The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 15, No. 4, pp.231-237, Aug. 31, 2015. pISSN 2289-0238, eISSN 2289-0246

http://dx.doi.org/10.7236/JIIBC.2015.15.4.231
JIIBC 2015-4-30

CMXML: XML의 개념적 모델링 기법

CMXML: A Conceptual Modeling Methodology for XML

김영웅*

Young-Ung Kim*

요 약 XML은 다양한 언어들을 이용하여 문서의 구조나 형식을 논리적으로 정의하고 있지만, 각각의 언어들은 서로 다른 구조와 문법을 채택하고 있어 실세계의 데이터의 의미나 데이터 사이의 관계를 표현하는 개념적 모델의 도구로 사용하기 어렵다. 본 논문은 XML 스키마 문서를 개념적으로 모델링할 수 있는 기법인 CMXML을 제안한다. CMXML은 XML을 형식으로(formal)으로 정의하고, 형식에 의해 각 요소들을 그래픽으로 표현하는 방법을 제시하고, 본 모델의 타당성을 보여주기 위해 CMXML으로 모델링한 개념적 모델을 논리적 모델인 XML 스키마로 매핑하는 기법을 제시한다.

Abstract However XML languages can logically define the type of structure with their's own grammar, it is inadequate to use them as a tool for conceptual model that represents the semantics of data and the relationships between the data in the real world. In this paper, we propose conceptual modeling techniques, called CMXML, for modeling the XML schema at the conceptual level. For this purpose, we define the model formally, and provide a way to represent the model in a graphical and text form. We also propose an mapping methodology providing transformation from CMXML to XML schema to show the feasibility of the proposed model.

Key Words : XML schema, Conceptual model, Logical model, Mapping methodology

Ⅰ. 서 론

XML(eXtensible Markup Language)은 웹문서를 표현하고 교환하는데 사실상의(de facto) 표준으로 자리 잡고 있으며, DTD[1], XML schema[2], RELAX-NG[3] 등의 언어는 XML 문서의 구문이나 구조를 정의하는 형식언어로 널리 사용되고 있다. 그러나 이들 언어들은 XML 문서를 논리적 레벨에서 설계하는데 초점이 맞추어져 있어 실세계의 데이터의 의미나 데이터 간의 관계를 표현하는 개념적 모델로 사용하기에는 어렵다.

XML을 개념적 단계에서 설계하기 위해서는 XML 고유의 특성인 계층적 구조(hierarchical structure), 반구조적 구조(semi-structured)와 더불어 컨텐츠 모델(content model), 혼합모델(mixed model) 등을 표현하여야 한다. 이로 인해 기존의 개념적 설계 단계에서 사용되는 개체관계모델(Entity-Relationship Model), UML(Unified Modeling Language), ORM(Object Relational Mapping) 등은 XML 특성을 표현하기 위해 확장된 형태로 개념적 모델을 제시하고 있다.

본 논문은 XML을 개념적으로 모델링할 수 있는 기법인 CMXML을 제안한다. CMXML은 개념적 스키마를 구성하는 각 요소들을 정의하고, 형식에 의해 각 요소들

*정회원, 한성대학교 컴퓨터공학과
접수일자 2015년 6월 11일, 수정완료 2015년 7월 10일
게재확정일자 2015년 8월 7일

Received: 11 June, 2015 / Revised: 10 July, 2015 /
Accepted: 7 August, 2015
*Corresponding Author: yukim@hansung.ac.kr
Dept. of Computer Engineering, Hansung University, Korea

을 그래픽 및 텍스트로 표현하는 방법을 제시하고, 본 모델의 타당성을 보여주기 위해 CMXML으로 모델링한 개념적 모델을 논리적 모델인 XML 스키마 정의(XSD: XML Schema Definition)로 매핑하는 기법을 제시한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 XML 기반의 개념적 모델에 관한 관련 연구를 기술하고, 제 3장에서는 본 연구에서 제시하는 CMXML을 구성요소들을 정의하고 그래픽 및 텍스트 표현 기법을 기술한다. 제 4장에서는 제안한 CMXML로 모델링한 개념적 모델을 논리적 모델인 XML 스키마 정의로 매핑하는 기법을 제시하고. 끝으로 제 5장에서 결론 및 향후 연구과제에 대해 기술한다.

II. 관련 연구

XML의 개념적 모델 연구로는 기존의 평면적 구조를 갖는 ER 모델 또는 UML에 XML 특성을 확장하는 연구와 계층적(hierarchical) 구조를 갖는 모델 연구로 구분할 수 있다. 전자의 대표적 연구로는 Xer[4,5], EReX[6], PIM/PSM[7] 등이 있다.

Xer 모델은 개체관계 모델에서 표현할 수 없는 개념(sequence/all/choice, mixed type 등)들을 반영한 확장된 개체관계 모델을 제안하였다. 또한 복합요소가 복합요소를 포함할 경우는 이를 관계로 표현하였고 매핑대응수(mapping cardinality)도 표현하고 있다. 개체 사이의 컨텐츠 모델이 choice의 경우 이를 개체 내부에서 중첩된 형태로 표현하고 있다.

EReX 모델은 XML 특성을 표현하기 위해 개체관계 모델에 category, coverage constraints, order constraints 개념을 확장한 개념적 모델을 제안하고 있다. 또한, 이 기법에서는 XGrammar를 정의하여 XML 스키마를 명세하였으며, EReX 스키마를 XGrammar 스키마로 변환하는 알고리즘을 제시하고 있다.

PIM/PSM 모델은 기존의 UML을 확장하여 개념적 모델을 위한 PIM(Platform Independent Model)과 XML 스키마를 위한 PSM(Platform Specific Model)의 두 단계의 설계 모델을 제시하고, PIM을 PSM으로 변환하기 위한 매핑기법과 RTG(Regular Tree Grammar)를 이용하여 PSM을 XML 문서로 변환하는 기법도 제시하고 있다.

XML의 개념적 모델 연구로 계층적(hierarchical) 구조를 갖는 모델 연구의 대표적 연구로는 C_XML[8], ORA-SS[9], XSemantic[10] 등이 있다.

C_XML 모델은 객체, 관계, 제약조건 개념으로 XML 스키마를 계층적 구조로 표현하는 개념적 모델을 제시하고 있다. 이 기법에서는 XML 고유의 특성인 컨텐츠 모델(sequence/all/choice), 중첩된 계층구조, mixed content, use 뿐만 아니라 any, anyAttribute도 표현가능하다. 또한, 이 기법에서는 C-XML과 XML 스키마와의 상호 변환하는 기법도 제시하고 있다.

ORA-SS(Object-Relationship-Attribute model for Semi-Structured data) 모델은 객체, 관계, 속성의 세 가지 기본 개념을 이용하여 네 가지의 다이어그램(schema, instance, functional dependency, inheritance)으로 모델을 표현하며, 특히 이 모델의 특징은 참조 객체의 표현이 가능하며, n-ary(n>2) 관계를 표현할 수 있고, 객체에 속하는 속성과 관계에 속하는 속성을 구분하여 표현할 수 있는 장점이 있다.

XSemantic 모델은 semantic level과 schema level의 두 단계의 레벨로 표현하며, semantic level은 semantic network 개념을 이용하여 node, edge, label, constraint로 모델을 표현하며, schema level은 semantic level을 element/attribute 선언, simple/complex type 정의로 표현하고, 이 두 레벨 사이의 매핑 기법을 제시하고 있다.

각각의 모델들은 접근하는 방법에 따라 각자의 장·단점을 가지고 있다. Necasky 연구[11]에서는 XML을 위한 개념적 모델이 갖추어야할 요구조건들을 일반 요구조건 6가지와 XML 특성 요구조건 9가지로 제시하고 기존 11개의 개념적 모델들을 이 요구조건에 근거하여 충족여부를 비교한 결과를 보여주고 있다.

III. CMXML: 개념적 모델

이 장에서는 XML 스키마의 개념적 모델인 CMXML에 대해 기술한다. XML 스카마를 개념적으로 모델링하기 위해 CMXML에서 사용하는 각 개념들을 형식적으로 정의하고, 이 개념들을 다이어그램으로 표현하기 위한 심볼들을 기술하고, 다이어그램으로 표현하기 곤란한 세부정보들을 텍스트 형태의 표현에 대해 기술한다.

1. CMXML – 형식 정의

[정의 1] 스키마는 다음과 같은 4-튜플로 정의한다.

$S = (N, E, L, C)$ where,

N: 실세계의 객체를 표현하기 위한 노드들의 집합

E: 노드들의 관계를 표현하기 위한 방향성 에지(directed edge)들의 집합

L: 관계 타입을 표현하기 위한 라벨의 집합

C: 노드와 에지에 적용되는 제약조건들의 집합

노드는 비단말노드(non-terminal node: N_N)와 단말노드(terminal node: N_T)로 구분되며, 비단말노드는 다른 노드와 에지로 연결되며, 에지에는 라벨을 붙여 관계타입을 구분한다. 단말노드의 값은 원자 값(atomic value), 다중 값(multi-valued), 리스트 값(순서가 있는 값), 유니온 값(선택적 값)이 존재한다. 노드에 인스턴스가 존재하지 않고 단지, 타입만 존재하는 경우 이 노드를 추상화 노드(abstract node: N_A)라 한다.

[정의 2] 노드 $n \in N$은 다음과 같이 정의한다.

$n = (n_{name}, n_{value})$ and

$n \in N_T$ 일 때,

n_{value} ::= *원자 값* | *다중 값* | *리스트 값* | *유니온 값*

$n \in Nn$ 일 때,

n_{value} ::= {<*l*, [n_{name} | {n_{name}}]>}

n_{name}은 스키마 상에서 구분되는 노드명이며, n_{value}는 노드가 단말노드일 경우 노드에 부여되는 값을 표현하며, 비단말노드일 경우 노드는 노드명과 함께 자식노드들의 노드명과 자식노드들에게 적용되는 라벨의 집합으로 표현한다.

[정의 3] 두 노드 사이의 관계를 표현하는 에지 $e \in E$는 다음과 같이 정의한다.

$e = (l, n_s, n_d)$ where,

$l \in L$, $n_s \in N_N$, $n_d \in N_N \cup N_T$

n_s는 에지에서 나가는 진출(outgoing)노드이고, n_d는 에지에서 들어오는 진입(incoming)노드를 나타낸다. 이때, n_s를 부모노드라 하고 n_d를 n_s의 자식노드라 한다. 또한, 부모노드가 없는 노드를 전역노드(global node)라 한다.

[정의 4] 관계 타입을 구분하는 라벨 $l \in L$은 다음과 같이 정의한다.

l ::= *generalization* | *aggregation* | *association* | *typeof*

generalization ::= [*overlapping* | *disjoint*][*total* | *partial*]

관계 타입이 *typeof*일 경우 자식노드는 추상화 노드가 되며, 자식노드가 부모노드의 타입이 된다.

[정의 5] 에지그룹(edge group) $E_G \subseteq E$는 다음의 성질을 만족하는 에지들의 집합이다.

$E_G = \{e1(l1, n_s1, n_d1), ..., ek(lk, n_sk, n_dk)\}$ where. $\forall i \forall j (1 \leq i, j \leq k)$에 대해 $(li = lj, n_si = n_sj, n_di \neq n_dj)$

이때, $n_d1, .., n_dk$를 n_s에 대해 형제노드라 한다. k가 1일 경우 e를 단순에지(simple edge)라 한다.

[정의 6] 노드, 에지, 에지그룹에서 준수해야 할 제약조건 C는 다음과 같이 정의한다.

C ::= *Ccardinal* | *Cdomain* | *Ccontent*

Ccardinal ::= [0..1] | [1..1] | [0..N] | [M..N], where M, N is positive integer or unbounded

Cdomain ::= [Val v] | [MinVal v] | [MaxVal v] | [Len v] | [MinLen v] | [MaxLen v], where v is positive integer

Ccontent ::= *sequence* | *all* | *choice*

Ccardinal 은 에지에, *Cdomain*은 노드에, *Ccontent*는 에지그룹에 각각 적용되는 제약조건들이다.

2. CMXML – 다이어그램 표현

다이어그램은 기호, 선, 점 등을 사용해 각종 사상의 상호 관계나 과정, 구조 등의 의미를 빠르고 정확하게 전달하는 시각 언어로써 개념적 설계 단계에서 유용한 수단으로 사용되고 있다. 본 절에서는 앞 절에서 기술한 형식 정의를 다이어그램으로 표현하기 위한 표기법을 기술한다.

표 1은 CMXML에서 사용하는 다이어그램 심볼들을 보여준다. 복합요소(complex element)를 나타내기 위한 비단말노드는 노드명을 포함한 사각형으로 표현하고, 추상화 노드의 경우 사각형 테두리를 점선으로 표현한다. 단순요소(simple element)와 속성을 나타내기 위한 단말노드는 노드명과 함께 작은 사각형으로 표현하는데, 노드가 키 값 을 가질 경우 작은 사각형을 색을 채우고, 다중 값을 가질 경우 도메인 제약조건으로 표시하며, 리스트 값을 가질 경우 작은 사각형을 이중으로 표시하고, 내

부를 채우고, 유니온 값을 가질 경우 작은 사각형 안에 세로 바로 표시한다.

에지는 두 노드사이의 관계를 표현하는데, 단일 에지는 부모노드에서 자식노드로 방향성 선분으로 표시하고, 에지그룹은 에지그룹에 속하는 모든 형제 노드들을 방향성 선분으로 묶어 하나의 선분으로 부모노드와 연결한다.

표 1. CMXML 다이어그램 심볼
Table 1. CMXML diagram symbols

구분		심볼	비고
노드	비단말노드	name	일반 노드
		name	추상화 노드
	단말노드	name	일반 속성
		name	키 속성
		name	다중 값 속성
		name	리스트 값 속성
		name	유니온 값 속성
에지	단일에지		
	에지그룹		
라벨	generalization		total/disjoint
			total/overlapping
			partial/disjoint
			partial/overlapping
	aggregation	에지 표기와 동일(default)	
	association		
	typeof	name	자식노드는 추상화 노드
컨텐츠 모델	sequence		
	all	에지그룹 표기와 동일(default)	
	choice		
제약 조건	도메인제약조건		[0..1]
			[1..1](default)
			[0..N]
			[1..N],

라벨은 관계타입을 표현하는데, 묵시적으로는 aggregation 타입이며, 관계타입이 generalization일 경우 반원으로 표현한다. 이때, 자식노드가 부분참여(partial participation)일 경우 부모노드와의 연결선을 단일 선분으로 표시하고, 전체참여(total participation)일 경우 부모노드와의 연결선을 이중 선분으로 표시하며, 자식노드들이 분리적(disjoint) 관계일 경우 반원 안에 "X"를 넣어 표시하고, 중첩적(overlapping) 관계일 경우 반원 안에 작은 원을 넣어 구분한다. 관계타입이 association일 경우 두 노드사이를 점선으로 표시한다.

에지그룹에 적용되는 컨텐츠 모델(content model)은 부모노드와 자식노드들 사이에 분기되는 지점에 원으로 표시하는데, 이 원이 없을 경우 묵시적으로 "*all*"을 나타내고, "*sequence*"일 경우 원 안에 화살표로 표시하고, "*choice*"일 경우 원 안에 세로 바로 표시한다.

도메인 제약조건의 경우 에지 끝에 아무 표시가 없으면 "1"을, 작은 원이 표시되어 있으면 "0"을, 까마귀 발이 표시되어 있으면 "N"을 조합하여 표시한다.

3. CMXML- 텍스트 표현

개념적 모델은 논리적 모델로 사상되어야 한다. 이 단계에서의 고민은 개념적 모델에서의 추상화 레벨이다. 개념적 모델에서 너무 자세한 사항들을 포함시키면 모델의 복잡성으로 인해 이해도가 떨어지는 반면, 너무 많은 사항들을 추상화하면 논리적 모델로의 매핑이 어려워진다. 본 연구에서는 자세한 세부 정보들을 텍스트 형태로 표현하여 논리적 모델로의 사상을 자연스럽게 이루어지도록 한다. CMXML에서의 주요 텍스트 표현은 표2와 같다. 그림 1은 표 1의 다이어그램 심볼과 표 2의 텍스트 표현을 이용하여 도서와 비디오 주문 처리를 CMXML 다이어그램으로 표현한 개념적 설계 결과를 보여준다.

표 2. CMXML 텍스트 표현
Table 2. CMXML texture representation

text 표현	설명	예
Type(n)	기본타입	*Type(OrderDate)*→ date
Default(n)	디폴트 값	Default(*quantity*)→ 1
MinOccurs(n)	최소대응수	MinOccurs(*actor*)→ 2
MaxOccurs(n)	최대대응수	MaxOccurs(*phone*)→ 4
MinValue(n)	최소값	MinValue(*quantity*)→ 1
MaxValue(n)	최대값	MaxValue(*quantity*)→ 10
Fixed(n)	고정값	Fixed(*exportcode*)→ 1
Enum(n)	열거값	Enum(*state*)→ AK LA NY…
Mixed(n)	혼합 유무	Mixed(*address*)→ true

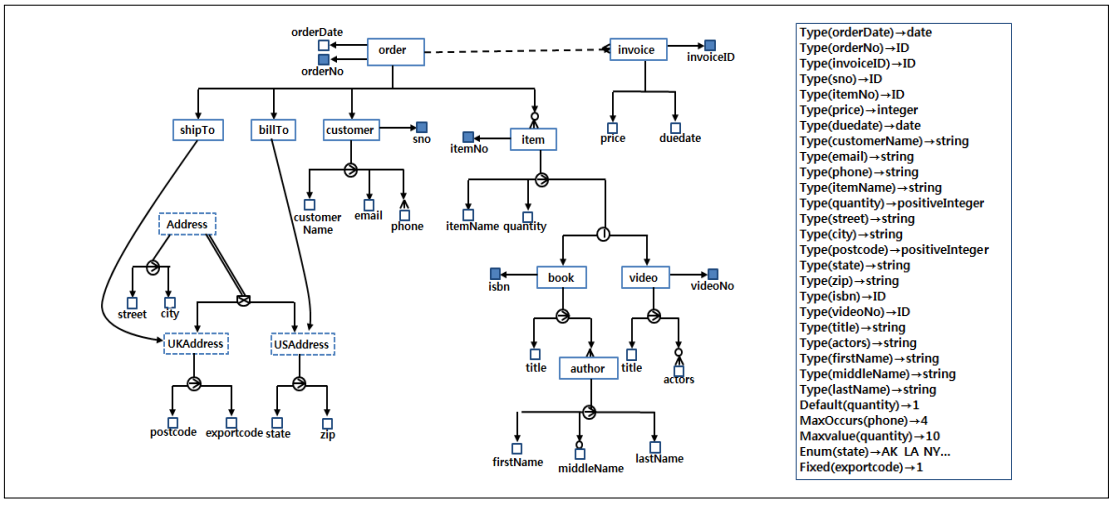

그림 1. CMXML 다이어그램 예
Fig. 1. Example of CMXML Diagram

IV. CMXML: 매핑 기법

본 장에서는 제 3장에서 개념적 모델로 제시한 CMXML 다이어그램으로 개념적 설계를 마치면 이를 토대로 논리적 모델인 XML 스키마로 매핑하는 기법을 기술한다.

1. 매핑 규칙

CMXML을 구성하는 각 요소들을 XML 스키마로 매핑하는 방법은 다양하다. 예를 들어 단말노드들은 요소로 정의하거나 속성으로 정의할 수도 있으며, 복합 타입은 해당 요소에 타입명 없이(anonymous) 직접 내포하여 표현하거나 별도의 타입으로 표현할 수도 있다. 본 연구에서는 매핑의 일관성을 위해 다음과 같은 매핑 규칙을 적용한다.

[규칙 1] 노드 명은 그대로 요소 명 또는 속성 명으로 선언한다.

[규칙 2] 비단말노드는 복합요소로 선언한다. 복합요소는 별도의 복합타입(complex type)으로 선언하며, 타입명은 복합요소명 뒤에 *Type*을 붙여 사용한다. 노드가 추상화 노드와 association 관계타입으로 연결되어 있을 경우 그 노드의 타입명은 추상화 노드명으로 한다.

[규칙 3] 단말노드는 부모노드에 내포하여 선언한다.

[규칙 4] 단말노드가 부모노드와 단순에지로 연결되고 원자 값을 가지는 경우에만 속성으로 선언하고, 그 이외에는 단순요소로 선언한다.

[규칙 5] 전역 노드와 추상화 노드는 전역적(global)으로 선언한다.

[규칙 6] 관계 타입이 generalization일 경우, 자식노드는 부모노드로부터 확장된 유도 타입(deriving type by extension)으로 선언한다.

[규칙 7] 관계 타입이 association일 경우, 키 속성이 없는 노드에(두 노드가 모두 키 속성이 있다면 임의의 노드에) 상대편 키 속성을 선언하여 IDREF(혹은 IDREFS)로 선언한다.

[규칙 8] Min, Max가 선언되어 있으면 기존 타입으로부터 제한된 유도 타입(deriving type by restriction)으로 선언한다.

2. 매핑 알고리즘

본 절에서는 앞에서 기술한 매핑 규칙을 통해 CMXML 다이어그램을 XML 스키마로 매핑하는 기법을 기술한다. 매핑 알고리즘을 위해 표 3과 같은 함수를 정의한다. 개략적인 매핑 알고리즘은 표 4와 같다. 자세한 매핑 알고리즘은 다른 논문에서 기술할 예정이다.

표 3. 매핑 알고리즘을 위한 함수
Table 3. Functions for mapping algorithm

함수	기능
EnQueue(n)	노드 n을 큐에 저장
DeQueue()	큐에서 노드를 꺼집어냄
Parent(n)	노드 n의 부모노드 반환, 없으면 NULL을 반환
MakeComplexE(Parent(n), n, Type(n), C(n))	복합요소 생성
MakeSimpleE(Parent(n), n, Type(n), C(n))	단순요소 생성
MakeAtt(Parent(n), n, Type(n), C(n))	속성 생성
CreateType(n)	복합노드 n의 복합타입을 생성

표 4. 매핑 알고리즘
Table 4. Mapping algorithm

[XML 스키마 생성 알고리즘]

Input: CMXML Diagram

Output: XML schema

```
for each global node n in N do
  Enqueue(n);
do
  n←DeQueue();
  if(n is abstract node) CreateType(n);
  else /* complex element */
  begin
    MakeComplexE(Parent(n), n, Type(n), C(n));
    CreateType(n);
  end
  for each child node c of n do
  begin
    if(c is non-terminal node)
    begin
      MakeComplexE(Parent(c), c, Type(c), C(c));
      CreateType(c);
      Enqueue(c);
    end
    else if(c is terminal node && c_value is atomic)
      MakeAtt(Parent(c), c, Type(c), C(c));
    else
      MakeSimpleE(Parent(c), c, Type(c), C(c));
  end
while(Queue is not empty);
```

그림 2는 그림 1의 CMXML 다이어그램으로부터 매핑 규칙과 매핑 알고리즘을 적용해 생성한 XML 스키마 문서를 보여준다.

```
<xs:complexType name="Address" abstract="true">
 <xs:sequence>
  <xs:element name="street" type="xs:string"/>
  <xs:element name="city"    type="xs:string"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="USAddress">
 <complexContent>
  <xs:extension base="target:Address">
   <xs:sequence>
    <xs:element name="state">
     <simpleType>
      <restriction base="xs:string">
       <enumeration value="AK"/>
       <enumeration value="LA"/>
       <! and so on .....................
      </restriction>
     </simpleType>
    <xs:element name="zip"    type="xs:string"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:complexType name="UKAddress">
 <xs:complexContent>
  <xs:extension base="target:Address">
   <xs:sequence>
    <xs:element name="postcode" type="xs:string"/>
    <xs:element name="exportcode" type="xs:positiveIntger“ fixed="1"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
<xs:element name="order" type="orderType"/>
<xs:complexType name="orderType">
 <xs:sequence>
   <xs:element name="shipTo" type="UKAddress"/>
   <xs:element name="billTo" type="USAddress"/>
   <xs:element name="customer" type="customerType"/>
   <xs:element name="item" type="itemType" minOccurs="0"
               maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="orderno" type="xs:ID"/>
 <xs:attribute name="orderDate" type="xs:date"/>
 <xs:attribute name="invoiceID" type="xs:IDREFS" use="required"/>
</xs:complexType>
<xs:element name="invoice" type="invoiceType"/>
<xs:complexType name="invoiceType">
 <xs:all>
   <xs:element name="price" type="xs:integer"/>
   <xs:element name="duedate" type="xs:date"/>
 </xs:all>
 <xs:attribute name="invoiceID" type="xs:ID"/>
</xs:complexType>
<xs:complexType name="customerType">
 <xs:sequence>
  <xs:element name=“customerName” type=“xs:string”/>
  <xs:element name="email" type="xs:string"/>
  <xs:element name="phone" type="xs:string" maxOccurs="4"/>
 </xs:sequence>
 <xs:attribute name="sno" type=“xs:ID”/>
</xs:complexType>
<xs:complexType name="itemType">
 <xs:sequence>
  <xs:element name="itemName" type="xs:string"/>
  <xs:element name="quantity" default="0">
   <xs:simpleType>
    <xs:restriction base="xs:positiveInteger">
     <xs:maxInclusive value="10">
    </xs:restriction>
   </xs:simpleType>
  <xs:choice>
   <xs:element name="book" type="bookType"/>
   <xs:element name="video" type="videoType"/>
  </xs:choice>
  <xs:attribute name="itemNo" type="xs:ID"/>
</xs:complexType>
<xs:complexType name="bookType">
 <xs:sequence>
  <xs:element name="title" type="xs:string"/>
  <xs:element name="author" type="authorType" maxOccurs="unbounded"/>
 </xs:sequence>
 <xs:attribute name="isbn" type="xs:ID"/>
</xs:complexType>
<xs:complexType name="authorType">
 <xs:sequence>
  <xs:element name="firstName" type="xs:string"/>
  <xs:element name="middleName" type="xs:string" minOccurs="0"/>
  <xs:element name="lastName" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
```

그림 2. XML 스키마 정의
Fig. 2. XML schema definition

V. 결 론

본 논문은 XML을 개념적으로 모델링할 수 있는 기법인 CMXML을 제안하였다. 이를 위해 개념적 스키마를 구성하는 각 요소들을 정의하고, 각 요소들을 그래픽으로 표현하는 방법을 제시하고, 논리적 모델로의 사상을 위해 그래픽으로 표현하기 곤란한 정보들은 텍스트 형태로 표현하였다. 또한 본 모델의 타당성을 보여주기 위해 CMXML으로 모델링한 개념적 모델을 논리적 모델인 XML 스키마로 매핑하는 기법을 제시하였다.

본 연구는 아직 프로토타입 구현 단계로서 향후 계속 진행할 연구 과제로는 본 연구의 완전성을 위해 아직 해결하지 못한 다양한 XML 개념들을 수용할 수 있는 모델로 확장하는 연구 및 XML 스키마를 CMXML으로 역공학으로 변환하는 연구가 진행할 예정이다.

References

[1] W3C, eXtensible Markup Language(XML) 1.0, http://www.w3.org/TR/1998/REC-xml-19980210, 1998.

[2] W3C, http://www.w3.org/XML/Schema, 2001.

[3] http://www.oasis-open.org/committees/relax-ng/spec-20011203.html, 2001.

[4] A. Sengupta, S. Mohan, R. Doshi, "XER-Extensible Entity Relationship Modeling", Proceedings of XML 2003, 2003.

[5] Y. Kim, "ERX: A Generation Tool of XML Schema based on Entity-Relationship Model", The Journal of The Institute of Webcasting, Internet and Telecommunication, Vol 13, No 2, 2013.

[6] M. Mani,"EReX: A Conceptual Model for XML", Proceeding s of the 2nd International XML Database Symposium(XSystem 2004), 2004.

[7] M. Necasky, et al., "When conceptual model meets grammar: A dual approach to XML data modeling", Journal of Data & Knowledge Engineering Vol 72, 2012.

[8] W. David, et al., "Enterprise Modeling with Conceptual XML", Proceeding of the 23rd International Conference on Conceptual Modeling, Vol 3288, 2004.

[9] G. Dobbie, W. Xiaoying, T. Wing, M. Lee, "ORA-SS: An Object-Relationship-Attribute Model for Semi-Structured Data", Technical Report, Department of Computer Science, National University of Singapore, 2000.

[10] L. Feng, E. Chang, T. Dillon, "A Semantic Network-Based Design Methodology for XML Documents", ACM Transactions on Information Systems, Vol 20, No 4, 2002.

[11] M. Necasky, "Conceptual Modeling for XML", Proceedings of the DATESO 2006 Annual International Workshop on Database, Texts, Specifications and Objects, 2006.

저자 소개

김 영 웅(정회원)

- 1993년 : KAIST 전산학과 박사
- 1984년 ~ 1997년 : KT 통신망 연구소
- 1997년 ~ 현재 : 한성대학교 컴퓨터공학과 교수

<주관심분야 :데이터모델링, 소프트웨어 신뢰도, 소프트웨어 설계>

※ 본 연구는 한성대학교 교내연구비 지원 과제임.