UML 클래스 다이어그램을 이용한 XML 문서 구조 다이어그래밍

채 원 석† · 하 얀†† · 김 용 성†††

요 약

최근에 이 기종 시스템들간의 구조적인 문서 교환을 위해 XML(eXtensible Markup Language) 문서가 증가함에 따라 이를 모델링하기 위한 객체지향적인 시각화 도구가 필요하다. 본 논문은 UML(Unified Modeling Language) 클래스 다이어그램(Class Diagram)을 이용하여 XML 문서 구조 다이어그래밍(Structure Diagramming)을 위한 규칙과 알고리즘을 제안한다. XML 문서 구조 다이어그래밍은 복잡한 링크 관계 등을 쉽게 파악 할 수 있으므로 XML 문서 개발자들이 편리하게 문서 생성을 하도록 해주며, 객체 모델링이기 때문에 XML 문서를 저장한 데이터베이스들간의 스키마 차이를 최소화시킨다. 그리고, 본 논문에서는 XML 링크에 대한 형식 모델과 모델링 함수도 제안하여 객체지향 문서 처리의 효율적인 환경을 제공하고자 한다.

Structure Diagramming for XML documents using UML Class Diagram

Won-Seok Chae† · Yan Ha†† · Yong-Sung Kim†††

ABSTRACT

XML documents which are used for exchanging structured documents between heterogeneous distributed systems are increasing recently. It needs an object-oriented visualization tool for XML documents. So, we propose rules and an algorithm to represent structure of XML documents using UML Class Diagram. It helps to generate XML documents which are included links by understanding easily constructs of them and reduce gap of schema for them between heterogeneous databases. We propose formal models and modeling functions of XML links which provide an efficient environment for processing object-oriented documents.

1. 서 론

최근에 전자도서관이나 전자상거래 등의 분야에서 문서를 교환하기 위해 SGML(Standard Generalized Mark-up Language) 문서에 이어 XML 문서에 관한 관심이 집중되고 있다. 이 기종 시스템에서 교환할 수 있는 XML 문서는 SGML 문서를 축약시켜 놓은 형태에 링크와 포인터를 추가한 개념이다[1]. 이러한 XML 문서를 체계적으로 관리하기 위해서는 객체지향 혹은 객체-관계형 데이터베이스에 저장할 필요성이 있다. 그러나, 객체 모델과 XML 문서의 문법이 일치하지 않기 때문에 같은 XML 문서라도 데이터베이스의 종류와 설계자의 해석에 따라 다른 스키마 형태를 제시하게 된다. 따라서, 데이터베이스들간의 XML 문서의 이식과 교환이 쉽지 않다. 이에 본 논문은 객체 기술을 연구하는

† 종신회원 : 원광보건대학 컴퓨터응용개발과 교수
†† 준 회 원 : 전북대학교 전산통계학과 박사과정
††† 종신회원 : 전북대학교 컴퓨터과학과 교수
논문접수 : 1999년 4월 14일, 심사완료 : 1999년 9월 15일

OMG(Object Management Group)의 공식 승인을 받아 객체지향 방법론의 표준으로 자리 잡아가고 있는 UML 클래스 다이어그램의 다양한 메카니즘(Mechanisms)과 표기법(Notations)[2]을 이용하여 XML 문서를 모델링 하고자 한다. 본 연구에서 제안하는 XML 문서 구조 다이어그래밍은 링크가 포함된 복잡한 문서의 구조를 보기 쉽고 이해하기 편리하게 시각화할 뿐만 아니라 객체 모델링이므로 객체지향 데이터베이스 스키마로 변환이 용이하다.

본 논문의 구성은 2장에서 관련 연구에 대해 살펴보고 3장에서는 XML 문서(특히 링크 부분)와 UML 클래스 다이어그램에 대해 알아보고 4장에서는 UML 클래스 다이어그램을 이용하여 XML 문서 구조 다이어그램을 생성하기 위한 사상 규칙, 그리고, XML 링크에 대한 형식 모델과 모델링 함수를 정의한다. 또한, 사상 규칙에 의해 XML 문서 구조 다이어그램을 생성하는 알고리즘 및 적용한 결과를 제시한다. 5장에서는 결론 및 향후 연구 과제에 대해서 논의한다.

2. 관련연구

SGML DTD를 모델링하는 연구는 구조도, XOMT (eXtended Object Modeling Technique) 다이어그램, 그리고, UML 클래스 다이어그램을 이용한 방법이 있다. 구조도는 트리 구조와 유사한 형태로 DTD를 시각화하는데, 대부분의 SGML/XML 관련 파서(parser), 브라우저(browser), 개발 도구(development tool)들이 이를 이용하고 있다. 그러나, 이것은 엔티티나 애트리뷰트 리스트를 갖는 엘리먼트와 공유되는 부분 등을 표현하기가 어렵다[3]. XOMT 다이어그램은 객체지향적으로 DTD를 시각화하는 도구로 OMT를 이용하였으며, OMT로 표현이 되지 않는 DTD 문법을 위해 확장된 표기법을 제안하고 있다[4]. 그러나, XOMT에서는 발생지시자가 없는 '( )'를 표현할 수 없으며, 문자데이터('#PCDATA')나 애트리뷰트를 정의하는 엔티티에 대해 불명확한 클래스가 발생하며, 연결자 '|'로 연결된 하위 엘리먼트들이 하나 이상 발생할 경우 일반화 관계를 사용하는데, 이것은 객체 지향의 상속 개념에 어긋나게 된다. 이에 비해 UML 클래스 다이어그램은 별도의 확장된 표기법 없이 스테레오타입, 제한 조건 등 UML의 메카니즘을 이용하여 DTD를 시각화할 뿐만 아니라 객체지향 개념을 적절하게 사용하여 데이터베이스 스키마로의 변환을 용이하게 한다[5]. 그러나, SGML DTD에 맞게 규칙이 정의되어 있기 때문에 SGML과 다른 XML의 문법이나 링크 부분을 표현할 수 없다.

링크 구조를 갖는 HyTime 문서의 모델링에 관한 연구로는 [6][7]이 있다. [6]은 HOMT 다이어그램은 HyTime DTD 설계를 위해 XOMT 다이어그램[5]을 확장하여 노테이션과 노테이션 속성을 지원하는 부분과 하위 엘리먼트에 상위 엘리먼트가 존재하는 자기 참조 기법을 추가하였다. [7]은 HyTime 문서의 6가지 측면에서 다이아그래밍 표기를 제안하고 있는데 각각 SGML 파싱된 문서, SGML DTD, HyTime 메타- DTD, HyTime 문서 구조, HyTime 정의된 문서 구조, 메타-HyTime 구조이다. 그러나, 이들 역시 HyTime 문서의 속성을 표현하기 위해 개발되었기 때문에 확장 링크 등 XML 문서의 고유한 부분에 대해서는 처리가 불가능하다.

따라서, 본 논문은 SGML DTD를 UML 클래스 다이어그램에 사상시킨 정의를 확장하여 XML 문서에 맞는 사상 규칙과 알고리즘을 제안한다. 즉, SGML 과 XML 문서간의 문법의 차이점을 기술하고 이에 대해 사상 규칙을 재정의한다. 그리고, XML 문서의 고유한 특성인 링크 구조에 대해서는 사상 규칙은 물론 형식 모델과 클래스 다이어그램 모델링 함수도 제안한다.

3. XML문서와 UML 클래스 다이어그램

본 장에서는 XML 문서, 특히 링크에 대해 알아보고 XML 문서를 UML 클래스 다이어그램으로 모델링하기 위해 필요한 UML의 구성 요소를 살펴본다.

3.1 XML 문서

XML은 SGML의 문법에서 잘 사용하지 않는 복잡한 것은 제거하고 꼭 필요한 것들만 포함하고 있다. 예를 들면, SGML의 내용 모델에서 포함(inclusion)과 제외(exclusion)를 지원하지 않으며 연결자로서 '&'를 지원하지 않는다[1]. 또한, 시작 태그는 물론 종료 태그 역시 반드시 있어야 하므로 태그 생략 표시를 지원하지 않는다.

그리고, XML은 내장된 링크 기능을 지원하는데 일반적으로 링크는 2개 이상의 데이터 객체(Data Object)나 데이터 객체의 부분들 사이에 존재하는 관계를 말한다[8].

XML 링크는 단순 링크 뿐 만 아니라 확장 링크를 지원하는데, 이를 구별하기 위해 엘리먼트를 선언할

때 'XML : LINK' 애트리뷰트를 선언하여 링크 기능을 정의한다. 'XML : LINK'에 들어갈 수 있는 링크 유형은 'SIMPLE', 'EXTENDED', 'LOCATOR', 'GROUP', 'DOCUMENT'가 있다[9]. 'SIMPLE'은 HTML의 <A> 태그와 유사한 기능을 갖는 것으로 2개 자원 사이의 링크를 나타낸다. 이에 대한 DTD는 (그림 1)과 같다.

```
<!ELEMENT SIMPLE ANY>
<!ATTLIST SIMPLE
        XML:LINK CDATA #FIXED "SIMPLE"
        ROLE CDATA #IMPLIED
        HREF CDATA #REQUIRED
        TITLE CDATA #IMPLIED
        INLINE (TRUE|FALSE) "TRUE"
        CONTENT-ROLE CDATA #IMPLIED
        CONTENT-TITLE CDATA #IMPLIED
        SHOW (EMBED|REPLACE|NEW)
        "REPLACE"
        ACTUATE (AUTO|USER) "USER"
        BEHAVIOR CDATA #IMPLIED
>
```

(그림 1) 단순 링크의 DTD

'EXTENDED'와 'LOCATOR'는 여러 개의 자원에 대한 링크를 지정하는 기능을 갖는다. 'EXTENDED'는 확장된 링크를 의미하고, 'LOCATOR'는 실제적인 목표 자원을 가리킨다. 이에 대한 DTD는 (그림 2)와 같다.

'GROUP'과 'DOCUMENT'는 여러 관련된 인스턴스들을 그룹화하는 기능을 지원할 때 선언한다. 'GROUP'은 확장된 링크 그룹이 하나 혹은 그 이상의 'DOCUMENT' 요소가 포함된 것을 의미하고. 'DOCUMENT'는 확장된 링크 문서를 가리킨다.

```
<!ELEMENT EXTENDED ANY>
<!ELEMENT LOCATOR ANY>
<!ATTLIST EXTENED
        XML:LINK CDATA #FIXED
        "EXTENDED"
        ROLE CDATA #IMPLIED
        TITLE CDATA #IMPLIED
        INLINE (TRUE|FALSE) "TRUE"
        CONTENT-ROLE CDATA #IMPLIED
        CONTENT-TITLE CDATA #IMPLED
        SHOW (EMBED|REPLACE|NEW)
        "REPLACE"
        ACTUATE (AUTO|USER) "USER"
        BEHAVIOR CDATA #IMPLIED
>
<!ATTLIST LOCATOR
        XML:LINK CDATA #FIXED
        "LOCATOR"
        ROLE CDATA #FIXED "LOCATOR"
        HREF CDATA #IMPLIED
        TITLE CDATA #IMPLIED
        SHOW (EMBED|REPLACE|NEW)
        "REPLACE"
        ACTUATE (AUTO|USER) "USER"
        BEHAVOIR CDATA #IMPLED
>
```

(그림 2) 확장 링크의 DTD

링크 수행(Link Behavior)은 링크 엘리먼트의 저작자가 링크가 클릭될 때 타이밍 효과를 정의하는 것으로 'SHOW'와 'ACTUATE'라는 애트리뷰트로 선언한다. 'SHOW'는 링크된 자원으로 넘어갈 때 내용을 어떻게 보여줄 것인가를 정의하는데, 이에 대한 값과 의미는 <표 1>과 같다.

<표 1> 'SHOW'의 값과 의미

값	의 미
EMBED	문서가 어떤 자원을 가리키는 포인터를 가지고 있으면 문서가 보여지거나 혹은 처리될 때 자원이 문서 안으로 포함된다.
REPLACE	링크가 가리키고 있는 자원이 링킹 엘리먼트를 교체할 수 있다.
NEW	HTML의 링크와 같이 새로운 페이지를 이전 페이지가 보여줬던 위치에 보인다.

'ACTUATE'는 링크에 대한 수행이 발생하는 때에 관련된 정의를 하는 것으로 이에 대한 값과 의미는 <표 2>와 같다.

<표 2> 'ACTUATE'의 값과 의미

값	의 미
AUTO	링크를 만났을 때 자동으로 링크된 자원으로 넘어간다.
USER	사용자가 클릭했을 때 링크된 자원으로 넘어 간다.

3.2 UML 클래스 다이어그램

UML 클래스 다이어그램은 정적 관계로 연결된 클래스들과 그 구성요소를 다이어그램으로 표현한 것이다.

XML 문서의 논리적 구조를 모델링하기 위해 필요한 UML 클래스 다이어그램의 요소들은 다음과 같다[10].

3.2.1 클래스(Class)와 애트리뷰트(Attribute)

클래스는 클래스 이름, 애트리뷰트 리스트, 오퍼레이션 리스트로 구성된다. 애트리뷰트는 공적(Public),사적(Private)으로 구별된다.

3.2.2 상속성(Inheritance)과 집단화(Aggregation)관계

클래스들간의 관계는 크게 상속성과 집단화 관계로

나눌 수 있다. 상속성 관계는 상위클래스와 하위클래스 간의 'is-a(일반화)' 관계를 갖는 것으로 '△'로 표시한다. 집단화 관계는 상위클래스와 하위클래스 간의 'part-of'로서 '◇'로 표기한다.

3.2.3 다중성(Multiplicity)

다중성은 집단화 관계에 함께 표시되는 것으로 상위클래스를 구성하는 하위클래스들의 발생 횟수를 나타낸다. 표기는 '최저 발생 횟수'..'최고 발생 횟수'로 하는데, 종류는 '0..1', '0..*', '1..*'가 있다.

3.2.4 스테레오타입(Stereotype)

스테레오타입은 기존의 모델링 요소와 같은 형태(애트리뷰트, 연관성)를 가지나 다른 취지로 사용할 서브클래스로서 기본 클래스에 부가적인 제한조건을 추가한 것이다. 표기는 '«' 와 '»' 사이에 키워드 스트링을 쓴다.

3.2.5 의존(Dependency) 관계

의존 관계는 2개 이상의 모델 요소 사이의 의미적인 관계를 나타내는 것으로 표기는 하나의 모델 요소에서 다른 모델 요소로의 파선을 긋는다. 필요에 따라 임의의 스테레오타입과 임의의 이름으로 라벨을 붙인다.

4. XML 문서 구조 다이어그래밍

본 장에서는 XML 문서 구조 다이어그래밍을 생성하기 위해 XML DTD와 문서 인스턴스로부터 UML 클래스 다이어그램으로 사상시키는 규칙과 알고리즘을 제안하고, 이를 XML 문서에 적용시킨 결과를 제시한다.

4.1 사상 규칙

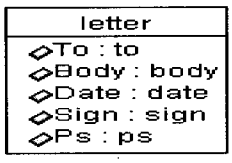
XML DTD와 문서 인스턴스로부터 UML 클래스 다이어그램을 생성하기 위해 다음과 같은 정의와 규칙들을 기술한다.

[정의 1] XML 문서의 요소들을 UML 클래스 다이어그램의 집합으로 사상한다.

엘리먼트(혹은 엔티티) 선언에서 선언 부분에 오는 엘리먼트나 엔티티는 내용 부분에 오는 엘리먼트나 엔티티와 구별된다. 엘리먼트(혹은 엔티티) 선언에 관련된 규칙은 다음과 같다.

[규칙 1] 엘리먼트(혹은 엔티티) 선언에서 선언 부분에 오는 엘리먼트(혹은 엔티티)는 클래스가 되고, 내용 부분에 오는 엘리먼트나 엔티티는 해당 클래스의 공적 애트리뷰트 타입이 된다. 단, 애트리뷰트를 정의하는 엔티티 선언의 경우 클래스를 만들지 않고 엔티티 확장을 한다.

예 1) <!ELEMENT letter (to,body,date,sign,ps?)>



(그림 3) [규칙 1]의 클래스 다이어그램

예 2) <!ENTITY %a.rid "rid IDREF
#REQUIRED" -- ID 애트리뷰트 정의 -->
<!ATTLIST name %a.rid; >
⇒ <!ATTLIST name rid IDREF #REQUIRED>

그리고, 엘리먼트(혹은 엔티티) 선언에서 선언 부분에 연결자 '|'가 오는 경우 엘리먼트와 엔티티 개수만큼 클래스를 생성한다.

다음은 [규칙 1]의 공적 애트리뷰트와 구별되는 사적 애트리뷰트에 관한 규칙이다.

[규칙 2] 애트리뷰트 리스트의 애트리뷰트는 엘리먼트 클래스의 사적 애트리뷰트 타입이 된다. 단, 애트리뷰트의 키워드가 'ID'인 경우는 공적 애트리뷰트가 된다.

예) <!ATTLIST date type "(1|2|3|4) #IMPLIED">

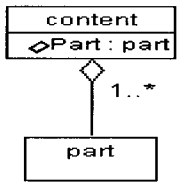
(그림 4) [규칙 2]의 클래스 다이어그램

엘리먼트나 엔티티 내용 부분에서 발생 지시자는

'*', '+', '?'가 있으며, 각각 0번 이상, 1번 이상, 0 혹은 1번 발생됨을 의미한다. 발생 지시자가 없는 경우 디폴트값은 1이다. 이에 대한 규칙은 다음과 같다.

[규칙 3] 엘리먼트(혹은 엔티티) 내용 부분에서 발생 지시자('*', '+', '?')는 다중성('0..*', '1..*', '0..1')으로 표시한다.

예) <!ELEMENT content (part+) >

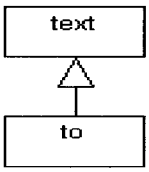


(그림 5) [규칙 3]의 클래스 다이어그램

엘리먼트나 엔티티 내용에서 '#PCDATA'가 하위 엘리먼트와 혼합된(mixed) 경우와 그렇지 않은 경우를 구별하여 다음과 같이 정의한다.

[규칙 4] 엘리먼트(혹은 엔티티) 내용 부분에 단일한 문자데이터('#PCDATA')가 오면 엘리먼트(혹은 엔티티) 클래스는 기본 데이터클래스(text)로부터 상속을 받는다.

예) <!ELEMENT to (#PCDATA) >

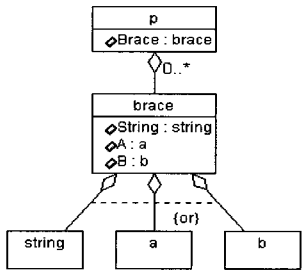


(그림 6) [규칙 4]의 클래스 다이어그램

[규칙 5] 엘리먼트(혹은 엔티티) 내용 부분에서 문자데이터(#PCDATA)가 다른 하위엘리먼트와 혼합된 경우 임의의 클래스(string)를 생성한다. 클래스(string)는 기본 데이터클래스(text)로부터 상속받는다.

[규칙 6] 엘리먼트(혹은 엔티티) 내용 부분에 '( )'가 있고 발생 지시자가 있거나 안과 밖의 연결자가 다를 경우 임의의(brace) 클래스를 생성한다. 상위클래스와 (brace) 클래스는 집단화 관계를 갖는다.

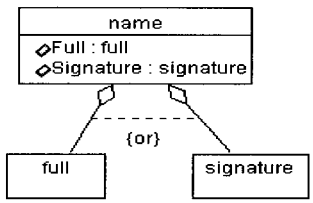
예) <!ELEMENT p (#PCDATA|a|b)*>



(그림 7) [규칙 5][규칙 6]의 클래스 다이어그램

[규칙 7] 엘리먼트(혹은 엔티티) 내용 부분에서 연결자가 '|'인 경우 엘리먼트 클래스와 하위클래스들간의 집단화 관계이고, 이를 연결하여 '{or}'라는 제한 조건을 표시한다.

예) <!ELEMENT name (full|signature)>

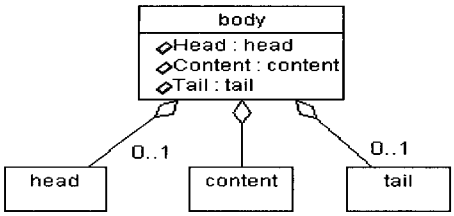


(그림 8) [규칙 7]의 클래스 다이어그램

엘리먼트나 엔티티 내용 부분에서 '&'는 지원하지 않으므로 ','의 경우 '{ordered}'라는 제한 조건 없이 하위 클래스들이 항상 순서의 의미를 갖도록 한다.

[규칙 9] 엘리먼트(혹은 엔티티) 내용 부분에서 연결자가 ','인 경우 엘리먼트(혹은 엔티티) 클래스와 하위클래스들간의 집단화 관계이다.

예) <!ELEMENT body (head?,content,tail?)>



(그림 10) [규칙 9]의 클래스 다이어그램

XML 문서에서 다른 자원(URL 등)을 가리키는 링크 구조를 포함하는 경우는 UML 클래스 다이어그램의 의존 관계와 스테레오타입을 이용하여 표기한다.

[규칙 10] 단순 링크에서 참조될 다른 자원을 가리킬 때 의존 관계를 갖고 스테레오타입으로 '«HREF»'라는 키워드를 쓴다.

(그림 11)과 (그림 12)는 단순 링크의 DTD와 문서 인스턴스에 대한 예이다.

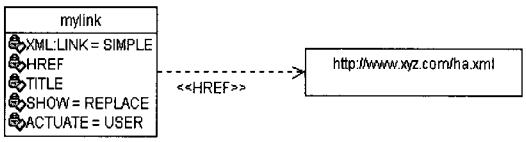
```
<!ELEMENT mylink (#PCDATA)>
<!ATTLIST mylink
        XML:LINK  CDATA              #FIXED "SIMPLE"
        HREF      CDATA              #REQUIRED
        TITLE     CDATA              #IMPLIED
        SHOW      (EMBED|REPLACE|NEW) "REPLACE"
        ACTUATE   (AUTO|USER)         "USER"
>
```

(그림 11) 단순 링크의 DTD

```
<mylink XML:LINK="SIMPLE" TITLE="Reference"
  HREF="http://www.xyz.com/ha.xml" SHOW="new"
  ACTUATE="USER"> written by Ha(1999) </mylink>
```

(그림 12) 단순 링크의 문서 인스턴스

(그림 13)은 (그림 11)과 (그림 12)에 대한 클래스 다이어그램을 나타낸 것이다.



(그림 13) 단순 링크의 클래스 다이어그램

확장 링크의 'LOCATOR'에 오는 'HREF' 역시 [규칙 11]과 마찬가지로 의존 관계에 '«HREF»' 라는 키워드를 사용한다.

[규칙 11] 확장 링크에서 참조될 다른 자원을 가리킬 때 의존 관계를 갖고 스테레오타입으로 '«LOCATOR»' 라는 키워드를 쓴다.

(그림 14)와 (그림 15)는 각각 확장 링크의 DTD와 문서 인스턴스의 예이다.

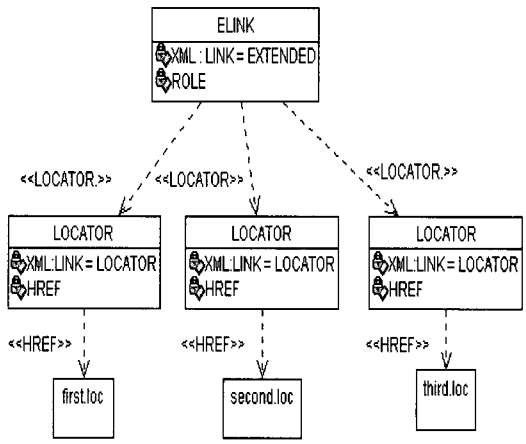
```
<!ELEMENT ELINK ANY>
<!ELEMNT LOCATOR ANY>
<!ATTLIST ELINK
        XML:LINK  CDATA     #FIXED "EXTENED"
        ROLE      CDATA     #IMPLIED
>
<!ATTLIST LOCATOR
        XML:LINK  CDATA     #FIXED "LOCATOR"
        HREF      CDATA     #REQUIRED
>
```

(그림 14) 확장 링크의 DTD

```
<ELINK XML:LINK="EXTENDED" ROLE="ANNOTATION">
<LOCATOR XML:LINK="LOCATOR" HREF="first.loc"> The
 The first text </LOCATOR>
<LOCATOR XML:LINK="LOCATOR" HREF="second.loc">
 The second text </LOCATOR>
<LOCATOR XML:LINK ="LOCATOR" HREF="third.loc">
 The third text </LOCATOR>
</ELINK>
```

(그림 15) 확장 링크의 문서 인스턴스

(그림 16)은 (그림 14)와 (그림 15)에 대한 클래스 다이어그램을 나타낸다.



(그림 16) 확장 링크의 문서 인스턴스

4.2 XML 링크의 형식 모델

본 절에서는 XML 문서 구조를 모델링한 UML 클래스 다이어그램에서 XML 링크 부분을 객체지향 스키마로 표현하기 위한 형식 모델을 정의하는데 이를

위해 몇 가지 가정들이 필요하다.

[가정 1] 기본 타입(integer, text 등)의 집합은 dom이고, 애트리뷰트의 집합 $A = \{ a_1, a_2, ..., a_n \}$, 객체의 집합 $O = \{ o_1, o_2, ..., o_n \}$, 클래스의 집합 $C = \{ c_1, c_2, ..., c_n \}$ 로 표기한다.

[가정 2] 객체 집합 O의 값들은 $V = \{ v_1, ..., v_i, ..., v_n \}$으로 표기한다. 단, V의 원소 v_i 는 *nil* 이거나 dom 혹은 O의 각 원소들이 되고, 튜플(tuple) $[a_1 : v_1, ..., a_n : v_k]$, 집합(set) $\{v_1, ..., v_k \}$ 그리고, 리스트(list) $[v_1, ..., v_k]$도 V의 원소이다.

[가정 3] 클래스 타입의 집합은 τ이고, 타입 τ의 언어(interpretation)는 $dom(\tau)$이다.

위와 같은 가정들을 이용하여 XML 링크에 대한 형식 모델을 정의하면 다음과 같다.

[정의 2] 튜플 $[a : \tau]$에서 a가 'HREF' 이면 링크 타입이다. 즉, $dom([a : \tau]) = \{ [a : v] \mid v \in dom(\tau) \}$ 이다

예) 튜플 [HREF : http://www.xyz.com/ha.xml]은 링크 타입이 되고 링크된 자원은 클래스의 타입이 된다.

[정의 3] 튜플 $[a : \tau]$에서 a가 'XML:LINK = EXTENDED' 이면 확장 링크 타입이다. 즉, $dom([a : \tau]) = \{ [a_1 : v_1, ..., a_n : v_n] \mid v_i \in dom(\tau_i),\ i = 1,...,n$; 단, n은 로케이터 개수 $\}$ 이다.

예) 튜플 [XML:LINK : EXTENDED]은 확장 링크 타입으로 'LOCATOR' 개수가 3일 때, $dom([$ XML:LINK : EXTENDED $])$ = { [$XML{:}LINK_1$: $LOCATOR_1$, ..., $XML{:}LINK_3$: $LOCATOR_3$] | $v_i \in dom(\tau_i),\ i = 1,...,3$ } 이다.

4.3 XML 링크의 모델링 함수

모델링 함수는 UML 클래스 다이어그램의 구조를 비교하여 통합 등을 하기 위한 것으로 XML 링크에 관련된 함수들을 정의하는데, 이를 위해 다음과 같은 가정이 필요하다.

[가정 4] 클래스 다이어그램은 (S, p)의 쌍으로 이루어진다. 단, S는 집합이고, 함수 $p : S \rightarrow S : \forall t \in S$ 이다.

S의 클래스 t는 2가지 특성을 만족한다[10].

① p(t)=t, t는 S의 클래스로서 클래스 다이어그램의 루트(root)가 된다.

② $\forall x \in S,\ \exists k \in N,\ p^k(x) = t$.

[정의 4] 함수 $h : \tau \rightarrow P(t) : \forall t \in \tau$, 링크된 자원을 나타낸다.

예) <!ATTLIST letterlinks
XML:link CDATA #FIXED "document"
HREF CDATA #FIXED "lett_links.xml">
h(letterlinks) = lett_links.xml

[정의 5] 함수 $n : \tau \rightarrow$ Integer : $\forall t \in \tau$, 확장된 링크에 연결된 LOCATOR 개수를 나타낸다.

예) <ELINK>
<LOCATOR ROLE="broad"
HREF="movie.xml"> </LOCATOR>
<LOCATOR ROLE="narrow"
HREF="animation.xml"> </LOCATOR>
n(ELINK) = 2

4.4 알고리즘

XML DTD와 문서 인스턴스를 입력받아 XML 문서 구조 다이어그램을 생성하는 알고리즘은 다음과 같다.

```
입력 : XML DTD, 문서 인스턴스
출력 : UML 클래스 다이어그램

begin
{
 switch ( 선언된 DTD 요소 )
 {
 case : entity or element
 {
   if ( 애트리뷰트 정의하는 엔티티 선언 )
      then entity_extension() // 엔티티 내용 대치
   else
   {
    make_class() // 클래스 생성
    while ( content_empty() )
    {
      if ( '( )' 안과 밖의 연결자가 다를 경우 )
      { // 엘리먼트 내용에 괄호가 있는 경우
         make_brace_class() // brace 클래스 생성
         make_attlist_function() // 애트리뷰트 추가
         make_aggregation() // 집단화 관계
         make_subclass() // 하위클래스 생성
         occurrence_function() // 발생 지시자 처리
      }
      if ( connector )
       { // 엘리먼트 내용에 연결자가 있는 경우
```

```
            make_attlist_function()
            make_subclass()
            for( 하위클래스 수 ) {
                make_aggregation()
                occurrence_function()
                if ( connector == '|' )
                    make_constraint_or()
            } }
        else // 연결자없이 '#PCDATA'가 오는 경우
             make_inherit_from()
             // 기본 타입으로부터 상속
    } } }
case : attlist
{
    make_attlist_function()
    // 엘리먼트 클래스에 애트리뷰트와 타입 첨가
    if ( attribute_name == 'HREF' )
        then {
                make_class()
                make_dependency()
                make_href_stereotype()
                // «href» 스테레오타입
                }
    if( attribute_name ==
        'XML:LINK = "EXTENDED" ' )
     // 확장 링크인 경우
      then
       {
           for ( LOCATOR 개수 ) {
              make_class() // locator 클래스 생성
              make_attlist_function()
              // 애트리뷰트 첨가
              make_dependency() // 의존 관계
              make_locator_stereotype()
              // «locator» 스테레오타입
              make_class()
              make_dependency()
              make_href_sterotype()
           } } } }
end;
```

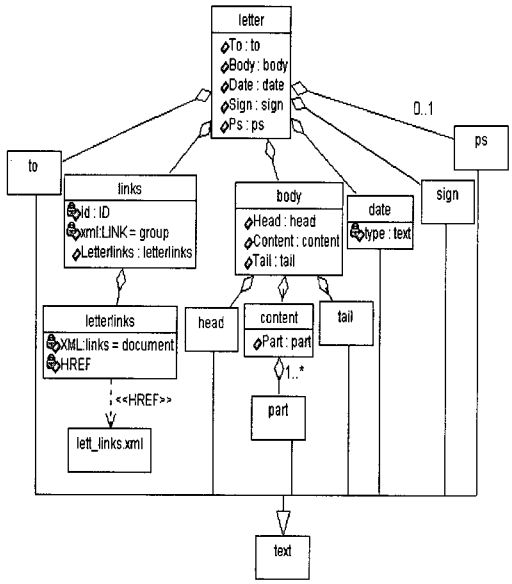
4.5 적용 결과

간단한 편지 양식의 XML DTD는 (그림 17)과 같다.

```
<?XML version = "1.0"? >
<!DOCTYPE  letter  [
<!ELEMENT  letter     (to, links, body, date, sign, ps?) >
<!ELEMENT  links      (letterlinks)                      >
<!ATTLIST  links      id  ID  #IMPLIED                   >
           xml:LINK CDATA #FIXED "group"                 >
<!ELEMENT letterlinks  EMPTY                            >
<!ATTLIST letterlinks XML:link CDATA #FIXED "document"
                      HREF CDATA #FIXED "lett_links.xml">
<!ELEMENT  to         (#PCDATA)                         >
<!ELEMENT  body       (head?, content, tail?)           >
<!ELEMENT  head       (#PCDATA)                         >
<!ELEMENT  content    (part+)                           >
<!ELEMENT  part       (#PCDATA)                         >
<!ELEMENT  tail       (#PCDATA)                         >
<!ELEMENT  date       (#PCDATA)                         >
<!ELEMENT  sign       (#PCDATA)                         >
<!ELEMENT  ps         (#PCDATA)                         >
]>
```

(그림 17) '편지' 의 XML DTD

(그림 17)의 DTD를 본 연구에서 제안한 사상 알고리즘에 적용시킨 결과는 (그림 18)이다. 즉, (그림 18)은 UML 클래스 다이어그램 형태의 XML 문서 구조 다이어그램이다.



(그림 18) '편지' 의 클래스 다이어그램

(그림 18)의 XML 문서 구조 다이어그램과 SGML DTD 클래스 다이어그램[5]을 비교해보면 태그 생략 표시와 내용 모델에서 예외(포함과 제외) 처리에 대한 표기법이 존재하지 않는다. 그리고, 연결자 '&'는 존재하지 않으므로 ','와 '&'를 구별하기 위한 '{ordered}' 라는 제한 조건은 사용하지 않는다. 즉, ','인 경우 디폴트로 순서의 의미를 갖는다.

또한, XML 문서의 특징인 단순 링크와 확장 링크를 표현하기 위해 UML 클래스 다이어그램의 의존 관계와 스테레오타입을 이용하였다.

5. 결론 및 향후 연구 과제

본 연구에서는 XML 문서를 모델링하기 위해 UML 클래스 다이어그램을 이용한 XML 문서 구조 다이어그래밍을 제안하였다. XML DTD와 SGML DTD의 차이점을 파악하여 SGML DTD에 대해 사상시킨 UML 클래스 다이어그램의 요소를 XML DTD에 맞게 다시 사상시키고 링크 구조에 대해 사상 규칙과 형식 모델, 그리고 클래스 다이어그램 모델링 함수를 확장하여 제안하였다.

본 연구의 XML 문서 구조 다이어그램은 복잡한 XML 문서를 쉽게 파악할 수 있도록 시각화하였으므로 XML 문서 개발자들이 이를 이용하여 문서 구조를 파악, 개발을 용이하게 해준다. 또한, 데이터베이스 설계자들은 일일이 XML 문서의 시맨틱을 파악할 필요없이 UML 클래스 다이어그램 형태의 모델링을 데이터베이스 스키마로 변환할 수 있으므로 데이터베이스로 저장이 용이하며, 동일한 객체 모델링으로 이 기종의 데이터베이스들간의 문서 교환과 이식이 용이하다. 그리고, XML 링크에 대한 형식 모델과 모델링 함수는 객체지향 데이터베이스 문서 처리를 위한 기반이 될 것이다.

향후 연구 과제는 본 연구의 UML 클래스 다이어그램을 다시 XML 문서로 역변환하는 알고리즘을 개발하여 본 연구에서 제안한 알고리즘을 검증하는 것과 객체 모델링된 XML 문서에 대한 객체지향 데이터베이스 스키마를 자동 생성하여 실제 데이터베이스에 저장하고 질의, 검색하는 것이다.

참 고 문 헌

[1] 이강찬, 이원석, "XML(eXtensible Markup Language)," http://kookmin.ac.kr/~yoon/xml.html

[2] Rational Software, "UML Semantics version 1.1," September 1997, http://www.rational.com/uml

[3] Eric van Herwijnen, "Practical SGML," Kluwer Academic Publishers, 1994.

[4] 박인호, 한에노, 정은주, 김은정, 배종민, 강현석, 김완석, "XOMT : SGML DTD 설계를 위한 객체 다이아그래밍 기법", 한국정보과학회 논문지(C), 제3권, 제3호, pp.228-237, 1997. 6.

[5] 하얀, 황용주, 김용성, "SGML DTD로부터 UML 클래스 다이어그램으로의 사상 알고리즘", 한국정보과학회 논문지(B), 제26권, 제4호, pp.508-520, 1999. 4.

[6] 장원호, 임혜정, 박인호, 강현석, "HOMT : HyTime DTD 설계를 지원하기 위한 XOMT의 확장", 한국정보처리학회 논문지, 제5권, 제9호, pp.2213-2223, 1998. 9.

[7] Lloyd Rutledge, John F. Buford, John L. Rutledge, "Modeling Techniques for HyTime," 1996, http://www.sil.org/sgml/rutledgeModelingHy.html

[8] Richard Light, 채규혁 역, "차세대 웹의 혁명 XML," 도서출판 대림, 1998.

[9] 이규철, "차세대 인터넷 전자문서 XML : 표준, 응용, 도구 및 연구과제," 한국정보과학회 춘계학술발표회 특강 요약집, pp.106-128, 1998. 4.

[10] James Rumbaugh, Ivar Jacobson, Grady Booch, "The Unified Modeling Language Reference Manual," Addison-Wesley, 1999.

[11] E. Akpotsui, V. Quint, C. Roisin. "Type Modelling for Document Transformation in Structured Edition Systems," August 1994, http://www.oasis-open.org/cover/

[12] Craig Larman, "Applying UML and PATTERNS : An Introduction to Object-Oriented Analysis and Design," Prentice-Hall, 1998.

[13] V. Christophides, S. Abiteboul, S. Cluet, M. Scholl, "From Structured Documents to Novel Query Facilities," Proceedings of ACM SIGMOD International Conference, Management of Data, pp.313-324, May 1994.

[14] Yan Ha, Yong-Joo Hwang, Yong-Sung Kim, "SGML DTD Modelling Using UML Class Diagram," Proceedings of ICT'99, pp.206-210, June 1999.

[15] Natanya Pitts-Moultis, Cheryl Kirk, "XML black book," The Coriolis Group, Inc., 1999.

채 원 석

e-mail : wschae@sky.wkhc.ac.kr

1984년 원광대학교 전자공학(공학사)

1986년 인하대학교 전자계산학과(이학석사)

1998년 전북대학교 전산통계학과 박사과정 수료

1989년~현재 원광보건대학 컴퓨터응용개발과 부교수

관심분야 : 멀티미디어 응용, 영상 처리, 전자 도서관, 정보 검색 등

하 얀

e-mail : yanha@cs.chonbuk.ac.kr

1992년 덕성여자대학교 전산학과 졸업(학사)

1994년 이화여자대학교 교육대학원 전자계산교육전공 졸업 (석사)

1996년~현재 전북대학교 전산통계학과 박사과정.

관심분야 : SGML/XML, 객체지향 모델링, 전자 도서관, 정보 검색, 지능형 교육 시스템 등

김 용 성

e-mail : yskim@moak.chonbuk.ac.kr

1978년 고려대학교 수학과 졸업 (이학사)

1984년 광운대학교 전산학과 (이학석사)

1992년 광운대학교 전산학과 (이학박사)

1985년~현재 전북대학교 컴퓨터학과 교수

1996년~1998년 한국학술진흥재단 전문위원

관심분야 : 디지털 도서관, 정보검색, 인터넷 기반 정보 검색, 멀티미디어 시스템, 인공지능 등