터렉션 오버뷰, 커뮤니케이션, 타이밍)이 인터렉션의 하위로 표현되어 있음을 볼 수 있다. UML의 이러한 다이어그램들을 이용하여 소프트웨어 시스템의 구조적인 관점과 동적인 관점, 그리고 물리적인 관점을 표현하는 것이다.



(그림 13) UML 2.0 다이어그램

(그림 13)에서 사각형으로 표현된 다이어그램들(Composite Structure Diagram, Package Diagram, Interaction Overview Diagram, Timing Diagram)이 추가된 것이며, 타원으로 표현된 것 중에서 'Communication Diagram'은 이전 버전의 다이어그램 중에서 'Collaboration Diagram'이라는 다이어그램이 변경된 것이다. 타원으로 표현된 State Machine Diagram은 이전 버전의 Statechart Diagram의 이름이 변경된 것이다.

2.3.3.1 Sequence Diagram

Sequence Diagram은 시스템 내부에서 일어나는 동적인 면을 표현하는 다이어그램이다. Sequence Diagram은 객체 사이에서 이뤄지는 메시지 교환을 순차적으로 나타낸다. 즉, 각각의 객체 사이에 오가는 내용, 순서, 시간을 명확하게 보여준다. 시퀀스 다이어그램은 인터랙션 다이어그램의 가장 일반적인 형태이며, 객체 간에 송·수신하는 메시지를 시간의 흐름에 따라 나열한 다이어그램이다.

이러한 시퀀스 다이어그램은 유스케이스의 흐름(절차) 속에서 객체들이 서로 송·수신하는 메시지를 표현하는 용도로 사용한다.

2.3.3.2 Use Case Diagram

유스케이스 다이어그램은 시스템이 제공하는 기능과 시스템을 사용하는 사용자(또는타 시스템)를 표현하는 다이어그램이다. 이러한 유스케이스 다이어그램은 시스템의 요건을 명시하기 위해 사용한다. 개발 주기 초기에 주로 사용자의 기능적 요구사항을 기술하는 데 사용한다. 유스케이스 다이어그램 작성시 사용자의 관점에서 작성하는 것이 필요하다. 시스템 개발 초기에 사용자가 원하는 기능을 제대로 도출해 내고, 개발 기간 내내 불충분한 요구사항으로 인한 변경 작업을 줄이기 위해서도 그렇게 해야 한다.

2.3.3.3 Communication Diagram

커뮤니케이션 다이어그램은 이전 버전의 콜래버레이션 다이어그램이며, 구성요소 간의 관계와 요소 간에 주고받는 메시지를 표현한 다이어그램이다. 메시지의 순서는 알 수없기 때문에 메시지 순서를 나타내려면 메시지 전송 순서를 알 수 있도록 숫자를 이용하여 표현해야 한다.

2.3.3.4 Timing Diagram

타이밍 다이어그램은 시간의 흐름에 따른 상태를 나타낸다. 가로 축에 시간을 표현하고 세로 축에 상태를 나타내어 시간이 지나감에 따른 상태 변화를 표현하는 다이어그램이다. 타이밍 다이어그램에 등장하는 요소는 프레임, 생명선(Lifeline), 상태, 메시지, 이벤트등이 있다.

2.3.3.5 Activity Diagram

Activity Diagram은 시스템의 동적인 측면을 모델링하는 것으로, 대체로 전체 시스템이나 서브 시스템을 대상으로 생성하지만, 함수나 클래스를 대상으로 사용할 수도 있다. 그리고 유스케이스를 대상으로 작성할 수도 있다.

액티비티 다이어그램은 대체로 다음과 같은 두 가지 방법으로 이용한다.

2.3.3.6 StateMachine Diagram

스테이트머신 다이어그램(StateMachine Diagram)은 클래스나 전체 시스템을 대상으로 하여, '이벤트에 따른 객체의 상태 변화'를 나타내는 다이어그램이다. 스테이트머신 다이어그램에는 상태와 전이 그리고 이벤트 등이 표현되며, 상태가 전이되기 위한 이벤트, 조건, 액션도 표현된다.

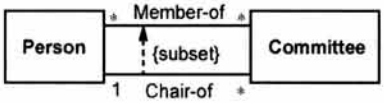
(1) UML의 확장 메커니즘(Extension Mechanisms)

여러 가지 객체 지향 기법을 결합한 형태인 UML은 기본 메커니즘으로 표현할 수 없는 부분을 지원하기 위하여 제한조건, 엘리먼트 특성(element properties), 스테레오타입과 같은 확장 메커니즘을 제공한다[OMG 2003]. 이러한 확장 매커니즘은 XML 문서 및 질의를 모델링하는데 필요하다.

① 제한조건

조건이나 가정을 규정하는 모델 엘리먼트들 간의 의미적인 관계로 항상 참이 되어야 한다. 제한조건의 내용은 '{ }' 안에 텍스트 스트링으로 기술한다.

(그림 14)는 사람과 위원회의 관계를 표현한 것으로 위원회의 의장은 1명이며 여러 명으로 구성된 위원 집합의 {subset}임을 나타낸다.



(그림 14) 제한조건의 예

② 엘리먼트 특성

많은 엘리먼트들은 시각적으로 표현되지 않은 세부적인 특성을 갖는다. 따라서 사용자는 태그된 값을 이용하여 새로운 엘리먼트의 특성을 정의할 수 있다. 엘리먼트 특성에 대한 표기는 '{ }' 안에 한 개 이상의 '키워드=값'을 ','로 구별하여 기술한다.

(그림 15)는 저자, 마감일 등 엘리먼트 특성을 표현한 것이다.

{author="Changyun Jeong", deadline=31-March-2005, status=analysis}

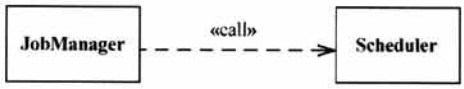
(그림 15) 엘리먼트 특성의 예

③ 스테레오타입

스테레오타입은 엘리먼트의 새로운 클래스이며 기존의 엘리먼트와 같은 형태를 가지나 다른 의도로 사용되는 하위클래스(subclass)로서 기존 클래스에 부가적인 제약 사항을 추가한 것을 말한다. 스테레오타입의 표기는 '« »'안에 스테레

오타입의 이름을 키워드 스트링으로 기술한다.

(그림 16)은 JobManager와 같은 형태를 가지는 하위클래스 Scheduler를 표현한 것이다.



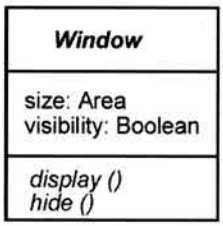
(그림 16) 스테레오타입의 예

(2) UML 클래스 다이어그램

UML 클래스 다이어그램은 클래스와 그들 간의 관계로 구성되는데[RUMB 99, OMG 2003] 그 내용은 다음과 같다.

① 클래스

클래스는 클래스 이름, 애트리뷰트 리스트(attribute list)와 오퍼레이션 리스트(operation list) 부분으로 구성된다. 애트리뷰트 리스트는 클래스의 멤버들을 나타내고, 오퍼레이션 리스트는 멤버들을 다루는 함수를 나타낸다.



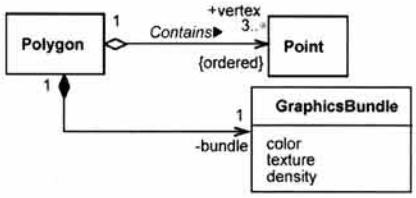
(그림 17) 클래스의 예

② 집단화와 컴포지션(Composition)

집단화는 상위클래스와 하위클래스가 'part-of' 관계이며 실선과 '◇'로 표현한다. 2개 이상의 집단화 관계에서 선택적으로 하위클래스가 발생하는 경우 'xor' 관계를 갖는다. 이때 여러 개의 집단화 관계를 점선으로 연결하고 '{xor}'라는 제한조건을 준다.

컴포지션 관계는 집단화 관계가 강한 경우이며 상위 클래스의 생명 주기에 맞추어 하위 클래스의 생명주기가 결정된다. 컴포지션 관계는 실선과 '◆'로 표현한다.

(그림 18)은 다각형과 다각형의 정의, 다각형과 색, 표면의 성질, 밀도 등의 그래픽 요소들 간의 관계를 나타낸다.

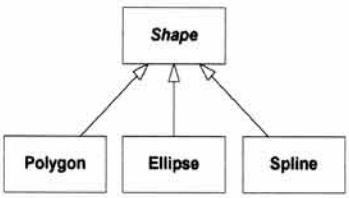


(그림 18) 집단화 관계의 예

③ 일반화와 특수화(Specialization)

일반화와 특수화에서 상위 클래스와 하위 클래스는 'is-a' 관계를 나타낸다. 일반화는 공통된 속성 또는 연산을 가진 클래스들에서 공통 요소들을 추출하여 상위 클래스를 구성하는 것을 의미한다. 반대로 새로운 요소를 추가하여 하위 클래스를 구성하는 것을 특수화라고 한다. 상위 클래스와 하위클래스 사이를 실선으로 연결하며 상위 클래스 쪽에 '△'을 연결한다.

(그림 19)는 다각형(polygon), 타원(ellipse), 스플라인(spline)의 공통요소를 추출하면 모양(shape)이라는 상위클래스가 만들어지는 것을 표현한 것이다.

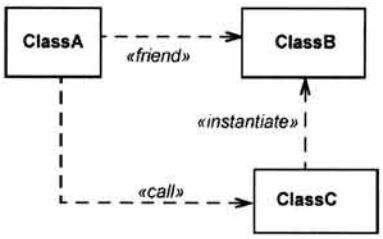


(그림 19) 일반화와 특수화의 예

④ 의존(Dependency) 관계

한 쪽 클래스의 변화가 다른 클래스에 영향을 줄 수 있는 관계로 점선으로 나타내며 방향성을 나타낼 수 있다.

(그림 20)는 클래스 A, B, C 간의 의존 관계를 표현한 것이다.



(그림 20) 의존 관계의 예

3. UML과 XML-GDM

3장에서는 UML의 클래스 다이어그램과 주요 확장 메커니즘을 기술하고, XML-GL의 데이터 모델인 XML-GDM에 대해 기술한다.

3.1 UML

3.1.1 UML 클래스 다이어그램

UML 클래스 다이어그램은 클래스와 그들 간의 관계로 구성된다. UML 클래스 다이어그램을 구성하는 각 구성요소에는 클래스(Class), 집단화(Aggregation) 관계, 일반화(Generalization)/특수화(Specialization) 관계, 의존(Dependency) 관계 등이 있다.

(1) 클래스(Class)

클래스는 클래스 이름, 애트리뷰트 리스트(Attribute list)

와 오퍼레이션 리스트(Operation list) 부분으로 구성된다. 애트리뷰트 리스트는 클래스의 멤버들을 나타내고, 오퍼레이션 리스트는 멤버들을 다루는 함수를 나타낸다.

(2) 집단화(Aggregation)와 합성화(Composition)관계

집단화 관계는 상위클래스와 하위클래스가 'part-of' 관계이며 '◇'로 표기한다. 강한 집단화 관계로 합성화 관계가 있는데, 이것은 하위클래스가 상위클래스와 같은 생명주기를 갖는 강한 형태의 구성 관계를 나타낸다. 표기는 '◆'로 한다.

2개 이상의 집단화 관계에서 선택적으로 하위클래스가 발생하는 경우 'xor' 관계를 갖는다. 이 때, 여러 개의 집단화 관계를 점선으로 연결하고 '{xor}'이라는 제한조건을 준다.

(3) 의존(Dependency) 관계

한 쪽 클래스의 변화가 다른 클래스에 영향을 줄 수 있는 관계로 점선으로 나타내며 방향성을 나타낼 수 있다.

클래스는 클래스 이름, 속성 리스트(Attribute List)와 오퍼레이션 리스트(Operation list) 부분으로 구성된다. 속성 리스트는 클래스의 정적인 멤버, 클래스의 특성을 표현하는 변수들을 나타내고, 오퍼레이션 리스트는 클래스의 속성을 처리하기 위한 동적인 멤버, 즉 클래스의 행위들을 표현하는 함수(메소드)들이다. 클래스는 세 개의 행으로 나누어진 사각형으로 표현하며, 각 행에 클래스 이름, 속성 리스트, 오퍼레이션 리스트를 정의한다.

집단화 관계는 상위 클래스와 하위 클래스의 관계가 'part-of' 관계임을 나타내며 '◇'로 표기한다. 2 개 이상의 집단화 관계에서 선택적으로 하위 클래스가 발생하는 경우 'xor' 관계를 갖는다. 이때, 여러 개의 집단화 관계를 점선으로 연결하고 '{xor}'이라는 제한조건을 준다.

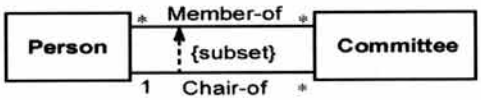
일반화는 공통된 속성이나 연산을 가진 클래스들에서 공통 요소들을 추출하여 상위 클래스를 만들어내는 것을 의미하며, 이와 반대로 상위 클래스에 새로운 요소를 추가하여 하위 클래스를 만들어내는 것을 특수화라고 한다. 일반화/특수화 관계는 상위 클래스와 하위 클래스를 실선으로 연결하고 상위 클래스 쪽에 빈 삼각형을 표시한다. 의존 관계는 한 클래스의 변화가 다른 클래스에 영향을 주는 관계로, 점선으로 표시하며 방향성을 나타낼 수 있다.

3.1.2 UML의 일반 확장 메커니즘(General Extension Mechanism)

UML은 여러 가지 객체지향 기법을 결합한 형태로 다양한 확장 메커니즘을 제공한다. 제한조건, 엘리먼트 특성, 스테레오 타입이 이에 해당된다.

(1) 제한조건(Constraints)

제한조건은 텍스트로 표현된 의미적 조건이나 제약사항으로, 모델 요소들 사의의 의미적 관계이다. 제한조건은 항상 참이 되어야 하며, 표기는 '{ }' 안에 조건을 나타내는 텍스트 스트링을 기술한다.



(그림 21) 제한조건의 예

(2) 엘리먼트 특성(Element Properties)

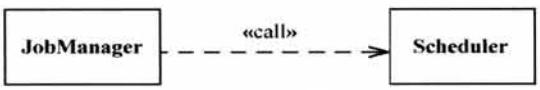
엘리먼트 특성은 직접 시각적으로 표기되지 않는 모델 요소의 세부적인 특성을 말하는 것으로, 모델 요소의 속성(Attribute), 연관(Association), 태그가 붙여진 값(Tagged Value) 등을 포함한다. 사용자는 태그가 붙여진 값을 이용하여 새로운 모델 요소의 특성을 정의할 수 있다. 엘리먼트 특성은 한 개 이상의 '키워드=값' 쌍을 ','로 구분하여 '{ }' 안에 기술한다.

```
{author="dhpark", deadline=31-December-2008,
status=analysis }
{ abstract }
```

(그림 22) 엘리먼트 특성의 예

(3) 스테레오타입(Stereotypes)

스테레오타입은 기존의 UML 요소를 기반으로 UML 어휘를 확장하여 새로운 메타 클래스의 인스턴스를 만들기 위해 고안되었다. 스테레오타입은 기존의 메타모델 요소와 같은 형태를 가지나 다른 의도로 사용되는 메타모델 요소의 하위 클래스이다. 스테레오타입은 '« »'안에 스테레오타입의 이름을 키워드 스트링으로 기술하고 이를 모델 요소의 이름 앞이나 위에 표기한다.



(그림 23) 스테레오타입의 예

3.2 XML-GDM

XML 문서에 대한 다른 질의 언어들과 같이 XML-GL도 XML 문서를 데이터베이스에 저장하기 위한 데이터 모델을 갖고 있는데, 이를 XML-GDM이라고 한다. XML-GDM은 실제 XML 문서(문서 인스턴스)와 XML 문서의 구조를 나타내는 XML DTD를 표현하기 위해 사용된다. 또한 XML-GDM의 구문은 XML-GL의 질의를 표현하기 위해서도 사용된다. XML-GDM의 주요 표기법은 <표 1>과 같다.

XML-GDM에서 XML 문서들은 XML 그래프라고 불리는, 레이블이 있는 방향성 그래프로 표현된다. XML 그래프는 엘리먼트 노드, 애트리뷰트 노드, 콘텐츠 노드로 구성되며 각각의 노드들은 포함 아크(Containment Arc)와 참조 아크(Reference Arc)로 연결된다. 포함 아크는 엘리먼트 노드들 사이의 부모-자식 관계를 나타내기 위해 사용되고, 참조 아크는 엘리먼트 노드와 콘텐츠 노드 또는 엘리먼트 노드와 애트리뷰트 노드를 연결하기 위해 사용된다. 엘리먼트 노드

〈표 1〉 XML-GDM 표기법

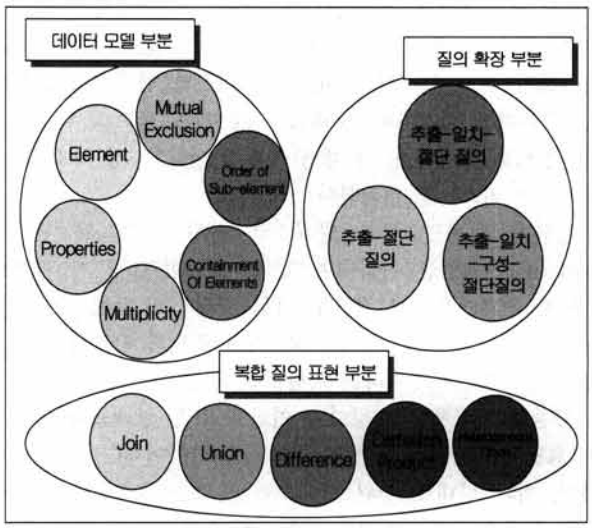
Feature		XML DOCUMENT	Graphic Representation
Element		<order>...</order>	order
Containment of Elements		<order> <delivery>...</delivery> </order>	order delivery
Order of Sub-elements		<order> <delivery>...</delivery> <broker>...</broker> <item>...</item> </order>	order delivery broker item
Mutual Exclusion		<delivery> <addr>...</ addr > </delivery> or <delivery> <ref>...</ref> </delivery>	delivery xor addr ref
Element with PCDATA Content		<date><year>2007</year></date>	date year 2007
ID Attribute		<order no="1">... </order>	order no 1
IDREF Attribute		<ref custom="C0001">... </ref>	ref customer=C00001 person
Mixed Content		<order> An year of order is <date> <year>2007</year> </date> </order>	order xor 2007 year date content
Multiplicity	0:1	<order>...</order> or <order><broker>...</broker><order>	order (0:1) broker
	1:N	<order><item>...</item></order> or <order><item>...</item> ...</order>	order (1:n) item
	0:N	<book>...</book> or <book><author>...<author>...</book>	book (0:n) author

는 엘리먼트 이름을 레이블로 하는, 레이블이 있는 사각형(Labeled Rectangle)으로 나타내고, 애트리뷰트 노드와 콘텐츠 노드는 레이블이 있는 원으로 나타낸다. 이때 해당 노드가 콘텐츠 노드인 경우는 흰색 원으로, 애트리뷰트 노드인 경우는 검은색 원으로 표현한다.

그리고 아크들은 노드들을 연결하는, 레이블이 있는 화살표로 표현하는데 포함 아크는 "CONT"라는 지정된 레이블을 사용하거나 이를 생략할 수 있다. 또한 참조 아크는 애트리뷰트의 이름이나 #PCDATA 형인 엘리먼트의 이름을 레이블로 사용한다. 한편 포함 아크에서 자식 노드들의 순서는 첫 번째 자식 노드를 가리키는 포함 아크에 짧은 빗금을 긋고 나머지 자식 노드들은 시계반대 방향으로 순서화하여 표현한다.

4. UML 클래스 다이어그램으로 사상하기 위한 XML 문서 객체 모델링

4장에서는 XML-GDM으로 표현된 XML 문서를 객체지향 데이터베이스에 저장하고 이를 검색하기 위하여 UML 클래스 다이어그램으로 변환하는 방안을 제안한다. XML-GL을 UML 클래스 다이어그램으로 사상하기 위한 XML 객체 모델링은 (그림 24)와 같이 XML-GDM을 UML 클래스 다이어그램으로 변환하는 데이터 모델 부분, XML-GL의 질의 유형에 따라 UML 다이어그램으로 변환하는 질의 확장 부분으로 나눌 수 있다. 그리고 질의 확장 부분은 다시 단순한 형태의 XML-GL 질의 표현을 변환하는 단순질의 표현 부분과 XML-GL의 복합질의 표현을 변환하는 복합질의 표현 부분으로 나눌 수 있다. 본 논문에서는 이 중에 XML 문서에 대한 시각적 질의를 효과적으로 지원하기 위한 데이터 모델 부분, 즉 XML-GDM을 UML 클래스 다이어그램으로 변환하는 부분만을 다룬다.

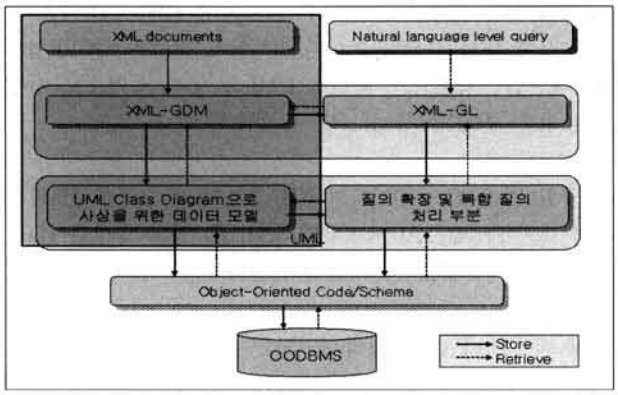


(그림 24) 객체 모델링 구성요소

4.1 UML 클래스 다이어그램으로 사상을 위한 데이터 모델

XML-GL을 이용하여 XML 문서를 검색하기 위하여 XML 문서를 데이터베이스에 저장하기 위한 데이터 모델이 XML-GDM이라면, UML 클래스 다이어그램으로 사상하기 위한 데이터 모델은 XML 문서를 UML 클래스 다이어그램으로 변환하여 객체지향 데이터베이스에 저장할 수 있게 해준다.

(그림 6)은 XML-GDM으로 표현된 XML 문서를 UML 클래스 다이어그램으로 사상하는 데이터 모델을 통해 XML 문서를 객체지향 데이터베이스에 저장하고, XML-GL을 이용하여 표현된 자연어 질의를 UML 다이어그램으로 사상하여 객체지향 데이터베이스에 저장된 XML 문서를 검색하기 위한 전체 시스템 구성도이다. 본 논문에서는 (그림 25)의 왼쪽에 표시된 부분인, XML-GDM으로 표현된 XML 문서를 UML 클래스 다이어그램으로 사상하는 부분만을 다룬다.



(그림 25) UML 클래스 다이어그램으로 사상을 위한 데이터 모델

4.2 사상 규칙

UML은 많은 모델링 요소들을 제공하고 있지만 XML 문서를 UML 클래스 다이어그램으로 사상하기 위한 XML 문서 객체 모델링을 위해서는 보다 명확하게 생성 규칙을 설정해야 한다. 이를 위해 본 논문에서는 XML-GDM으로 표현된 XML 문서의 DTD와 XML 문서 인스턴스로부터 UML 클래스 다이어그램을 생성하기 위해 다음과 같은 정의와 규칙들을 적용한다.

[정의 1] XML-GDM의 요소들은 UML 클래스 다이어그램 집합으로 사상된다.

먼저 XML-GDM으로 표현된 XML 문서를 UML 클래스 다이어그램으로 사상하기 위한 방법을 요약하면 다음 <표 2>와 같다.

다음 각 절에서는 XML-GDM의 각 요소들을 UML 클래스 다이어그램으로 사상하는 방법을 자세하게 기술한다.

〈표 2〉 XML-GDM과 UML 사상

XML-GDM		UML 기반 그래픽 데이터 모델
엘리먼트	Default	Class
	Containment of Elements	집단화 관계
	Order of Sub-element	{ordered} 제한조건
	Mutual Exclusion	{XOR} 제한 조건
값 (Property)	Element with PCDATA Content	"String" Class로부터 상속
	Mixed Content	{XOR} 제한 조건
	ID Attributes	Private Attribute
	IDREF Attributes	Public Attribute
다중성	0:1	'0..1'
	1:N	'1..*'
	0:N	'0..*'

4.2.1 엘리먼트

XML-GDM의 엘리먼트를 UML로 사상하는 규칙은 [규칙 1]과 같다.

[규칙 1] XML-GDM의 엘리먼트는 UML에서의 클래스로 사상한다.

예 1) 엘리먼트

XML-GDM	UML 기반 그래픽 데이터 모델
order	order

하위 엘리먼트를 갖는 엘리먼트는 그 특성에 따라 [규칙 2], [규칙 3], [규칙 4]를 적용한다. [규칙 2]는 하위 엘리먼트를 포함하는 엘리먼트에 대한 사상 규칙으로, 하위 엘리먼트를 포함하는 엘리먼트는 UML의 집단화 관계를 이용하여 사상한다.

[규칙 2] XML-GDM의 엘리먼트가 하위 엘리먼트를 포함하고 있으면 각 하위 엘리먼트는 [규칙 1]을 적용하여 UML에서의 클래스로 각각 사상하고, 상위 클래스와 하위 클래스는 집단화 관계를 갖는다.

예 2) 하위 엘리먼트를 포함하는 엘리먼트

XML-GDM	UML 기반 그래픽 데이터 모델
order delivery	order delivery

[규칙 3]은 하위 엘리먼트들 사이에 순서 관계가 지정된 경우에 대한 사상 규칙으로, UML의 제한 조건 확장 메커니즘을 이용하여 사상한다.

[규칙 3] XML-GDM의 엘리먼트가 순서 관계가 지정된 하위 엘리먼트를 포함하고 있으면 UML에서는 집단화 관계에 {ordered} 제한조건을 지정한다.

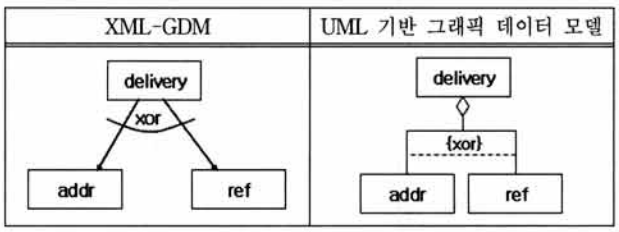
예 3) 순서가 있는 하위 엘리먼트

XML-GDM	UML 기반 그래픽 데이터 모델
order delivery broker item	order {ordered} delivery broker item

[규칙 4]는 하위 엘리먼트들 사이에 선택 관계가 지정된 경우에 대한 사상 규칙으로, UML의 제한 조건 확장 메커니즘을 이용하여 사상한다.

[규칙 4] XML-GDM의 엘리먼트가 선택 연결자가 지정된 하위 엘리먼트를 포함하고 있으면 UML에서는 집단화 관계에 {xor} 제한조건을 지정한다.

예 4) 선택 관계가 지정된 하위 엘리먼트



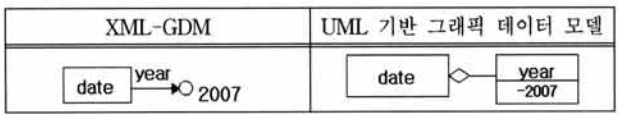
4.2.2 값(Property)

XML-GDM의 프로퍼티(Property)는 문자 데이터(Character Data; CDATA)나 파싱된 문자 데이터 스트링(Parse Character Data String; PCDATA)처럼 표시할 수 있는 값을 나타내며, 엘리먼트 내용(PCDATA)과 속성으로 구분할 수 있다. 여기서 속성은 다시 ID, CDATA인 경우와 IDREF, IDREFS인 경우로 나눌 수 있다. ID, CDATA는 자신만의 고유한 값을 가지는 속성이며, IDREF, IDREFS는 다른 속성의 값을 참조하는 속성이다. XML-GDM의 프로퍼티에 대한 사상은 [규칙 5], [규칙 6], [규칙 7], [규칙 8]을 적용한다.

[규칙 5]는 XML-GDM의 Property 중 엘리먼트 내용에 대한 사상 규칙으로, 엘리먼트의 내용은 텍스트 문자열(#PCDATA) 형이므로 UML의 기본 자료형 클래스인 "String"을 상속받아 표현한다.

[규칙 5] XML-GDM의 엘리먼트 내용 부분은 UML의 기본 데이터 클래스인 "String"으로부터 상속을 받는다. 엘리먼트의 내용은 상속 받은 데이터 클래스의 사적 속성(Private Attribute)이 된다.

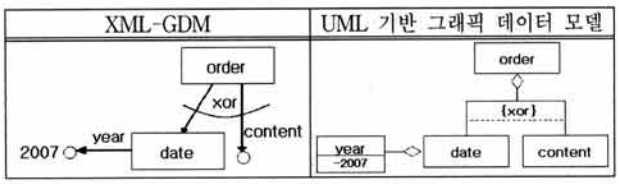
예 5) 엘리먼트 내용



[규칙 6]은 XML-GDM의 Property 중 혼합 내용(Mixed Content)에 대한 사상 규칙으로, [규칙 4]와 [규칙 5]를 이용하여 사상한다.

[규칙 6] XML-GDM의 혼합 내용 부분의 경우 엘리먼트 부분은 [규칙 2]와 [규칙 5]를 적용하고 문자열 부분은 "content"라는 이름의 임시 클래스를 생성하고 [규칙 5]를 적용한 후, 엘리먼트 부분과 문자열 부분에는 [규칙 4]에 의한 선택 관계를 지정한다.

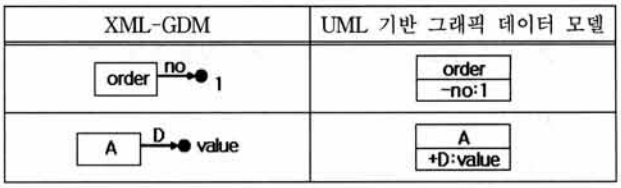
예 6) 혼합 콘텐츠



[규칙 7]은 XML-GDM의 Property 중 ID와 CDATA 형의 속성에 대한 사상 규칙이다. ID 형의 속성은 IDREF형의 속성에 의해 참조되기 때문에 UML의 공적 속성으로 사상하여 외부 클래스에서 참조할 수 있도록 한다.

[규칙 7] XML-GDM에서 CDATA 형의 속성은 UML의 사적 속성(Private Attribute)이 되며, ID 형의 속성은 UML의 공적 속성(Public Attribute)이 된다.

예 7) CDATA, ID 형의 속성

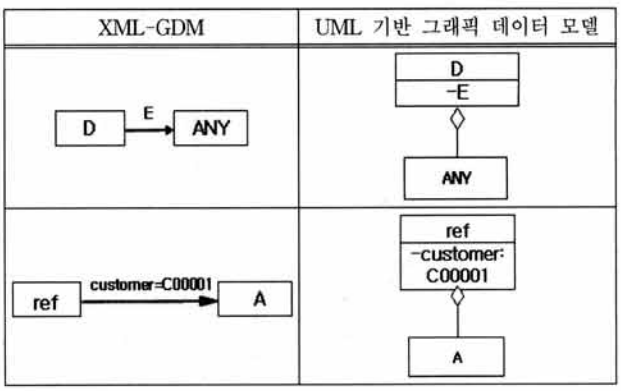


다음은 XML-GDM의 Property 중 IDREF와 IDREFS 형의 속성에 대한 사상 규칙이다. DTD에서의 IDREF 형이나 IDREFS 형은 다른 엘리먼트의 ID 속성을 값을 참조하므로, 참조되는 클래스를 나타내가 위해 임시 클래스를 생성하여 사상한다.

[규칙 8] XML-GDM의 IDREF(S) 형의 속성은 해당 엘리먼트 클래스의 사적 속성으로 사상한다. 그리고 "ANY"라는 이름의 임시 클래스를 생성한 후 "ANY" 클래스와 IDREF(S) 속성을 가지는 엘리먼트 클래스 사이에 [규칙 2]를 적용한다.

XML 문서 인스턴스에서는 "ANY" 클래스를 실제 ID 속성을 가지는 엘리먼트 클래스로 대체하여 IDREF(S) 속성을 가지는 엘리먼트 클래스와 ID 속성을 가지는 엘리먼트 클래스 사이에 [규칙 2]를 적용하여 사상한다.

예 8) IDREF와 IDREFS 형의 속성



4.2.3 다중성

XML-GDM에서 다중성은 '0:1', '1:N', '0:N'이 있으며 각각 0 또는 1번, 1번 이상, 0번 이상 발생됨을 의미하는 것으로 다음 규칙에 의해 사상한다. 다중성이 지정되지 않은 경우 디폴트 값은 1이다.

[규칙 9] XML-GDM의 다중성인 '0:1', '1:N', '0:N'은 UML에서 각각 '0..1', '1..*', '0..*'으로 표시한다.

예 9) 다중성

XML-GDM	UML 기반 그래픽 데이터 모델
order (0:1) broker	order 0..1 broker
order (1:n) item	order 1..* item
book (0:n) author	book 0..* author

4.3 사상 알고리즘

XML-GDM 기반의 XML DTD와 XML 문서를 입력 받아 UML 클래스 다이어그램을 생성하는 사상 알고리즘은 [알고리즘 1]과 같다.

[알고리즘 1] XML 문서를 UML 클래스 다이어그램으로 사상하는 알고리즘

```
입력 : XML DTD, XML 문서
출력 : UML 클래스 다이어그램
begin   // XML DTD와 XML 문서 처리
  switch(선언된 DTD 요소) {
    case : element
      make-class();        // 엘리먼트 클래스 생성
        make-class-note();   // 클래스에 설명 추가
        while(element-content-empty()) {
          // element 클래스가 공집합이 아니면 DTD와 문서처리
          if(connector-content-empty()) then {    // 엘리먼트내
          연결자가 있는 경우
            make-attlist-function();
            for(public  attribute 개수) {
              make-subclass();  // 하위 클래스 생성
              make-aggregation();   // 집단화 관계 설정
              multiplicity-indicator();   // 다중성 처리
            }
            connector-function(); // 연결자 조건에 따른 제한 조
            건 부여
          }
          else {   // 연결자 없이 #PCDATA가 오는 경우
            make-inherit-from();   // 기본 타입으로부터 상속
            switch(exception-type) {   // 클래스의 예외처리
              case : containment of elements
                make-attlist-function();   // attribute 추가
                make-include-aggregation();   // 집단화 관계
              case : mutual exclusion
                make-exclude-stereotype();   // xor 제한조건
              case : order of sub-element
                make-inherit-ordered();   // ordered 제한조건
            }
          }
        }
      case : attlist
        make-attlist-function();   // 엘리먼트 클래스에 애트리뷰
트와 타입 추가
    }
end.
```

다중성의 종류에 따라 발생 횟수를 표시하는 함수와 알고리즘은 [알고리즘 2]와 같다.

[알고리즘 2] 다중성 처리 알고리즘

```
multiplicity-indicator()
begin
  switch (multiplicity-indicator-type)
    // 발생 지시자의 종류별 처리
    case : "0 : 1"  // 0번 또는 1번 발생
           insert-option-function();
    case : "1 : N"  // 1번 이상 발생
           insert-plus-function();
    case : "0 : N"  // 0번 이상 발생
           insert-rep-function();
end
```

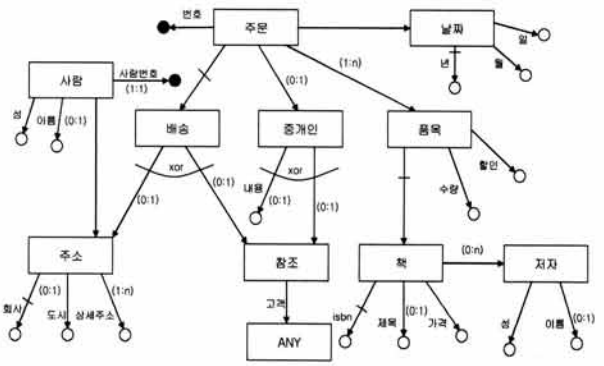
4.4 적용사례

4.4절에서는 4.3절의 규칙들을 도서 주문과 관련된 XML 문서 DTD와 XML에 적용하여 사상 규칙과 사상 알고리즘의 효용성을 검증한다. (그림 26)은 도서 주문과 관련된 문서의 구조를 나타내는 DTD("주문.dtd")이다.

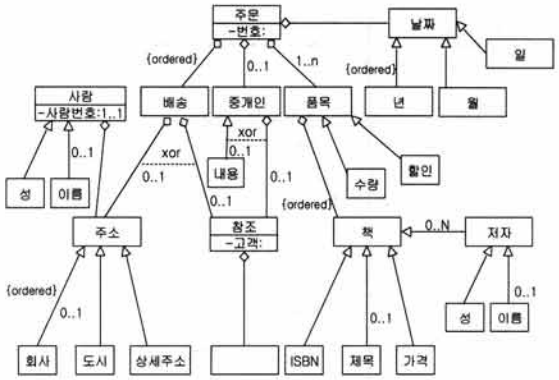
(그림 26)은 (그림 27)의 XML문서 DTD를 XML-GDM로

```
<!ELEMENT 주문 (배송, 중개인?, 품목+, 날짜)>
  <!ATTLIST 주문 번호 CDATA #REQUIRED>
  <!ELEMENT 배송 (주소|참조)>
    <!ELEMENT 주소 (회사?, 도시, 상세주소+)>
      <!ELEMENT 회사 (#PCDATA)>
      <!ELEMENT 도시 (#PCDATA)>
      <!ELEMENT 상세주소 (#PCDATA)>
    <!ELEMENT 참조 EMPTY>
      <!ATTLIST 참조 고객 IDREF>
  <!ELEMENT 중개인 (#PCDATA|참조)>
  <!ELEMENT 품목 (책, 수량, 할인)>
    <!ELEMENT 책 (ISBN, 제목?, 가격, 저자*)
     <!ELEMENT ISBN (#PCDATA)>
      <!ELEMENT 제목 (#PCDATA)>
      <!ELEMENT 가격 (#PCDATA)>
      <!ELEMENT 저자 (성, 이름)>
        <!ELEMENT 성 (#PCDATA)>
        <!ELEMENT 이름 (#PCDATA)>
    <!ELEMENT 수량 (#PCDATA)>
    <!ELEMENT 할인 (#PCDATA)>
  <!ELEMENT 날짜 (년, 월, 일)>
    <!ELEMENT 년 (#PCDATA)>
    <!ELEMENT 월 (#PCDATA)>
    <!ELEMENT 일 (#PCDATA)>
<!ELEMENT 사람 (성, 이름?, 주소)>
  <!ATTLIST 사람 사람번호 ID>
  <!ELEMENT 성 (#PCDATA)>
  <!ELEMENT 이름 (#PCDATA)>
```

(그림 26) 도서주문 관련 XML 문서 DTD



(그림 27) XML-GDM으로 표현한 "주문.dtd"



(그림 28) UML 클래스 다이어그램으로 표현한 "주문.dtd"

모델링한 결과이다.

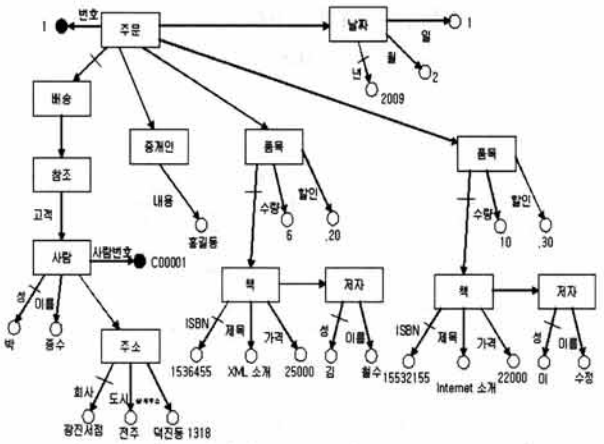
(그림 27)은 XML-GDM으로 모델링된 (그림 26)의 XML DTD를 UML 클래스 다이어그램으로 사상한 결과이다.

(그림 29)는 (그림 26)의 XML DTD에 의해 작성된 도서 관련 XML 문서("주문.xml")에 대한 예이다.

(그림 29)의 XML 문서를 XML-GDM으로 모델링하면 (그림 30)과 같다.

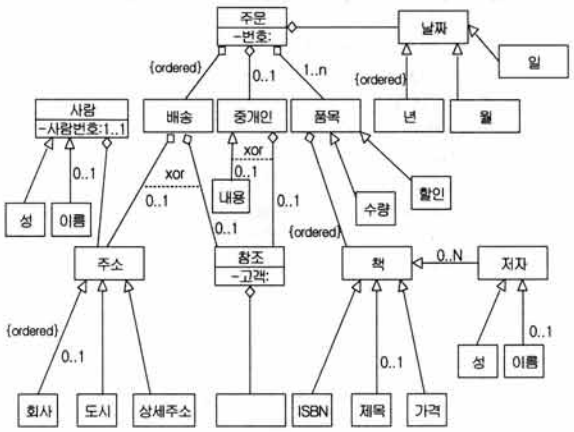
```
<?xml version="1.0" standalone="no" encoding="UTF-8"?>
<DOCTYPE 주문 SYSTEM "주문.dtd">

<주문 번호=1>
  <배송>
    <참조 고객="C00001"></참조>
  </배송>
  <중개인>홍길동</중개인>
  <품목>
    <책>
      <ISBN>15536455</ISBN>
      <제목>XML 소개</제목>
      <가격>25000</가격>
      <저자><성>김</성><이름>철수</이름></저자>
    </책>
    <수량>6</수량>
    <할인>.20</할인>
<품목>
    ...
<품목>
    <책>
      <ISBN>15532155</ISBN>
      <제목>Internet 소개</제목>
      <가격>22000</가격>
      <저자><성>정</성><이름>최고</이름></저자>
    </책>
    <수량>10</수량>
    <할인>.30</할인>
  <날짜><년>2004</년><월>2월</월><일>10</일></날짜>

<사람 사람번호="C00001">
   <성>박</성>
   <이름>종수</이름>
   <주소><회사>광진서점</회사>
      <도시>전주</도시>
      <상세주소>덕진동 1318</상세주소>
   </주소>
</사람>
   ...
<사람 사람번호="C00003">
   <성>이</성>
   <이름>수정</이름>
   <주소>
      <도시>익산</도시>
      <상세주소>모현동 56</상세주소>
   </주소>
</사람>
```

(그림 29) 도서관련 XML 문서("주문.xml")의 예



(그림 30) 도서관련 XML 문서를 XML-GL로 표현

XML-GDM으로 모델링된 (그림 30)의 XML 문서를 UML 클래스 다이어그램으로 사상한 결과는 (그림 31)과 같다.



(그림 31) 도서관련 XML 문서를 UML 클래스 다이어그램으로 표현

5. 결론 및 향후 연구 과제

본 논문에서는 WWW의 활성화와 유연하고 개방적인 속성으로 인하여 그 응용 범위가 광범위해지고 있는 XML 문서를 검색하기 위한 UML 클래스 다이어그램을 이용한 XML-GL 질의 모델링 기법을 제안하였다. 이를 위해 XML 문서와 DTD에 대한 모델링 기능과 이에 대한 시각적 질의 기능을 가지고 있는 XML-GL을 UML 다이어그램으로 사상하는 모델링 규칙과 알고리즘을 제안하였다. 또한, 모델 구성요소에 대한 제약사항을 정확하게 묘사하기 위하여 객체 제약언어인 OCL(Object Constraint Language)로 정의하였다. UML 클래스 다이어그램의 구성 요소에 대한 제약사항을 객체 제약언어인 OCL로 기술하여 기존 그래픽 모델에서 구성 요소에 대한 제약 사항을 단순히 자연어로 기술할 경우 발생하는 모호성을 배제하였다. 이를 통해 XML 문서 질의시 객체지향 속성을 적용할 수 있었으며, 결과적으로 XML 문서에 대한 검색시 UML 클래스 다이어그램과 OCL

을 이용한 질의 기능을 제공하기 위한 기반을 구축할 수 있었다.

그리고, 급증하는 인터넷의 활용으로 생성되는 다양한 XML 문서에 대하여 복합질의 유형에 따라 UML 클래스 다이어그램과 OCL을 이용한 XML-GL 질의 모델링 방안을 제시하여 증가되는 복합질의에 대한 기반을 마련하였다.

또한, 객체지향 모델링에서 그 중요성 및 활용도가 증가하는 UML 메카니즘을 적용하였기 때문에 사용자가 별도의 표기법을 익힐 필요가 없고 직관적인 사용이 가능하다. 특히, Rational Rose나 Together와 같은 상용 UML 도구 및 StarUML, Argo UML과 같은 무료 UML툴을 활용하여 JAVA, C++, C#과 같은 객체지향 코드나 스키마로의 변환 및 응용이 가능하게 되었다.

본 논문에서 제안한 UML 클래스 다이어그램을 이용한 XML-GL 질의 모델링 기법을 적용하여 XML 문서의 구조적 특징을 충분히 활용하지 못하고 있는 기존의 데이터베이스 저장 기법이나 질의 기법의 한계를 벗어나 XML 문서의 내용기반 데이터 특성과 객체지향 속성을 활용함으로써 객체지향 데이터베이스에 XML 문서를 효율적으로 저장, 관리, 검색할 수 있는 기반이 될 것이다.

향후 본 논문에서 제안한 사상 규칙과 사상 알고리즘을 객체지향 데이터베이스에 보다 체계적으로 적용할 수 있도록 하기 위하여 사상 규칙을 형식언어로 표현하기 위한 연구와 XML 문서 데이터베이스에 대한 질의를 직관적으로 시각화하기 위한 연구를 수행할 것이다. 또한 제안된 객체 모델링 기법과 질의 모델링 기법을 적용한 시각적 XML 데이터베이스 시스템을 개발하기 위한 연구도 수행되어야 한다.

참 고 문 헌

[1] [김노환 2002] XPath 질의를 기반으로 하는 DB2XML 알고리즘 설계 및 구현, 강원대학교 대학원 박사학위논문, 2002. 2.

[2] [문현창 2003] XML 구조적 특징을 이용한 온톨리지 기반의 지식 탐사 모델, 창원대학교 대학원 박사학위논문, 2003. 8.

[3] [이돈양 2004] 이돈양, 송영재, "XMI기반 객체지향 메타모델 생성," 정보처리학회논문지 D, 제11권 제2호, pp.397-406. 2004. 2.

[4] [조완섭 2003] 조완섭, "그래픽 객체 질의어에서 집합 속성과 메소드를 포함한 경로식의 시각화," 정보과학회논문지 : 데이터베이스, 제30권 제2호, 2003. 4.

[5] [하얀 2003b] 하얀, "OSD, CDF 문서로부터 UML 클래스 다이어그램으로 변환 시스템," 정보처리학회논문지 A, 제10권-A권 제5호, 2003. 10.

[6] [정창윤 2005] XML 문서를 위한 UML 기반 그래픽 웹 질의 언어의 설계 및 구현, 2005. 8

[7] [ADLE 2001] Sharon Adler, et al., "Extensible Stylesheet Language (XSL) Version 1.0," http://www.w3.org/TR/xsl/, 2001.

[8] [BALK 2002] N. H. Balkir et al., "A Graphical Query Language : VISUAL, and its Query Processing," IEEE Trans on Knowledge and Data Engineering, Vol.14, No.5, pp. 955-978, Sep., 2002.

[9] [BOAG 2004] Scott Boag, et al., "XQuery 1.0: An XML Query Language," http://www.w3.org/TR/xquery, 2004.

[10] [BONI 2000] Angela Bonifati, Stefano Ceri, "Comparative Analysis of Five XML Query Languages," ACM SIGMOD Record, Vol.29, No.3, pp.76-87, 2000.

[11] [BRA 2004] Tim Bray, et al., "XML 1.0(Third Edition)," W3C Recommendation, http://www.w3.org/TR/2004/ REC-xml-20040204, Feb., 2004.

[12] [CABO 2009] Cabot, J, Clariso, R, Riera, D, "Verifying UML/OCL Operation Contracts," Lecture notes in computer science, Vol.5423, pp.40-55, 2009.

[13] [CERI 1999] Stefano Ceri, et al., "XML-GL: a graphical language for querying and restructuring XML Documents," Computer Networks, Vol.31, pp.1171-1187, 1999.

[14] [CERI 2000] Stefano Ceri, et al., "Complex queries in XML-GL," Proc. of 2000 ACM symposium on Applied Computing(SAC2000), pp.888-893, Como, Italy, Mar., 2000.

[15] [CHRI 1994] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl, "From Structured Documents to Novel Query Facilities," ACM SIGMOD Record, Vol.23, No.2, pp.313-324, June, 1994.

[16] [COMA 2001] Sara Comai, Ernesto Damiani, and Piero Fraternali, "Computing Graphical Queries over XML Data," ACM Transaction on Information System(TOIS), Vol.19, No.4, pp371-430, 2001.

[17] [DEUT 1998] Alin Deutsch, et al., "XML-QL: A Query Language for XML," http://www.w3.org/TR/NOTE-xml-ql, 1998.

[18] [KUHL 2008] Kuhlmann, M, Gogolla, M, "Modeling and validating Mondex scenarios described in UML and OCL with USE," Formal aspects fo computing, Vol.20, No.1, pp 79-100, 2008.

[19] [NI 2003] W. Ni, T. W. Ling, "GLASS: A Graphical Query Language for Semi-Structured Data," Database Systems for Advanced Applications(DASFAA'03), 2003.

[20] [OMG 2003] OMG Unified Modeliing Language Specification Version 1.5, http://www.omg.org/docs/formal/03-03-01.pdf, Mar. 2003.

[21] [OMG 2007] OMG, Unified Modeling Language : Superstructure, version 2.1.1, 2007.

[22] [OMG 2008] OMG, UML 2.0 Superstructure Specification, http://www.uml.org.

[23] [PARK 2005] U. Park and Y. S대, "An Implementation of XML Documents Search System based on Similarity in Structure and Semantics," In Proc. of the Web Information retrieval and Integration, 2005(wiri '05), pp.97-103, April, 2005.

[24] [SENG 2008] Sengupta. S, Kanjilal.A, and Bhattacharya. S, "Requirement Traceability in Software Development Process:

An Empirical Approach," Proc. of the 19th IEEE/IFIP Symposium on Rapid System Prototyping, pp.105-111, 2008.

[25] [STRA 2007] Straeten, Ragnhild, Jonckers, Viviane and Mens, Tom, "A Formal Approach to Model Refactoring and Model Refinement," Software and Systems Modeling, Vol.6, No.2, 2007.

[26] [ZOU 2006] Zou, Joe and Pavlovski, Christopher, Modeling Architectural Non Functional Requirements: From Use Case to Control Case, IEEE ICEBE' 06, 2006.

[27] [XU 2007] Xu, Yang and Yu, Youwei, Towards Aspect Oriented Web Service Composition with UML, The 6th ICIS, 2007.

[28] [OMG 2007a] OMG, UML Infrastructure Specification v2.1.2 Nov., 2007.

[29] [OMG 2007b] OMG, UML Superstructure Specification v2.1.2 Nov., 2007.

[30] [DOD 2008] Dodani, Mahesh H., Application At Your Service, Jorunal of Object Technology, Vol.7, No,7, 2008.

[31] [FOW 2003] Fowler, Martin, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Ed., Addison-Wesley, 2003.

[32] [WAR 2003] J.Warmer and A. Kleppe. The Object Constraint Language : Getting Your Models ready for MDA. Addison-Wesley Professional, 2nd edition, 2003.

박 대 현

e-mail : empire@nrf.go.kr

1996년 숭실대학교 정보산업학과(이학석사)

2010년 현재 전북대학교 전산통계학과 박사과정 수료

2009년 6월~현 재 한국연구재단 인문사회연구지원단 단장

관심분야 : 컴퓨터알고리즘, 데이터 모델링 등

김 용 성

e-mail : yskim@chonbuk.ac.kr

1978년 고려대학교 수학과(이학사)

1984년 광운대학교 전산학과(이학석사)

1992년 광운대학교 전산학과(공학박사)

1985년~현 재 전북대학교 전자정보공학부 교수

관심분야 : XML, 정보검색, 웹서비스 등